Rede Neural Difusa com T-normas Diferenciáveis e Interativas

Fabiano Ricardo de Oliveira

Dissertação de Mestrado

Orientação: Profª. Drª. Maria Bernadete Zanusso

Área de Concentração: Inteligência Artificial

Dissertação apresentada como requisito para a obtenção do título de mestre em Ciência da Computação.

Departamento de Computação e Estatística
Centro de Ciências Exatas e Tecnologia
Universidade Federal de Mato Grosso do Sul
16 de Novembro de 2006

Rede Neural Difusa com T-normas Diferenciáveis e Interativas

Este exemplar corresponde à redação final
da dissertação devidamente corrigida
e defendida por Fabiano Ricardo de Oliveira
e aprovada pela comissão julgadora.

Campo Grande/MS, 16 de Novembro de 2006.

Banca Examinadora

- Prof^{a}. Dr^{a}. Maria Bernadete Zanusso (Orientadora) (DCT-UFMS)
- Prof. Dr. Aldrovando Luís Azeredo Araújo (MTM-UFSC)
- Prof. Dr. Hemerson Pistori (CCET-UCDB)

Agradecimentos

À Prof^{a}. Dr^{a}. Maria Bernadete Zanusso por sua orientação e confiança durante todo o decorrer deste trabalho.

Aos professores Dr. Aldrovando Luís Azeredo Araújo da Universidade Federal de Santa Catarina e Dr. Hemerson Pistori da Universidade Católica Dom Bosco por suas sugestões ao trabalho.

A Hozana, minha querida noiva, pelo amor e paciência com os quais tem conduzido nossas vidas.

A minha mãe Maria e a toda a minha família pela compreensão nos momentos em que estive ausente.

Aos colegas do curso de mestrado pelas idéias, debates, questionamentos e troca de experiências que fizeram com que as disciplinas do curso fossem ainda mais interessantes.

Aos professores e funcionários do DCT-UFMS.

Finalmente, agradeço a todos os que direta ou indiretamente me ajudaram na conclusão deste trabalho.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AB	Algoritmo *Backtracking*
AD	Árvore de Decisão
ANFIS	*Adaptive Network-based Fuzzy Inference System*
	Sistema de Inferência Difusa baseada em Rede Adaptativa
EN	*Explanation Facility* Facilidade de Explanação
EQM	Erro Quadrático Médio
PMC	*Perceptron* de Múltiplas Camadas
RNA	Rede Neural Artificial
RND	Rede Neural Difusa
RNDD	Rede Neural Difusa Diferenciável
RULEX	*RULe-plus-EXception*
SQL	*Structured Query Language* Linguagem de Consulta Estruturada
TREPAN	*TREes PArroting Networks*

Resumo

Conjuntos difusos são usados na representação de conhecimentos vagos e imprecisos. Redes neurais, além de proverem paralelismo computacional, também possuem capacidade de aprendizado. A combinação desses dois paradigmas é uma tentativa de reunir os benefícios de ambos em um sistema híbrido integrado, tal como uma rede neural difusa. T-normas são funções que atuam como operadores de intersecção e união de conjuntos difusos. Na maioria dos casos, redes neurais difusas utilizam as t-normas do mínimo e do máximo para realizarem as operações de conjunção e disjunção, respectivamente. O uso de tais t-normas impossibilita a aplicação direta do algoritmo de treinamento por retropropagação de erros, baseado no cálculo do gradiente descendente, devido à inexistência da derivada do erro quadrático médio. Uma rede neural, na maioria das vezes, é incapaz de tornar explícito seu raciocínio de decisão. Existem vários algoritmos para geração de regras a partir de redes treinadas. No entanto, o volume das regras geradas por estes algoritmos pode ser muito grande, dificultando sua análise por parte do usuário. Na literatura, foram apresentadas várias medidas de avaliação e algoritmos de pós-processamento, que permitem ao usuário focalizar sua atenção nas regras que mais se destacam dentro do conjunto gerado. O objetivo principal deste projeto é aplicar t-normas diferenciáveis e interativas para realizarem as operações de conjunção e disjunção numa rede neural difusa, visando a classificação e geração de regras.

A Rede Neural Difusa Diferenciável (RNDD), objeto desta dissertação, é completamente conectada, com arquitetura alimentada adiante em três camadas: a camada de entrada difusa, a camada intermediária (composta por neurônios *e*) e a camada de saída (composta por neurônios *ou*). Utiliza o algoritmo de retropropagação para o treinamento. Para a geração de regras, será utilizado o algoritmo *backtracking*. A metodologia proposta foi avaliada em aplicações sobre o conjunto de dados *Iris*, *Vogal* e em outros três conjuntos aleatoriamente gerados, com resultados promissores de treino, teste e geração de regras. Em testes sobre o conjunto *Vogal*, a taxa de acerto alcançada foi de 80.3% e o coeficiente Kappa calculado foi de 0.771. A partir da rede treinada com os dados *Iris* e *Vogal*, foram geradas regras difusas justificando a decisão da rede. Com o desenvolvimento do sistema proposto, espera-se obter um método automático de aquisição de conhecimento a partir de exemplos de dados.

Abstract

Fuzzy sets are used in the representation of vague and imprecise knowledge. Neural networks, besides their computational parallelism, also have learning capabilities. The combination of such both paradigms is an attempt to congregate their benefits in an integrated system, such a fuzzy neural network. T-norms are functions that actuate like intersection and union operators. Most of times, fuzzy neural networks use the minimum and the maximum t-norms to perform the conjunction and disjunction, respectively. The use of such t-norms makes impossible the direct application of the backpropagation algorithm, based on the descendent gradient method, given the absence of the mean square error derivative. A neural network, most of times, does not allow the extraction of the knowledge encoded into its synaptical weights. In other words, it is not capable to make more explicit its decision reasoning. Despite this, there are several rule generation algorithms from trained neural networks. The number of rules generated by these algorithms might be very large, making hard to analise them. It has presented various evaluation measures and post-processing algorithms that allows the user to focalize his attention to the better ones in the considered rule set. The main goal of this project is to apply differentiable and interactive t-norms to perform the conjunction and disjunction operations in a fuzzy neural network, to pattern classification and rule generation.

The differentiable fuzzy neural network, denominated RNDD, object of this dissertation, is a feedforward fully-connected network with three layers: the input layer, the hidden layer of *and* neurons and the output layer of *or* neurons. It was used the backpropagation algorithm for training. Rules will be generated by the backtracking algorithm because of its intuitive nature an to be applicable to RNDD. The proposed methodology was evaluated over an application in the *Iris* and *Vowel* datasets and other ones randomly generated with promising training, test and rule generation results.

For the classification tests over *Vowel* dataset, the system reached a hit rate of 80.3%. The Kappa coefficient was calculated and the value reached was 0.771. From the trained network with the *Iris* dataset, it was generated a set of fuzzy rules justifying the network decision. After the development of the proposed system, it be expected to obtain an automatic method of knowledge acquisition from data examples.

Capítulo 1

Introdução

Recentemente, os avanços na tecnologia dos computadores asseguraram a utilização deste importante recurso a grandes porções da população mundial. Órgãos públicos, institutos de pesquisa, organizações comerciais e financeiras usam computadores não apenas para fins de processamento, mas também para armazenamento de imensos volumes de dados que são gerados diariamente ou mesmo extraídos de outras fontes, como por exemplo, a Internet. Essas bases de dados massivas podem ser reunidas em *datawarehouses*[1], isto é, repositórios de dados centralizados, construídos com intuito de facilitar a submissão de consultas e obtenção de relatórios gerenciais sobre os dados brutos.

O Descobrimento de Conhecimento em Bases de Dados[2] é definido por (Fayyad *et al.*, 1996) *apud* [KZ02] como o processo não-trivial de identificar padrões válidos, novos, potencialmente úteis e inteligíveis em bases de dados. A mineração de dados, tida como o passo principal deste processo, consiste em transformar dados brutos em conhecimento. Os dados brutos advêm de bases de dados massivas ou mesmo de um *datawarehouse*. Esses dados são integrados e pré-processados, incluindo a remoção de dados faltantes e a redução de dimensionalidade. Os dados pré-processados são então submetidos a algum algoritmo ou método de mineração de dados, que poderá ser, por exemplo, uma rede neural.

Uma rede neural pode possuir aprendizado supervisionado ou não-supervisionado. O aprendizado não-supervisionado se aplica quando não há um supervisor que já conheça alguns padrões ou instâncias[3] de dados do domínio considerado. Nesse caso, uma das tarefas de mineração de dados não-supervisionada poderá ser aplicada, por exemplo, a aglomeração. A aglomeração consiste em agrupar um conjunto selecionado de dados, assegurando a similaridade intra-grupo e a heterogeneidade inter-grupos [dOZ05a]. Uma avaliação posterior irá confirmar a representatividade deste conhecimento que estava escondido nos dados. Em resumo, este tipo de aprendizado é, via de regra, aplicado no caso em que não se tem muito conhecimento sobre os dados.

[1]Optou-se por utilizar o termo em Inglês por não ter sido encontrado o termo equivalente em Português.

[2]Do Inglês: *Knowledge Discovery in Databases*.

[3]Em toda a dissertação, as palavras padrão de entrada ou instância de dados se referem a um vetor cujas coordenadas são valores de atributos observados nos objetos mais a informação da classe à qual pertence cada um dos objetos que necessitam ser classificados.

No aprendizado supervisionado, existe um supervisor que conhece a classificação de alguns exemplos de dados que poderão ser usados para treinar e testar a rede. Por exemplo, o conjunto de dados *Iris* foi classificado pelo taxionomista Anderson [AND35]. Ao serem apresentados novos exemplos do mesmo domínio de dados à rede treinada, espera-se que ela possa classificá-los e, mediante algum algoritmo, gerar regras de decisão quanto à classificação na forma Se - Então. Essas regras irão decodificar o conhecimento aprendido a partir dos dados de treino. O presente trabalho situa-se neste contexto, i. é., na utilização de uma rede neural com aprendizado supervisionado como método de mineração de dados para as tarefas de classificação e geração de regras.

A crescente demanda por tais modelos computacionais, capazes de realizar tarefas inerentemente humanas, motivou o surgimento de uma nova classe de aplicações para classificação de conjuntos de dados, construção de modelos de classes, aglomeração e geração de regras. Conjuntos difusos[4] e Redes Neurais Artificiais (RNAs) figuram como algumas das tecnologias desta tendência quando é necessário manipular incertezas.

Conjuntos difusos freqüentemente são usados na representação de conhecimentos vagos e imprecisos. Dessa forma, podem representar, basicamente, as relações conhecidas, porém "imprecisas" entre a entrada e a saída de um sistema. O conhecimento que um especialista tem de um processo é descrito usando regras lingüísticas simples ao invés de fórmulas matemáticas precisas [YRXY01]. A teoria dos conjuntos difusos permite modelar estas regras na forma Se - Então. Esta capacidade de lidar com incertezas permitiu sua utilização em várias aplicações na área de mineração de dados [KZ02], [BKP05], entre as quais: linguagem de consulta difusa, busca por dependências funcionais[5] difusas, aglomeração difusa, etc [HUL05].

Redes neurais artificiais, por meio da interconexão de unidades de processamento simples, realizam várias tarefas inerentes ao reconhecimento de padrões e à mineração de dados, permitindo a generalização do conhecimento contido em um conjunto de dados, aproximação de funções, classificação [CM03], previsão de séries temporais, entre outras tarefas [MP02]. Uma rede neural é um modelo cujos parâmetros, ditos pesos sinápticos das conexões entre os neurônios, são obtidos por treinamento da rede a partir de um conjunto de dados e de algoritmos de aprendizado.

Uma abordagem para se conseguir atingir tanto os benefícios das redes neurais quanto dos sistemas difusos, consiste em combiná-los em um sistema integrado que possa prover o paralelismo computacional e o poder de aprendizado das redes neurais aos sistemas difusos e as capacidades de representação e inferência dos sistemas difusos às redes neurais. Redes neurais que incorporam conjuntos difusos são chamadas de Redes Neurais Difusas (RNDs) ou sistemas neurodifusos [GFP93].

A integração das RNAs com os sistemas difusos, resultando em um sistema híbrido, i. é., um sistema que combina duas ou mais técnicas distintas para resolver um dado problema, provê um modelo apropriado para a geração de regras difusas a partir de dados numéricos, [MP94], [BABZ99], [MP02], [CM03] e para a construção e otimização de mo-

[4]Do Inglês: *Fuzzy Sets*.

[5]Uma dependência funcional $X \longrightarrow Y$ existe se, em um conjunto de dados, o valor do atributo Y em uma instância for determinado pelo valor do atributo X.

delos difusos [GFP93], [CF00], [XXL05]. Em [JAN93] é apresentado o sistema *Adaptive Network-based Fuzzy Inference System* (ANFIS) que implementa um modelo difuso de inferência do tipo TSK [TS85] em uma rede neural. Os sistemas NEFCON e NEFCLASS são apresentados em detalhes por (Nauk, Klawonn e Kruse, 1997) *apud* [REZ03] e foram respectivamente projetados para controle e classificação. Outras técnicas, como a aceleração do treinamento de redes neurais por algoritmos genéticos [REZ03] e a utilização de *Rough Sets* [XXL05], podem também ser introduzidas nos sistemas híbridos, com intuito de lhes conferir maior robustez.

Em [GFP93] foi proposta uma estrutura neurodifusa, explorando um enfoque simplificado para a inferência. Dado um conjunto de regras Se - Então, a topologia da rede facilmente permite sua codificação e processamento. Esta estrutura permite que um conhecimento inicial seja facilmente embutido na rede, acelerando o aprendizado e o ajuste das regras. Inversamente, regras podem ser descobertas se um conjunto suficiente de exemplos de entrada-saída for fornecido.

T-normas e t-conormas são funções, também chamadas de operadores lógicos, que atuam respectivamente como operadores de intersecção e de união de conjuntos difusos [KY95]. Redes neurais difusas utilizam t-normas e t-conormas, na maioria dos casos, o mínimo e o máximo, propostas por Zadeh [ZAD65]. Essas t-normas não são interativas nem diferenciáveis. Quando são usadas em redes neurais difusas, a aplicação direta do algoritmo de retropropagação é impossibilitada. As t-normas utilizadas neste projeto foram propostas em [ZAN97] e, além de serem diferenciáveis, também são interativas. Uma t-norma é interativa quando seus resultados dependerem de todos os seus argumentos. A interatividade é uma característica desejável nas t-normas pois evita que para introduzi-las sejam utilizados operadores de implicação difusa [MP94].

Apesar dos bons resultados de classificação [PM92], [CM03] e de aglomeração [dOZ05a], uma rede neural, na maioria das vezes, não permite a extração do conhecimento codificado em seus pesos sinápticos de uma forma direta, i. é., é incapaz de explicar suas decisões de uma forma compreensível [REZ03]. As redes neurais aproximam uma função ou hipótese de relacionamento entre as entradas e as saídas desejadas. Uma das representações da hipótese aprendida mais expressivas e legíveis para um usuário pode ser alcançada por um conjunto de regras na forma Se - Então.

Em [GAL88] foi apresentada uma introdução aos sistemas especialistas que utilizam redes neurais treinadas como bases de conhecimento, mediante o emprego de um mecanismo para geração de regras. Tais sistemas são chamados de sistemas especialistas conexionistas. Estes sistemas podem extrair regras a partir de um conjunto de exemplos, ao invés de adquiri-las a partir de um especialista do domínio considerado. Entretanto, neste projeto, as regras geradas serão para oferecer uma justificativa para a decisão tomada pela rede neural. O consumidor final será um usuário que as lerá para compreender a decisão tomada pela rede e não uma máquina de inferência de um sistema especialista, ainda que esta também possa integrar um sistema híbrido de apoio à decisão.

Há diversos algoritmos para a geração de regras a partir de redes treinadas: *Knowledgetron* (KT) [FU94], *RULe-plus-EXception* (RULEX) [AG95], *TREes PArroting Networks* (TREPAN) [CS96], *Explanation Facility* (EN) [PT92], *Backtracking* [MP94], [BABZ99], entre outros. Estes algoritmos convertem o estado interno de uma rede treinada em regras difusas, passíveis de serem compreendidas por humanos. Por sua natureza intuitiva e visando uma comparação com os resultados reportados em [MP94], foi utilizado o algoritmo *backtracking*.

A quantidade de regras geradas por esses algoritmos pode ser muito grande, dificultando a análise do seu poder descritivo por parte do usuário. Há, na literatura, várias medidas de avaliação e algoritmos de pós-processamento de regras. As medidas de avaliação utilizam técnicas estatísticas para avaliar a capacidade de representação do conhecimento adquirido sob a forma de regras. O cálculo dessas medidas pode ser visto como um pós-processamento já que reduz as regras originais a um conjunto pequeno para que o usuário possa focalizar sua atenção sob as regras de maior qualidade. Medidas de avaliação foram calculadas para regras geradas a partir do modelo proposto.

Neste projeto, é desenvolvida uma rede neural difusa denominada Rede Neural Difusa Diferenciável (RNDD) que será usada para classificação de padrões e geração de regras difusas. A denominação RNDD deve-se ao fato de as t-normas utilizadas neste modelo de RND serem diferenciáveis e interativas. Esta pode ser considerada uma distinção em relação às RNDs encontradas na literatura que, em geral, usam t-normas não-diferenciáveis e não-interativas.

O desempenho da RNDD foi avaliado sob os conjunto de dados *Iris* [AND35], *Vogal*[6] [PM77] e três outros conjuntos gerados aleatoriamente diferentes níveis de sobreposição entre as classes consideradas. Algumas das metodologias de teste propostas por [FUK90] foram utilizadas, bem como os índices de desempenho taxa de acerto e Kappa foram calculados. Resultados de treino, teste e geração de regras serão apresentados nas Seções 5.2, 5.3 e 5.4.

1.1 Objetivos

O objetivo principal deste projeto é implementar uma rede neural difusa usando t-normas e t-conormas diferenciáveis para realizar as tarefas de classificação de padrões e de geração de regras.

Objetivos Específicos:

1. Implementar a RNDD;
2. Avaliar a sua capacidade de classificação sobre o conjunto de dados *Iris*, *Vogal* e sobre três conjuntos de dados artificialmente gerados;
3. Comparar a precisão da RNDD com a alcançada por outros modelos de redes neurais e outros classificadores, tais como árvores de decisão;
4. Gerar regras difusas a partir de seus pesos sinápticos ajustados usando o algoritmo *backtracking* para justificar a decisão tomada pela rede;

[6] Do Inglês: *Vowel*

5. Comparar os resultados de classificação e geração de regras do conjunto *Vogal* com os resultados de [MP94];

6. Fazer pós-processamento das regras geradas a partir do conjunto *Iris*, utilizando medidas de avaliação de regras;

1.2 Justificativa para o Sistema Proposto

A principal justificativa para a utilização de um sistema híbrido neurodifuso está em aliar a capacidade das redes neurais de aprender a partir de exemplos, com a capacidade dos conjuntos difusos de modelar e manipular as incertezas ou ambigüidades tão presentes no domínio das aplicações reais, como por exemplo, a sobreposição de classes, ou seja, a pertinência de uma determinada instância de dados a mais de uma classe, ou mesmo o uso de termos lingüísticos como *baixo*, *médio* e *alto*.

Obter-se uma base de regras de um determinado domínio, única e exclusivamente a partir de dados, é uma tarefa difícil, principalmente quando não se tem um especialista com experiência suficiente sobre os dados a partir dos quais, aplicando-se técnicas de engenharia do conhecimento, poder-se-ia extrair regras. A capacidade das redes neurais de processar dados e codificar seu conhecimento em seus pesos sinápticos, aliada a um algoritmo para extrair regras, permite a possibilidade de produção de bases de conhecimento a partir de bases de dados. Se estas regras se mostrarem válidas, o sistema apresenta uma grande vantagem, pois segundo [GAL88] a construção de bases de conhecimento está entre as tarefas mais dispendiosas no desenvolvimento de um sistema especialista. No entanto, o enfoque deste projeto está na geração de regras como uma justificativa para a decisão tomada pela rede. Esta justificativa torna essa decisão mais aceitável para o usuário, visto que explicita o raciocínio da rede.

Apesar de serem consideradas modelos de fácil compreensão e aplicação, a dificuldade de análise de um elevado número de regras pode tornar proibitivo o entendimento do conhecimento extraído. A idéia é ter-se um pós-processamento das regras geradas, reduzindo-as a um pequeno conjunto de regras mantendo a qualidade das regras originais e facilitando o seu entendimento por parte do usuário.

Muitas são as possibilidades de aplicação dos sistemas neurodifusos com geração de regras, tanto no contexto acadêmico quanto no comercial, financeiro, de transporte, médico, etc. Entre estas aplicações encontram-se: o reconhecimento da fala [MP94], a classificação de casos reais de crises epiléticas [BABZ99], o diagnóstico de doenças hepatobiliares a partir de dados médicos [MP02], a classificação de modos de propagação de ondas de rádio [CM03], a seleção de fornecedores de suprimentos para indústrias automobilísticas [ZTR04] e, mais recentemente, o auxílio à descoberta e projeto de novas drogas contra a AIDS e outras doenças [AFAC$^+$05].

Com o desenvolvimento do sistema proposto, espera-se obter: primeiro, uma rede neural difusa com aprendizado supervisionado que classifique bem instâncias de uma base de dados; segundo, um método para geração de regras Se - Então que explique as decisões de classificação tomadas pela rede, já que existem casos em que essa justificativa é extremamente necessária, e.g.: diagnóstico médico. Dessa forma, espera-se poder aplicar esta rede para classificar bases de dados reais que apresentem problemas de sobreposição de classes e que dela se possa gerar regras Se - Então que torne o raciocínio de decisão da rede mais explícito para o usuário.

1.3 Metodologia

O objeto de estudo deste projeto, a RNDD, consiste de uma RND completamente conectada, com arquitetura alimentada adiante, em três camadas: a camada de entrada difusa, a camada intermediária *e* e a camada de saída *ou*, utilizando t-normas diferenciáveis e interativas, propostas em [ZAN97]. A descrição mais detalhada da RNDD está na Seção 3.3 e das t-normas, na Seção 2.4.

Os valores reais de cada atributo de entrada, que são medidas diretas dos objetos a serem classificados, serão pré-processados por um difusificador, de maneira que a cada valor de atributo seja associado graus de pertinência aos conjuntos difusos *baixo*, *médio* e *alto*. Estas serão as entradas difusas da rede. Este pré-processamento e a forma das funções de pertinência utilizadas serão mostrados na Seção 3.3.1.

A camada *e* é formada por neurônios que executam a conjunção ou intersecção difusa, usando t-normas e a camada *ou* é formada por neurônios que executam a disjunção ou união difusa, usando t-conormas. A camada *ou* é a camada de saída da rede e terá tantos neurônios quantas forem as classes do conjunto de dados considerado. A saída será formada por graus de pertinência aos conjuntos difusos que definem as classes. A topologia da rede é mostrada na Seção 3.3.

A aplicação do algoritmo de aprendizagem em um perceptron difuso, que é o modelo de rede deste trabalho, não é tão simples como no perceptron convencional porque, usualmente, as t-normas e t-conormas usadas como funções de ativação para as unidades processadoras da rede não são diferenciáveis, por exemplo as t-normas e t-conormas que se baseiam no máximo e no mínimo. Deste modo, a rigor, o método do gradiente descendente não pode ser usado.

Neste projeto, será empregado o algoritmo de aprendizado supervisionado por retropropagação[7], cuja derivada da função de Erro Quadrático Médio (EQM) foi calculada derivando as t-normas e t-conormas descritas na Seção 2.4 e que foram usadas no lugar do mínimo e máximo e do produto e soma utilizados por [MP94]. A derivada do erro das t-normas utilizadas neste projeto está descrita na Seção 3.3.

O modo de treinamento utilizado será o seqüencial, no qual o conjunto dos pesos é atualizado a cada apresentação dos padrões de dados. Este modo de treinamento além de requerer menos armazenamento local para cada conexão sináptica e ser de mais fácil implementação, também possibilita a apresentação dos padrões de maneira aleatória a cada época[8], tornando a busca no espaço de pesos de natureza estocástica. Isto diminui

[7] Do Inglês: *backpropagation*.

[8] Uma época corresponde a passagem de todo o conjunto de treinamento pela rede.

a probabilidade de que o algoritmo de retropropagação fique oscilando em torno de um mínimo local [HAY99]. Esta função foi implementada para a rede RNDD.

Para ajudar a evitar o problema dos mínimos locais e, eventualmente, prevenir oscilações do EQM no espaço dos pesos durante o processo de convergência para uma solução com erro mínimo, será utilizada uma heurística consistindo de diminuições gradativas, em passos discretos, da taxa de aprendizagem e do momento [MP94]. Na implementação, são escolhidos conjuntos de valores para ambos os parâmetros. Esses valores são colocados em ordem decrescente e utilizados seqüencialmente no treinamento da rede a cada período pré-determinado de épocas.

Um dos critérios para se avaliar o desempenho de um método de classificação é usar a comparação da classificação feita por um especialista no domínio de dados considerado com os resultados do classificador artificial para derivar a sua precisão. A matriz de confusão permite que várias medidas de precisão da classificação sejam calculadas. As medidas de precisão escolhidas, neste projeto, serão a taxa de acerto e o coeficiente Kappa que foi dado por [COH60], apresentado na Seção 5.1. A taxa de acerto é dada pela soma dos elementos diagonais da matriz de confusão e é usada como medida padrão para avaliar classificadores. O coeficiente Kappa é recomendado como uma medida de precisão aceitável na temática da classificação, já que é calculado a partir da matriz de confusão inteira. O cálculo destas medidas de precisão permite a comparação com resultados obtidos por outros classificadores e reportados na literatura.

Os conjuntos de dados usados para testar a RNDD foram o *Iris*, apresentado na Seção 5.2, o *Vogal*, apresentado na Seção 5.3 e outros três conjuntos aleatoriamente gerados, descritos na Seção 5.4. O *Iris* foi escolhido por representar um problema real e ser bastante explorado na literatura de classificadores, com taxas de acerto publicadas. Já o *Vogal* foi utilizado para permitir uma comparação com [MP94]. Os três conjuntos aleatórios foram utilizados para avaliar a capacidade da rede de classificar dados com diferentes graus de sobreposição de classes variando o valor do parâmetro $Sobrep$ das funções de pertinência, explicado na Seção 3.3.1. Os dados foram divididos aleatoriamente em dois conjuntos: o de treino e o de teste. Em ambos os conjuntos, os padrões serão proporcionalmente distribuídos nas classes consideradas. Após o teste, as medidas de precisão serão calculadas. Os testes de classificação serão apresentados nas Seções 5.2, 5.3 e 5.4.

As saídas da rede são graus de pertinência às classes consideradas. O modo pelo qual serão interpretados os resultados da saída da rede treinada durante a classificação dos padrões de teste será o do "vencedor-leva-tudo": a máxima saída do j-ésimo neurônio denota a pertinência à j-ésima classe considerada. Na Figura 1.1, é mostrado o diagrama simplificado do sistema proposto.

Três algoritmos para a geração de regras a partir da rede treinada são apresentados no Capítulo 4. No entanto, pareceu razoável considerar-se o algoritmo *backtracking* [9], utilizado em [MP94] e [BABZ99], devido à sua natureza intuitiva e por ser aplicável à RNDD. A escolha também se deu para permitir uma comparação com os resultados de geração de regras obtidos em [MP94].

[9] Optou-se por utilizar o termo em Inglês por não ter sido encontrado o termo equivalente em Português.

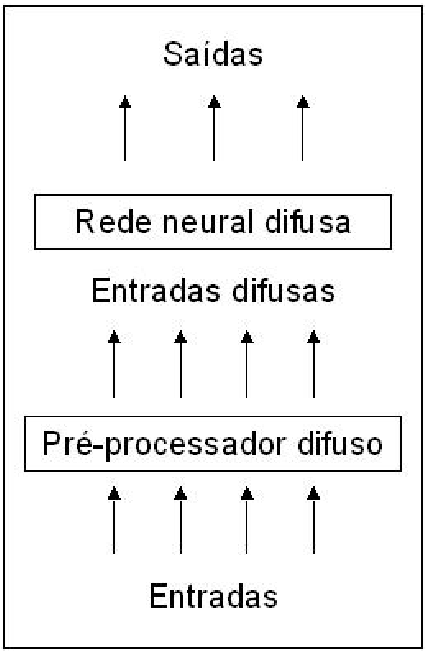

Figura 1.1: Diagrama simplificado da RNDD.

Foram apresentadas as medidas de avaliação de regras: confiança, suporte e novidade. Essas medidas utilizam técnicas estatísticas para avaliar a capacidade de representação do conhecimento adquirido sob a forma de regras e podem ser utilizadas para efetuar uma análise de como essas regras são sustentadas pelos exemplos, i. é., podem ser utilizadas para efetuar uma análise qualitativa das regras geradas (Horst, 1999; Horst e Monard, 2000) *apud* [GOM02]. Também são mostrados dois métodos de pós-processamento: por poda [TKR$^+$95] e por estimativa da interessabilidade [HLL99]. As medidas de avaliação apresentadas foram calculadas para as regras extraídas da rede treinada com o conjunto de dados *Iris*, na Seção 5.2.3.

1.4 Organização do Texto

Os capítulos que compõem este trabalho são sucintamente descritos a seguir:
No Capítulo 2, é apresentada uma pequena revisão dos conceitos relativos à teoria dos conjuntos difusos além de introduzir a família de t-normas que será utilizada na implementação deste projeto. No Capítulo 3, são expostos os conceitos básicos acerca das redes neurais artificiais, em particular das difusas, trazendo uma breve revisão do tema e a apresentação de uma rede neural difusa baseada em t-normas diferenciáveis: a RNDD. No Capítulo 4, são comentados três algoritmos de extração de conhecimento a partir de RNAs treinadas: o TREPAN, o EN e o *backtracking*. No Capítulo 4.2, são apresentadas algumas das medidas de avaliação de regras. Dois métodos de pós-processamento de regras são apresentados. No Capítulo 5, o conjunto de dados *Iris*, *Vogal* e três outros conjuntos aleatoriamente gerados são apresentados. Também são expostos resultados de treino, de teste e de geração de regras, bem como são calculadas medidas de avaliação

para as regras geradas. Finalmente, no Capítulo 6, são apresentadas algumas conclusões e uma avaliação dos resultados obtidos. Finalmente, no Apêndice, são mostrados alguns aspectos relativos à implementação da RNDD.

Capítulo 2

Conjuntos Difusos

Imprecisão e incerteza têm uma história bem estabelecida com conceitos estudados por filósofos, psicólogos, lingüistas e, mais recentemente, por matemáticos e cientistas da computação. Em 1965 houve um maior interesse neste tema depois que o engenheiro eletrônico Lotfi A. Zadeh [ZAD65] lançou as bases da teoria dos conjuntos difusos[1]. A proposta original define os operadores máximo e mínimo como os operadores de união e intersecção de conjuntos difusos.

Em geral, obter-se um modelo matemático e completo de um sistema complexo é uma tarefa difícil, seja pelo número das variáveis envolvidas ou mesmo pela incerteza em seus parâmetros. Além disso, foi provado que um modelo difuso pode aproximar qualquer mapeamento não-linear entre entrada e saída [KEE94]. Isso justifica sua utilização na modelagem de problemas difíceis de se definir por modelos matemáticos.

Dessa forma, os conjuntos difusos podem representar, basicamente, as relações conhecidas, porém "imprecisas" entre a entrada e a saída de um sistema. O conhecimento que um especialista tem de um processo é sempre descrito em regras lingüísticas ao invés de funções matemáticas precisas [YRXY01]. Tal perspectiva não permite a definição precisa de uma solução, mas conduz a uma classificação em categorias gerais das possíveis soluções.

Alguns tópicos de interesse para a teoria e as aplicações dos conjuntos difusos foram discutidos por [GOG68]. Dificuldades resultantes da imprecisão na linguagem foram notadas por cientistas da computação [WIN72] bem como por lingüistas [LAK73]. Esta capacidade de lidar com incertezas permitiu sua utilização em várias aplicações nas áreas de tomada de decisão e de mineração de dados [BKP05], entre as quais: linguagens de consulta difusa, busca por dependências funcionais utilizando conjuntos difusos, aglomeração difusa, etc. [HUL05]. O amplo espectro de aplicações é derivado da habilidade dos conjuntos difusos de estabelecer conclusões baseadas em conhecimento vago e impreciso, habilidade esta muito parecida com a dos seres humanos, permitindo a construção de sistemas de controle de uma maneira natural e simples [REZ03].

[1]O termo *fuzzy* não tem um tradução definida e consensual para o português, mas é certo que uma boa tentativa passa pelo "difuso", "nebuloso", "vago" e "incerto". Em face deste pequeno problema de escolha de nomenclatura, procurar-se-á adotar o termo "difuso" como padrão.

O objetivo deste capítulo é apresentar uma revisão dos conceitos relativos à teoria dos conjuntos difusos, além de apresentar as t-normas que serão utilizadas na implementação da Rede Neural Difusa Diferenciável (RNDD), objeto de estudo deste projeto.

2.1 Definições Básicas

A função característica de um conjunto A, $A : \mathbf{X} \longrightarrow \{0,1\}$ induz uma restrição com um limite bem definido sobre os elementos de um domínio $\mathbf{X}$ que podem ser atribuídos ao conjunto A. Os conjuntos clássicos são construções bivaloradas, i. é, seus elementos podem assumir os valores *falso* ou *verdadeiro*. Nessa acepção, um elemento é totalmente admitido como pertencente ou não-pertencente a um certo conjunto.

A idéia principal dos conjuntos difusos é relaxar essa exigência dicotômica e permitir valores intermediários de pertinência às classes. Essa idéia permite uma interpretação mais realística de um modelo, na medida em que admite sentenças com pertinências parciais. Curiosamente, a maioria das categorias usadas na descrição de objetos do mundo real não possuem limites de pertinência bem definidos [CPS98]. Nessas circunstâncias, a pertinência de um objeto a uma determinada categoria será dada por um grau, tipicamente um número real definido no intervalo unitário $[0,1]$. Quanto mais próximo de 1, maior será o nível de compatibilidade do objeto à categoria considerada.

Um conjunto difuso [ZAD65][2] é caracterizado por uma função de pertinência que mapeia os elementos de um domínio ou universo de discurso para o intervalo unitário $[0,1]$, ou seja

$$A : \mathbf{X} \longrightarrow [0,1] \tag{2.1}$$

Na verdade, um conjunto difuso A, definido em $\mathbf{X}$, pode ser representado na sua forma analítica por um conjunto de pares ordenados, separados por vírgulas, sendo cada par formado por um elemento genérico $x \in \mathbf{X}$ e seu respectivo grau de pertinência $A(x)$, tal que $A = \{(x, A(x)) | x \in \mathbf{X}\}$.

Um conjunto difuso é uma generalização de um conjunto clássico cuja função de pertinência assume apenas os valores $\{0,1\}$. O valor de $A(x)$ descreve o grau de pertinência de um elemento x em A. Conjuntos difusos podem ser definidos tanto em universos finitos discretos, quanto em infinitos contínuos.

2.2 Funções de Pertinência

A noção de pertinência parcial pode ser capturada de diversas maneiras, dada a utilização de uma ou outra função de pertinência. As funções de pertinência podem ser triangulares, trapezoidais, em forma de S, gaussianas, etc. Geralmente, possuem domínio real com contradomínio em $[0,1]$. Neste trabalho, o contradomínio das funções será o intervalo $[-1,1]$.

[2]As expressões conjunto difuso e conceito difuso serão usadas neste texto com o mesmo significado.

Para efetuar a comparação deste trabalho com a rede neural difusa proposta por Mitra e Pal [MP94] foi usada a mesma função de pertinência que estes autores usaram. A expressão da função é definida por:

$$A(x) = \begin{cases} 2(1 - \frac{|x-c|}{\lambda})^2 & \text{se } \frac{\lambda}{2} \leq |x-c| \leq \lambda \\ 1 - 2(\frac{|x-c|}{\lambda})^2 & \text{se } 0 \leq |x-c| \leq \frac{\lambda}{2} \\ 0 & \text{caso contrário} \end{cases} \tag{2.2}$$

onde $x \in R$, $A(x) \in (0,1)$, $\lambda > 0$ é o raio da função e c é o centro. Esta será a função de pertinência utilizada na implementação da RNDD na fase de pré-processamento que será mostrada na Seção 3.3.1. Na Figura 3.3 é mostrado o gráfico da função.

2.3 Operações sobre Conjuntos Difusos

As funções de pertinência são representações dos conjuntos difusos. As operações básicas: união, intersecção e complemento sobre conjuntos difusos podem ser representadas pelo máximo, mínimo e complemento [ZAD65] das funções de pertinência correspondentes para todo $x \in \mathbf{X}$, ou seja:

$$\begin{aligned} (A \cup B)(x) &= max(A(x), B(x)) = A(x) \vee B(x) \\ (A \cap B)(x) &= min(A(x), B(x)) = A(x) \wedge B(x) \end{aligned}$$

onde A e B são conjuntos definidos em um universo $\mathbf{X}$, dito universo de discurso, e $(A \cup B)(x)$, $(A \cap B)(x)$ denotam, respectivamente, a união e a intersecção de A e B.

O objetivo desta seção é introduzir as operações que serão realizadas nas unidades processadoras ou neurônios do modelo de rede neural difusa diferenciável proposto neste trabalho.

2.3.1 T-normas

Conforme Menger [MEN42], as t-normas são funções que surgem naturalmente no estudo das desigualdades triangulares para espaços métricos probabilísticos. Ele foi o primeiro a usar a denominação norma triangular, chamada resumidamente de t-norma. Schewizer [SS63] afirmou a importância de se ter um grande repertório de t-normas e foi quem estabeleceu uma caracterização que permite construí-las à vontade. Klir [KY95] faz uma revisão dos operadores de intersecção e união difusa colocando as t-normas e t-conormas como funções qualificadas para efetivá-las. Dessa forma, as t-normas são aplicadas sobre graus de pertinência a conjuntos difusos.

A teoria dos conjuntos difusos tem tido inúmeras e variadas aplicações, colocadas nos mais diferentes ramos do conhecimento nos últimos anos. As operações-padrão de intersecção e união de conjuntos difusos foram definidas por Zadeh [ZAD65] de modo a se reduzirem à definição usual de intersecção e união quando os graus de pertinência estão no conjunto $\{0, 1\}$. Em outras palavras, elas são generalizações das correspondentes operações clássicas sobre conjuntos ordinários. Mas estas não são as únicas generalizações possíveis. Muitas famílias de operadores com extensas propriedades foram formuladas. O problema da escolha das funções de pertinência e dos operadores nas aplicações foi tema de vários trabalhos na área [DOM82], [DP88], [AZM97], [ZAN97], [ZAM97].

O operador *max* é a t-conorma mais empregada para a união, enquanto que o operador *min* é a t-norma mais empregada para a intersecção. Outras t-normas são mostradas na Tabela 2.1, onde a e b são graus de pertinência aos conjuntos difusos A e B, respectivamente. Os números p, w e α são parâmetros. Uma tabela mais completa de t-normas é apresentada por Klir em [KY95].

Tabela 2.1: T-normas e t-conormas para dois conjuntos difusos.

Denominação	T-norma	T-conorma
Zadeh	$min(A(x), B(x))$	$max(A(x), B(x))$
Schweizer e Skalar 2	$1 - [(1-A(x))^p + (1-B(x))^p - (1-A(x))^p(1-B(x))^p]^{\frac{1}{p}}$	$[A(x)^p + B(x)^p - A(x)^pB(x)^p]^{\frac{1}{p}}$
Yager	1 - $min\{1, [(1-A(x))^w + (1-B(x))^w]^{\frac{1}{w}}\}$	$min\{1, (A(x)^w + B(x)^w)^{\frac{1}{w}}\}$
Dubois e Prade	$\frac{A(x)B(x)}{max(A(x),B(x),\alpha)}$	$1 - \frac{(1-A(x))(1-B(x))}{max((1-A(x)),(1-B(x)),\alpha)}$
Produto	$A(x)B(x)$	$[A(x) + B(x)] - [A(x)B(x)]$

Estas classes de operações de união e intersecção difusas permitem muitas alternativas de aplicação e avaliação de suas propriedades dentre de um determinado contexto. Assim, não só a função de pertinência de conjuntos de difusos, mas também as operações sobre eles são dependentes do contexto. A capacidade de determinar funções de pertinência apropriadas a operações difusas significativas para o contexto de cada aplicação particular é muito importante para a prática da teoria dos conjuntos difusos [ZAN97]. A seguir serão dadas as definições de t-norma[3] e t-conorma.

Definição 2.1 (T-norma) *Uma função $T : [-1,1] \times [-1,1] \rightarrow [-1,1]$ é uma norma triangular (ou t-norma) se*

*(**T1**)* $T(-1,-1) = -1, T(a,1) = T(1,a) = a$ *(condições limite)*
*(**T2**)* $T(a,b) \leq T(c,d)$ *quando* $a \leq c$ *e* $b \leq d$ *(monotonicidade)*
*(**T3**)* $T(a,b) = T(b,a)$ *(comutatividade)*
*(**T4**)* $T(T(a,b),c) = T(a,T(b,c))$ *(associatividade)*
Se além disto, T satisfizer
*(**T5**)* T *é contínua*
*(**T6**)* $T(a,a) < a$, $a \neq -1$ *e* $a \neq 1$
*T denomina-se t-norma arquimediana. Finalmente, se a condição (**T2**) é estrita, quando $a < c$ e $b < d$, T denomina-se t-norma arquimediana estrita.*

Definição 2.2 (T-conorma) *Uma função $S : [-1,1] \times [-1,1] \rightarrow [-1,1]$ é uma t-conorma se*

[3]Geralmente, na literatura de conjuntos difusos são usadas as notações T para t-norma e S para t-conorma

*(**S1**) $S(1,1) = 1, S(a,-1) = S(-1,a) = a$ (condições limite)*
*(**S2**) $S(a,b) \leq S(c,d)$ quando $a \leq c$ $b \leq d$ (monotonicidade)*
*(**S3**) $S(a,b) = S(b,a)$ (comutatividade)*
*(**S4**) $S(S(a,b),c) = S(a,S(b,c))$ (associatividade)*
Se, além disto, S satisfizer:
*(**S5**) S é contínua*
*(**S6**) $S(a,a) > a$, $a \neq -1$ e $a \neq 1$*
*S denomina-se t-conorma arquimediana. Finalmente, se a condição (**S2**) é estrita, quando $a < c$ e $b < d$, S denomina-se t-conorma arquimediana estrita.*

Parece mais natural expressar um grau de pertinência por um valor no intervalo $[-1,1]$ do que no intervalo $[0,1]$ que é prática comum na teoria de conjuntos difusos. Em [PM92], o intervalo de pertinência considerado é $[0,1]$ e pode-se observar o uso do termo *grau de pertinência positivo*, usado para designar graus de pertinência maiores que 0.5, tido como o valor mais ambíguo de todo o intervalo considerado. Alternativamente, ao se considerar o intervalo $[-1,1]$, o termo *grau de pertinência* parece ter um significado mais definido, já que neste intervalo faz sentido considerar que valores negativos denotam *graus de pertinência negativos*; e valores positivos, *graus de pertinência positivos*. O zero seria considerado, mais convenientemente, como o valor mais ambíguo de todo o intervalo.

O operador *min* é uma t-norma e o operador *max* é uma t-conorma. Esses operadores correspondem, respectivamente, à intersecção e à união de conjuntos, sejam eles difusos ou não. Estão entre algumas das t-normas freqüentemente encontradas na literatura o produto e o operador conjuntivo de Lukasiewicz [KY95]. Entre as t-conormas, está a soma probabilística e o operador disjuntivo de Lukasiewicz.

A seleção de uma determinada t-norma é definida por dois critérios: a interatividade entre os conjuntos difusos e a complexidade computacional. Se não houver interatividade entre os conjuntos, os operadores *min* e *max* poderão servir como bons operadores lógicos. Se houver interatividade, a escolha de t-normas e t-conormas interativas deverá ser favorecida. Por exemplo, a semântica dos conjuntos difusos *caro* e *rápido*, usados para caracterizar um veículo, sugere que estes conjuntos interagem. Na verdade, o resultado da combinação pode incorporar o fato destes dois conjuntos interagirem (espera-se que um veículo rápido seja também caro). Em oposição, a combinação dos conjuntos *novo* e *vermelho* envolve dois pedaços de informação que são altamente não-interativos (um veículo novo não é, necessariamente, vermelho) [CPS98].

Sobre o aspecto computacional, a falta de interação será bem representada pelos operadores *min* e *max*. Nota-se que $min(x,a)$ retornará x se x for menor que a e este resultado não reflete o valor de a. Isto não ocorre no caso do produto cujo resultado depende de ambos os valores dos dois argumentos.

2.3.2 Significados das propriedades de t-norma e t-conorma

A seguir será feita uma série de considerações sobre o significado de cada axioma, para que a t-norma seja considerada intuitivamente aceitável como intersecções difusas significativas para quaisquer pares de conjuntos difusos.

1. A primeira condição para t-norma e t-conorma estabelece que as operações de intersecção e união para conjuntos difusos sejam extensões das mesmas operações clássicas sobre os conjuntos ordinários, pois estas operações se reduzem às operações não-difusas quando os graus de pertinência estão restritos ao conjunto $\{0, 1\}$. Também por isto são denominados operadores lógicos Dombi [DOM82]. Quando um argumento de T é 1, expressando pertinência total, as condições-limite e a comutatividade também asseguram, como a concepção intuitiva requer, que o grau de pertinência na intersecção seja igual ao do outro argumento.

2. A monotonicidade e a comutatividade expressam a exigência natural de que um decrescimento no grau de pertinência no conjunto A ou B não pode produzir um aumento no grau de pertinência da intersecção.

3. A comutatividade assegura que a intersecção difusa é simétrica, i. é., indiferente quanto à ordem em que os conjuntos a serem combinados são considerados.

4. A associatividade assegura que se pode tomar a intersecção para qualquer número de conjuntos em qualquer ordem de agrupamento pareada desejada. Esta operação permite estender a intersecção difusa para além de dois conjuntos. O grau de pertinência independe do agrupamento dos conceitos.

5. A continuidade impede que ocorra uma situação em que uma mudança muito pequena no grau de pertinência do conjunto A ou do conjunto B produza uma grande (descontínua) mudança no grau de pertinência de $A \cap B$.

6. A subidempotência expressa que o grau de pertinência ao conjunto difuso $A \cap B$ neste caso especial não pode exceder a. Se se exigisse a idempotência, isto significaria que a conjunção de conjuntos iguais resultaria no mesmo conjunto.

7. A monotonicidade estrita é uma forma mais forte de monotonicidade.

2.3.3 Força da intersecção e da união

Se T e S são t-norma e t-conorma arquimedianas estritas então $T(a, b) \leq \inf\{a, b\}$ e $S(a, b) \geq \sup\{a, b\}$. Dessa forma, sabe-se que a intersecção difusa padrão produz, para quaisquer conjuntos difusos dados, o maior conjunto difuso entre todos aqueles produzidos por todas as possíveis intersecções difusas (t-normas). Esta é a intersecção mais fraca. Por outro lado, a união difusa padrão produz o menor conjunto difuso entre os conjuntos difusos produzidos por todas as possíveis uniões difusas (t-conormas). Esta é a união mais forte.

Por exemplo, seja $A = \textit{inteligente}$ e $B = \textit{bonita}$. O grau de pertinência de uma menina ao conceito determinado pela conjunção $A\,e\,B$ é maior quando calculado pelo operador *min* do que por qualquer outra t-norma. Assim, a força do *e* é menor no caso do mínimo.

Nota-se que, pela declaração lógica $A\,e\,B$, é requerida a satisfação simultânea de A e de B. É um fenômeno comum na linguagem falada enfatizar o *e* quando se requer forte satisfação destas duas condições. Assim, pela *força* de um *e* se quer expressar o quão forte se quer esta satisfação simultânea. Raciocínio semelhante pode ser aplicado à união difusa.

2.4 Famílias de T-Normas Diferenciáveis

Para operar com conjuntos difusos, em [ZAN97] foram propostas famílias de t-normas e t-conormas diferenciáveis (uma t-conorma também é uma t-norma). Os operadores de conjunção e disjunção para conceitos difusos foram originalmente formulados por [MAI91]. Considerando φ_1 e φ_2 funções de pertinência quaisquer, foram definidos

$$C(\varphi_1 : q_1 : w_1; \varphi_2 : q_2 : w_2) = \frac{1-y}{1+y}, \tag{2.3}$$

onde

$$y = \left(\frac{q_1}{z_1}\right)^{w_1} + \left(\frac{q_2}{z_2}\right)^{w_2} \tag{2.4}$$

e

$$D(\varphi_1 : q_1, w_1; \varphi_2 : q_2 : w_2) = \frac{y-1}{y+1}, \tag{2.5}$$

onde

$$y = \left(\frac{z_1}{q_1}\right)^{w_1} + \left(\frac{z_2}{q_2},\right)^{w_2} \tag{2.6}$$

com $z_i = \frac{1+\varphi_i}{1-\varphi_i}$, $i = 1, 2$.

Entretanto, como demonstrado em [ZAN97] os operadores C e D, não são operadores conjuntivos e disjuntivos propriamente ditos, no sentido que caracterizam interseção e união de conjuntos difusos.

Ao estudar estes operadores, observa-se que se definidas as funções

$$Op_1 : (-1, +\infty) \to [-\infty, 1)$$

$$Op_2 : [-1, 1) \to [0, +\infty)$$

$$Op_3 : (-1, +\infty) \to (-1, +\infty)$$

dadas por

$$Op_1(x) = \frac{x-1}{1+x} \tag{2.7}$$

$$Op_2(x) = \frac{1+x}{1-x} \tag{2.8}$$

$$Op_3(x) = \frac{1-x}{1+x}, \tag{2.9}$$

facilmente pode-se ver que

$$C(\varphi_1 : q_1 : w_1; \varphi_2 : q_2 : w_2) = Op_3\left((q_1 Op3(\varphi_1))^{w_1} + (q_2 Op3(\varphi_2))^{w_2}\right) \tag{2.10}$$

e

$$D(\varphi_1 : q_1 : w_1; \varphi_2 : q_2 : w_2) = Op_1\left((q_1 Op2(\varphi_1))^{w_1} + (q_2 Op2(\varphi_2))^{w_2}\right). \tag{2.11}$$

Tomando-se $q_1 = q_2 = w_1 = w_2 = 1$, verifica-se que os operadores definidos deste modo se caracterizam como t-normas e t-conormas conforme o teorema de caracterização [SS63]. As linhas gerais para esta demostração também podem ser encontradas em Zanusso [ZAN97] , [ZAM97] e Araújo [AZM97].

2.4.1 Definição das famílias

As definições a seguir foram formuladas em [ZAN97]. Os parâmetros iguais a 1 foram omitidos para simplificação da notação e os graus de pertinência a conjuntos (ou conceitos) difusos foram denotados por a e b.

Definição 2.3 (Família de t-normas) *Por definição, se* $a, b \in (-1, 1]$

$$U_c^{\alpha,\beta}(a,b) \stackrel{\text{def}}{=} f_{\alpha,\beta}^{-1}\left(f_{\alpha,\beta}(a) + f_{\alpha,\beta}(b)\right), \alpha, \beta > 0. \tag{2.12}$$

$$a = 1 \Leftrightarrow U_c^{\alpha,\beta}(a,b) = b$$
$$a = -1 \Leftrightarrow U_c^{\alpha,\beta}(a,b) = -1$$

onde $f_{\alpha,\beta} \colon (-1, 1] \to [0, +\infty)$ *é definida via*

$$f_{\alpha,\beta}(x) = \alpha \left(\frac{1-x}{1+x}\right)^{\beta} \tag{2.13}$$

Definição 2.4 (Família de t-conormas) *Por definição, se* $a, b \in [-1, 1)$

$$U_d^{\alpha,\beta}(a,b) \stackrel{\text{def}}{=} g_{\alpha,\beta}^{-1}\left(g_{\alpha,\beta}(a) + g_{\alpha,\beta}(b)\right) \tag{2.14}$$

$$a = 1 \Leftrightarrow U_d^{\alpha,\beta}(a,b) = 1$$
$$a = -1 \Leftrightarrow U_d^{\alpha,\beta}(a,b) = b$$

onde $g_{\alpha,\beta} \colon [-1, 1) \to [0, +\infty)$ *é definida via*

$$g_{\alpha,\beta}(x) = \alpha \left(\frac{1+x}{1-x}\right)^{\beta} \tag{2.15}$$

As famílias $U_c^{\alpha,\beta}$, $U_d^{\alpha,\beta}$ são t-norma e t-conorma arquimedianas estritas, respectivamente.

Na Figura 2.1, é mostrado que conforme os valores de α e de β são variados é dada maior ou menor força para a t-norma. Na cor verde estão as funções de pertinência definidas por $A(x) = \frac{x^k - p^k}{x^k + p^k}$, com $p > 0$ e $k \neq 0$.

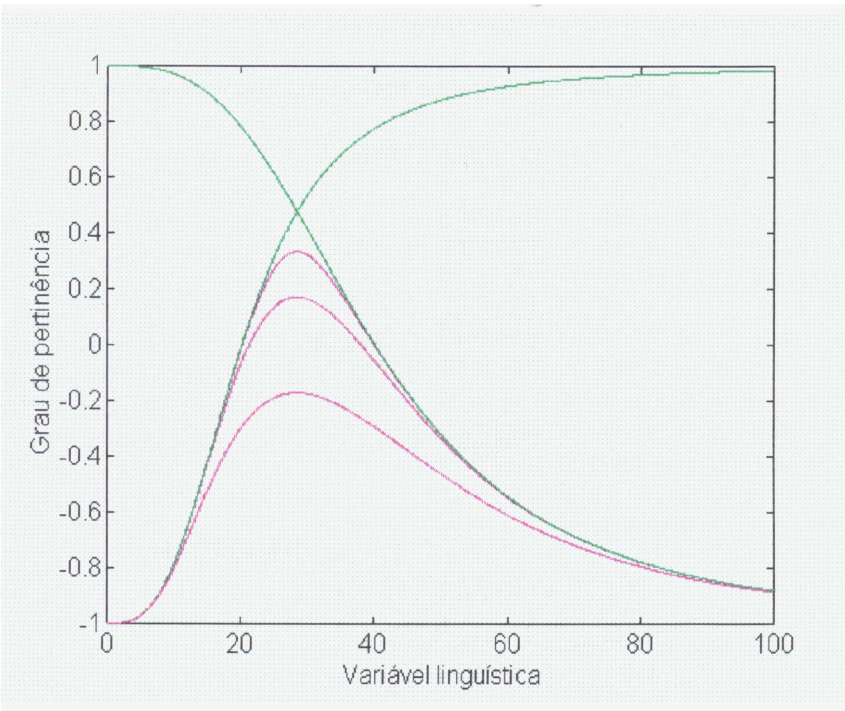

Figura 2.1: Variações de α e de β na família de t-normas.

Usando as propriedades (**T4**) e (**S4**), pode-se estender os operadores considerados, ficando bem definido

$$U_c^{\alpha,\beta}(x_1, x_2, \cdots, x_n) \tag{2.16}$$

e

$$U_d^{\alpha,\beta}(x_1, x_2, \cdots, x_n) \tag{2.17}$$

para $\forall (x_1, x_2, \cdots, x_n) \in [-1, 1]^n$.

Neste trabalho, o estudo se restringe aos operadores $U_c^{1,1}$ e $U_d^{1,1}$, denotados por

$$U_c^{1,1} = C \tag{2.18}$$

$$U_d^{1,1} = D \tag{2.19}$$

Usando a notação da introdução, pode-se escrever

$$C(x_1, \cdots, x_n) = Op_3(Op_3(x_1) + \cdots + Op_3(x_n)) \tag{2.20}$$

$$D(x_1, \cdots, x_n) = Op_1(Op_2(x_1) + \cdots + Op_2(x_n)) \tag{2.21}$$

com as precauções quando algum $x_i = -1$, no caso de C, e algum $x_i = 1$, no caso de D. Assume-se que o leitor esteja ciente dos valores de C e D nestes casos. O uso dessas t-normas é proposto para realizar as operações lógicas *e* e *ou* do modelo da rede neural difusa RNDD a ser definido na Seção 3.3.

2.4.2 Interatividade das t-normas

Conforme [KZ02], a seleção de uma determinada t-norma é definida por dois critérios: a interatividade entre os conjuntos e a complexidade computacional. Se não houver interatividade entre os conjuntos, os operadores *min* e *max* poderão servir como bons operadores lógicos. Se houver, a escolha de t-normas e t-conormas interativas deverá ser favorecida. Por exemplo, a semântica dos conjuntos difusos *caro* e *rápido*, usados para caracterizar um veículo, sugere que estes conjuntos interagem.

Na verdade, o resultado da combinação pode incorporar o fato destes dois conjuntos interagirem (espera-se que um veículo rápido seja também caro). Em oposição, a combinação dos conjuntos *novo* e *vermelho* envolve dois grânulos de informação que são altamente não-interativos (um veículo novo não é necessariamente vermelho). A falta de interação será bem representada por *min* e *max*. Nota-se que $min(x, a)$ retornará x se x for menor que a e este resultado não reflete o valor de a.

As t-normas definidas nesta seção também são interativas, pois o resultado de sua aplicação sobre graus de pertinência depende dos valores de todos os seus argumentos, como se pode conferir nas equações abaixo:

$$C(a,b) = Op_3(Op_3(a) + Op_3(b)) = \frac{1 - (\frac{1-a}{1+a} + \frac{1-b}{1+b})}{1 + (\frac{1-a}{1+a} + \frac{1-b}{1+b})}$$

$$D(a,b) = Op_1(Op_2(a) + Op_2(b)) = \frac{(\frac{1+a}{1-a} + \frac{1+b}{1-b}) - 1}{1 + (\frac{1+a}{1-a} + \frac{1+b}{1-b})}$$

Redes difusas como as de [GFP93], [GR94] e [MP94], descritas na Seção 3.2, utilizam os operadores *min* e *max*. Em [MP94] são introduzidos operadores de implicação difusa para resolver o problema da não-interatividade. O objetivo deste trabalho consiste em mostrar que as t-normas diferenciáveis e interativas definidas pelas Equações 2.20 e 2.21 conseguem classificar e produzir regras tão bem quanto a rede de [MP94] sem a necessidade da introdução de operadores de implicação difusa e com uma formulação matemática do gradiente descendente mais definida.

2.5 Regras Difusas

Uma regra difusa é formada por duas partes:

Se *antecedente* Então *conseqüente.*

O antecedente é composto por um conjunto de condições, denominadas cláusulas, que quando satisfeitas, mesmo que parcialmente, determinam o processamento do conseqüente. Este processo é conhecido como disparo de uma regra.

As regras difusas podem ser usadas para fornecer uma maneira de visualização do conhecimento adquirido por uma rede neural. Neste trabalho estas regras oferecerão uma justificativa da decisão tomada pela rede neural, como será visto no Capítulo 4.

A unidade básica de informação representando regras difusas é uma cláusula do tipo:

O (atributo) do (objeto) é (valor).

Por exemplo, A *temperatura* do *forno* é *alta* e A *pressão* do *forno* é *baixa.* Na verdade, nestes exemplos, a temperatura e a pressão do forno podem ser consideradas variáveis lingüísticas ou atributos. Logo, as unidades básicas de informação, ou as cláusulas simples, podem ser escritas na forma canônica: X é A, onde X é uma variável lingüística com variação em um universo de discurso $\mathbf{X}$ e A é um termo primário correspondente a um conjunto difuso que possui uma função de pertinência associada. Para qualquer valor real que X assuma, há um grau de pertinência ao conjunto difuso A correspondente.

Cláusulas compostas podem ser construídas com conjunções e disjunções. Uma regra difusa tem a forma geral:

$$\text{Se } (X_1 \text{ é } A_1) \textit{ conectivo } (X_2 \text{ é } A_2) \ldots \textit{ conectivo } (X_m \text{ é } A_m)$$

$$\text{Então } (Y_1 \text{ é } B_1) \textit{ conectivo } (Y_2 \text{ é } B_2) \ldots \textit{ conectivo } (Y_p \text{ é } B_p)$$

onde A_i e B_j são conjuntos difusos definidos nos universos de discurso $\mathbf{X}_i$ e $\mathbf{Y}_j$, respectivamente, com $i = 1, \ldots, m$ e $j = 1, \ldots, p$ e *conectivo* pode ser *e* ou *ou.*

Pode-se sumarizar a sintaxe das regras difusas consideradas neste projeto, usando a seguinte gramática em notação BNF[4]:

<regra Se-Então> ::= Se (<cláusula>) Então <cláusula-simples>
<cláusula> ::= <disjunção> {) *e* (<disjunção>}
<disjunção> ::= <cláusula-simples> { *ou* <cláusula-simples>}
<cláusula-simples> ::= <atributo> é <termo-primário>

No Capítulo 4, serão citados alguns algoritmos para geração de regras a partir de redes neurais, sendo que, ao algoritmo *backtracking*, caberá uma descrição completa, pois será usado neste trabalho. As Seções 5.2.2 e 5.3.2 apresentam as regras geradas para os dados *Iris* e *Vogal.* No próximo capítulo, será feita uma revisão de conceitos elementares em torno do tema de redes neurais artificiais e será descrito o modelo de RNDD proposto neste projeto.

[4]Do Inglês: *Backus-Naur Form.*

Capítulo 3

A Rede RNDD

Os primeiros estudos sobre Redes Neurais Artificiais (RNAs) surgiram em 1943 com o psiquiatra e neuroanatomista *McCulloch* e o matemático *Pitts* (McCulloch e Pitts, 1943) *apud* [HAY99]. Em 1949, houve um avanço significativo com a publicação do livro entitulado: *The organization of behavior*, do neuropsicólogo Hebb [HEB49], no qual foi proposto que, embora os mecanismos de aprendizado humano, no nível neural, não sejam conhecidos completamente, estes podem ser simples. Essa conjectura supõe que o aprendizado, que resulta das modificações corretas das sinapses, consiste em fortalecer as conexões entre elementos da rede somente quando as unidades pré-sinápticas e pós-sinápticas estivessem ativas simultaneamente.

As RNAs e suas capacidades de aprendizagem foram extensivamente pesquisadas desde a primeira metade do século passado, experimentando uma grande popularidade nos anos 60 com o desenvolvimento dos *perceptrons*. No entanto, depois da descoberta das limitações dos perceptrons descritas por Minsky e Papert (Minsky e Papert, 1969) *apud* [HAY99] em só resolver problemas linearmente separáveis, i. é., problemas cuja solução pode ser obtida dividindo-se o espaço de entrada em duas regiões por meio de um hiperplano, o interesse nas pesquisas de redes neurais sofreu um impacto severo. Esse interesse voltou a surgir quando algoritmos de aprendizado mais poderosos foram descobertos, entre eles o aprendizado por retropropagação (Rumelhart *et. al*, 1986) *apud* [HAY99]. Desde então, as RNAs formaram uma grande área de pesquisa e são utilizadas em muitas aplicações, do controle de robôs a previsões financeiras, passando pelo suporte à tomada de decisão e pela classificação de padrões.

O reconhecimento de que as áreas de conjuntos difusos e redes neurais estão interconectadas cresce a cada dia. Por exemplo, a arquitetura de uma rede neural pode ser usada para aproximar funções de pertinência a conjuntos difusos. Uma abordagem promissora consiste em combinar essas duas tecnologias em um sistema integrado tal que se possa prover as capacidades de aprendizado das RNAs aos sistemas difusos e as capacidades de representação dos conjuntos difusos às redes neurais. Sistemas que combinam redes neurais com conjuntos difusos são chamados sistemas neurodifusos ou Redes Neurais Difusas (RNDs) [GFP93].

O primeiro objetivo deste capítulo é revisar a nomenclatura e alguns conceitos acerca das RNAs, particularmente de uma rede alimentada adiante e treinada com o algoritmo de retropropagação de erros. O segundo objetivo é introduzir alguns modelos de RNDs conhecidos da literatura e por fim introduzir a Rede Neural Difusa Diferenciável (RNDD), objeto de estudo deste projeto, juntamente com a descrição do algoritmo de retropropagação de erros que será usado para treiná-la. Alguns modelos de RNDs serão apresentados na Seção 3.2. A diversidade dos esquemas de representação do conhecimento suportados pelos conjuntos difusos pode ajudar a acelerar o aprendizado e facilita a interpretação posterior da rede treinada [PG98].

3.1 Redes Neurais Artificiais

O objetivo desta seção é revisar alguns conceitos das RNAs e aspectos de seus treinamentos. A maior parte do texto foi extraída do Capítulo 2 do livro de Haykin [HAY99]. A literatura nestes tópicos é extensa e bastante explicativa.

Uma RNA é definida como um processador maciça e paralelamente distribuído, constituído de unidades de processamento simples, denominadas neurônios, que armazenam conhecimento experimental e o tornam disponível para uso. Se assemelha ao cérebro natural em dois aspectos: o conhecimento é adquirido pela rede a partir de seu ambiente, por meio de um processo de aprendizagem e as forças de conexão entre os neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

Outra característica das RNAs, diz respeito ao fato de elas serem aproximadores universais de funções: dada uma função contínua arbitrária $f : [0,1]^m \rightarrow R^{n_3}$, $f(x) = y$, existe sempre para f, uma implementação exata com uma rede neural de três camadas, sendo a camada de entrada um vetor de dimensão m, a camada intermediária composta por $n_2 = 2m + 1$ neurônios e a camada de saída com n_3 neurônios representando as n_3 componentes do vetor de saída [KOV97].

Neurônio biológico: a característica fundamental é que a informação de suas sinapses é propagada em apenas uma direção: da membrana pré-sináptica (axônio) para a pós-sináptica (dendrito). Dependendo do tipo de neurotransmissor, esta propagação poderá ser facilitada ou inibida. Assim, as sinapses são classificadas em sinapses excitatórias e sinapses inibitórias.

Neurônio artificial: a saída de um neurônio artificial clássico é basicamente a soma dos produtos dos pesos sinápticos pelos valores de entrada neste neurônio, representada por números reais. Importante ressaltar que um neurônio artificial nada mais é do que uma unidade processadora que executa cálculos a cada iteração de um algoritmo de treinamento. Para a RNDD, as entradas e as saídas da rede serão graus de pertinência a conjuntos difusos.

Ativação de um neurônio artificial: a ativação de um neurônio define a sua saída em função de seus dados de entrada. A ativação dos neurônios da RNDD será calculada pela t-norma (e) e t-conorma (ou) definidas na Seção 2.4.

Arquitetura de uma rede neural: a arquitetura de rede usada neste trabalho será a alimentada adiante[1]. Nessas redes, os sinais são propagados em apenas um sentido: começando da camada de entrada, passando por sucessivas camadas intermediárias e alcançando a camada de saída. Observa-se neste tipo de rede a ausência de ciclos na propagação dos sinais, característicos das redes com arquitetura recorrente.

Uma RNA alimentada adiante possui tipicamente três camadas de neurônios: uma para os neurônios de entrada, uma para os neurônios processadores ou intermediários e outra para os neurônios de saída. Há conexões partindo de cada neurônio na camada de entrada para cada neurônio na camada intermediária (no caso das redes completamente conectadas) e de cada neurônio na camada intermediária para cada neurônio na camada de saída.

Pesos sinápticos: no caso de redes com três camadas, há dois conjuntos de pesos: aqueles participando das ativações dos neurônios da camada intermediária e aqueles que ajudam a determinar as ativações dos neurônios da camada de saída. Não há processamento nos neurônios da camada de entrada, também chamados de nós-fonte. Os valores dos pesos utilizados na RNDD estão no intervalo $[-1, 1]$.

Treinamento de uma rede neural: para o treinamento, considera-se que o conjunto de dados será dividido em dois subconjuntos: o conjunto de treino e o conjunto de teste. Várias técnicas de divisão de conjuntos de dados foram propostas em [FUK90], e algumas delas serão utilizadas na Seção 5.2.1 para testar a RNDD.

O processo de treinamento visa encontrar os pesos das conexões sinápticas entre os neurônios, que façam com que a rede classifique de maneira correta as instâncias de dados pertencentes ao conjunto de teste. Inicia-se com uma amostra de treino submetida à rede que, por meio de um algoritmo de aprendizado, ajustará os pesos sinápticos codificando os exemplos de treino para dentro da rede. Espera-se que a rede, assim projetada, seja capaz de generalizar.

Generalização: diz-se que uma rede generaliza bem quando o mapeamento de entrada-saída computado for correto ou aproximadamente correto para dados de teste não utilizados no treino da rede. Uma rede neural que é projetada para generalizar bem produzirá um mapeamento de entrada-saída correto, mesmo quando a entrada for um pouco diferente dos exemplos usados para treiná-la.

O algoritmo de retropropagação: o algoritmo de treinamento por retropropagação é o algoritmo mais utilizado para treinar Perceptrons de Múltiplas Camadas (PMCs). Recebeu este nome pois, a cada iteração, faz uma retropropagação dos erros da última camada da rede em direção à primeira, para ajustar os pesos sinápticos.

Os erros na saída determinam a medida dos erros de saída da camada intermediária, que são utilizados como base para o ajuste dos dois conjuntos de pesos entre os pares de camadas. Este ajuste e o cálculo das saídas é um processo iterativo que é levado adiante até que os erros estejam em um nível de tolerância prescrita.

A taxa de aprendizagem e o momento: a taxa de aprendizagem e o momento são parâmetros de treino usados para dosar o ajuste dos pesos. Quanto menor for o parâmetro

[1]Do Inglês: *feedforward*.

taxa de aprendizagem, menor serão as variações dos pesos sinápticos da rede de uma iteração para outra e mais suave será a trajetória no espaço de pesos. No entanto, esta suavidade é obtida às custas de uma aprendizagem proporcionalmente mais lenta. Por outro lado, se a taxa de aprendizagem for muito grande, com o intuito de acelerar o aprendizado, as altas modificações resultantes nos pesos sinápticos podem tornar a rede instável, i. é., oscilatória. Para aumentar a taxa de aprendizagem, amenizando o perigo da instabilidade, é que se utiliza o momento[2].

Há muitas variações do algoritmo de retropropagação que tentam amenizar alguns de seus problemas-chave. Um deles é que sua convergência no treino não garante a descoberta de um mínimo global. Para evitar o problema de o algoritmo ficar parado em um mínimo local e, eventualmente, prevenir oscilações do EQM no espaço dos pesos, durante o processo de convergência para uma solução com erro mínimo, é geralmente utilizada uma heurística. Esta heurística consiste de diminuições gradativas da taxa de aprendizagem e do momento [MP94]. Esta heurística será utilizada na RNDD.

Modos de treinamento: há dois modos de treinamento: o seqüencial e o em lote. No modo seqüencial, os pesos são atualizados pelo algoritmo de treinamento a cada vez que um padrão de entrada é apresentado à rede. Há, também, o modo de atualização em lote, em que as atualizações nos pesos são realizadas após a apresentação de todos os exemplos de treinamento que constituem uma época.

Do ponto de vista operacional, o modo seqüencial é preferível em relação ao modo por lote, porque requer menos armazenamento local para cada conexão sináptica. Além disso, dado que os padrões são apresentados à rede de maneira aleatória, o uso do ajuste padrão-por-padrão torna a busca no espaço de pesos de natureza estocástica. Isto faz com que se diminua a probabilidade de que o algoritmo de retropropagação fique preso em um mínimo local.

Quando os dados de treinamento são redundantes, i. é., o conjunto de dados contém várias cópias dos mesmos padrões, constata-se ainda que, diferentemente do modo por lote, o modo seqüencial é capaz de tirar vantagem de sua redundância porque os exemplos são apresentados um de cada vez. Acrescenta-se ainda o fato de o modo seqüencial ser mais simples de se implementar. Este será o modo de treinamento usado neste projeto para a implementação da RNDD.

Tipos de aprendizado: o tipo de aprendizado diz respeito ao processo pelo qual a RNA aprende por meio dos padrões a ela apresentados. Os procedimentos de aprendizado que levam as RNAs a aprender determinadas tarefas podem ser classificados em duas classes: aprendizado supervisionado e aprendizado não-supervisionado. No aprendizado supervisionado, a rede é alimentada por dados de entrada e seus respectivos valores de saída. No aprendizado não-supervisionado, que não requer valores de saída desejada, a rede agrupa os vetores com maior semelhança (*clusters*). O tipo de aprendizado realizado pela RNDD será o supervisionado.

[2]Também conhecido como coeficiente de amortecimento.

3.2 Redes Neurais Difusas

A partir de 1980, diversos pesquisadores têm procurado reunir as potencialidades dos sistemas difusos e das redes neurais nas chamadas Redes Neurais Difusas (RNDs). Diferentemente do modelo convencional de RNA, uma RND pode ser capaz de manipular entradas disponíveis na forma lingüística. Além disso, os modelos clássicos de RNAs geralmente modelam a condição ideal, em que os valores dos atributos definem a pertinência de um objeto a uma ou a outra classe, mas não a mais de uma. Já os modelos difusos podem considerar a pertinência gradual de objetos à várias classes, o que ocorre de fato em problemas reais de classificação. O modelo proposto no presente trabalho incorpora estes conceitos e é capaz de classificar padrões difusos, i. é., padrões que podem pertencer a mais de uma classe.

Segundo [KY95], as seguintes características, ou algumas delas, distinguem as RNDs de suas equivalentes clássicas: as entradas, saídas ou pesos da rede são conjuntos difusos ou são graus de pertinência a conjuntos difusos. Além disso, as entradas ponderadas pelos pesos de cada neurônio não são agregadas por um somatório, mas por alguma outra operação de agregação.

Em [NKK97] é apresentada uma taxonomia para as diferentes combinações de redes neurais e conjuntos difusos:

1. Redes neurais difusas: métodos difusos são usados para aperfeiçoar as capacidades de aprendizado ou de desempenho de uma rede neural. Isto pode ser feito usando regras difusas para mudar a taxa de aprendizado ou criando-se uma rede cujas entradas são conjuntos difusos;

2. Sistemas concorrentemente neural/difuso: uma rede neural e um sistema difuso trabalham juntos na mesma tarefa, mas sem a influência de um sobre o outro, i. é., nenhum dos dois sistemas é usado para determinar os parâmetros do outro;

3. Modelos neurodifusos cooperativos: uma rede neural é usada para determinar os parâmetros (regras, pesos das regras ou conjuntos difusos) de um sistema difuso;

4. Modelos neurodifusos híbridos: os enfoques neurodifusos modernos são desta forma. Uma rede neural e um sistema difuso são combinados em uma arquitetura homogênea. O sistema pode ser interpretado tanto como uma rede neural com parâmetros difusos quanto um sistema difuso implementado de uma forma paralela e distribuída. E. g., o sistema *Adaptive Network-based Fuzzy Inference System* (ANFIS) [JAN93].

Segundo esta taxonomia a rede que é proposta neste projeto é do tipo 4. Contudo, a nomenclatura que será usada é a de rede neural difusa diferenciável.

Uma RND típica se propõe a desempenhar duas tarefas principais: a classificação de padrões em classes e a geração de regras. Desta forma, os pesos da conexões são usados para gerar regras de decisão para a classificação, como será visto na Seção 4.1.3. Em outras palavras, a rede deve ser capaz de codificar automaticamente regras do tipo Se - Então. A incorporação de conjuntos difusos no modelo permite também a manipulação de incertezas nos vários estágios do processamento. RNDs com geração de regras têm sido desenvolvidas e utilizadas para a mineração de dados em muitas áreas de aplicação [MP02], tais como: diagnóstico médico, tomada de decisão, classificação e predição.

A topologia da RND que é objeto de estudo deste trabalho é mostrada na Figura 3.1 e será descrita na Seção 3.3. As notações O_{nl}, O_{ni} e O_{nj} da figura denotam, respectivamente, a saída do l-ésimo neurônio da camada de entrada difusa, a saída do i-ésimo neurônio da camada *e* e a saída do j-ésimo neurônio da camada *ou*. As notações w_{il} e w_{ji} denotam, respectivamente, o peso da conexão entre o l-ésimo neurônio da camada de entrada e o i-ésimo neurônio da camada *e* e o peso da conexão entre o i-ésimo neurônio da camada *e* e o j-ésimo neurônio da camada *ou*.

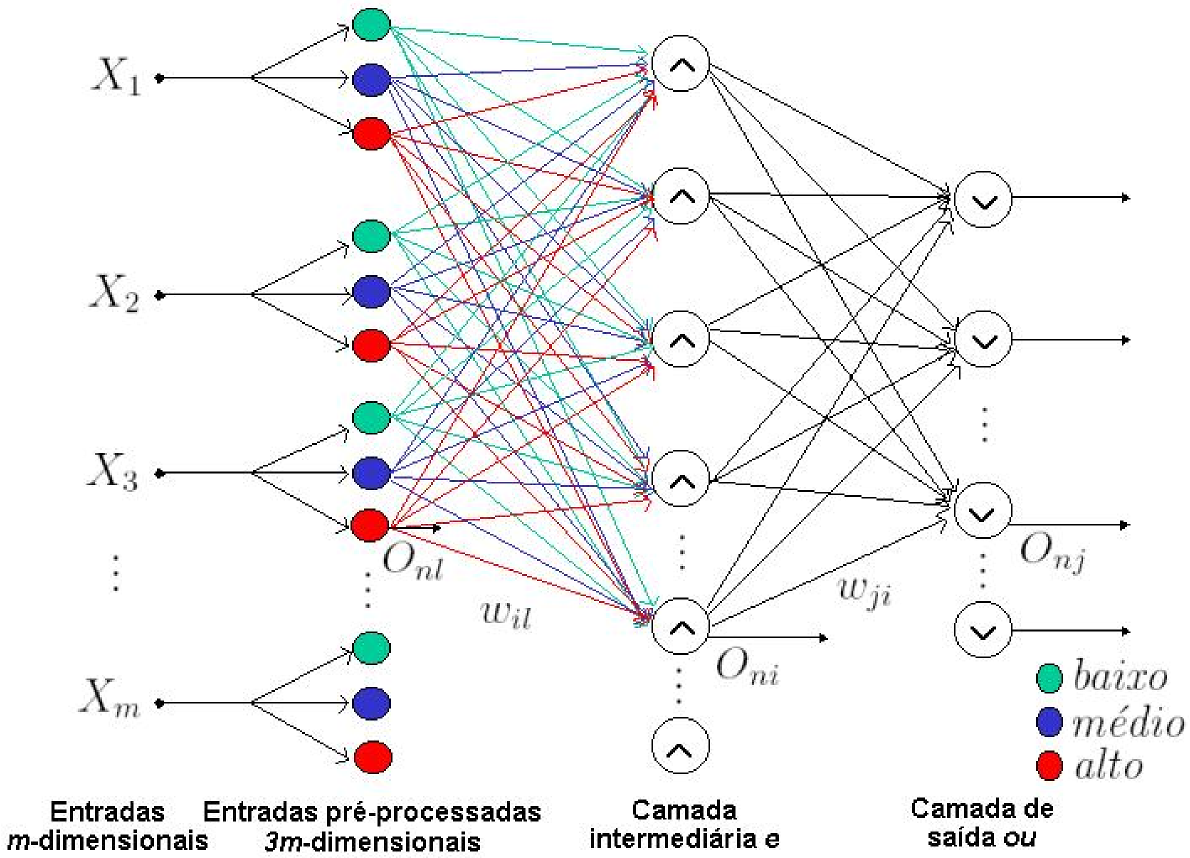

Figura 3.1: Rede neural difusa diferenciável.

Semelhantes a este modelo, na literatura de RNDs, encontram-se os modelos de rede funcional de Gomide e Pedrycz [GFP93], de Gupta [GR94] e o modelo de Mitra e Pal [MP94]. A seguir tem-se uma breve descrição destes modelos:

No modelo funcional [GFP93], um conjunto de regras é codificado na topologia da rede. Também é possível gerar regras a partir dela, se um conjunto de dados de entrada com suas saídas correspondentes estiver disponível para treinamento da rede. Devido a sua estrutura, o conhecimento inicial pode ser embutido na rede, acelerando o aprendizado e o ajuste das regras. Sua arquitetura consiste em agrupamentos de neurônios lógicos *e* e *ou* não-completamente conectados. Este modelo de rede recebe, como entrada, graus de pertinência a conjuntos difusos e produz, como saídas, valores reais. As t-normas e t-conormas utilizadas são o *min* e o *max*, respectivamente.

A topologia do segundo modelo, o de Gupta [GR94], só contém neurônios *ou* completamente conectados. Este modelo de rede é destinado à classificação de padrões e geração de regras e recebe como entrada e produz como saídas, tanto graus de pertinência, quanto valores reais. Os operadores *min* e *max* são utilizados como t-norma e t-conorma, respectivamente.

A topologia da rede de Mitra e Pal [MP94] é a mesma da Figura 3.1. Os símbolos $\wedge$ e $\vee$ representam a t-norma e a t-conorma, respectivamente. Tal como o modelo proposto nesse trabalho, destina-se a classificação de padrões e a geração de regras, trabalha com incerteza tanto nas entradas quanto nas saídas. A topologia desta rede é completamente conectada e recebe entradas na forma de valores lingüísticos, valores reais ou conjuntos difusos. Essas entradas são representadas como graus de pertinência aos conjuntos difusos *baixo*, *médio* e *alto* modelados pela função apresentada na Equação 2.2. As saídas desta rede são graus de pertinência. Os testes foram feitos com o conjunto de dados *Vogal*, descrito na Seção 5.3. Na Figura 3.2, é dada uma visão geral dos vários estágios envolvidos no processo de tomada de decisão e de geração de regras da rede de Mitra e Pal. Basicamente, estes pesquisadores utilizaram este modelo de RND com t-normas e sem t-normas. No modelo sem t-normas, denominado de *Fuzzy O*, somas ponderadas e função de ativação sigmoidal foram utilizadas. O modelo usando as t-normas produto e soma probabilística foi chamado de *P*.

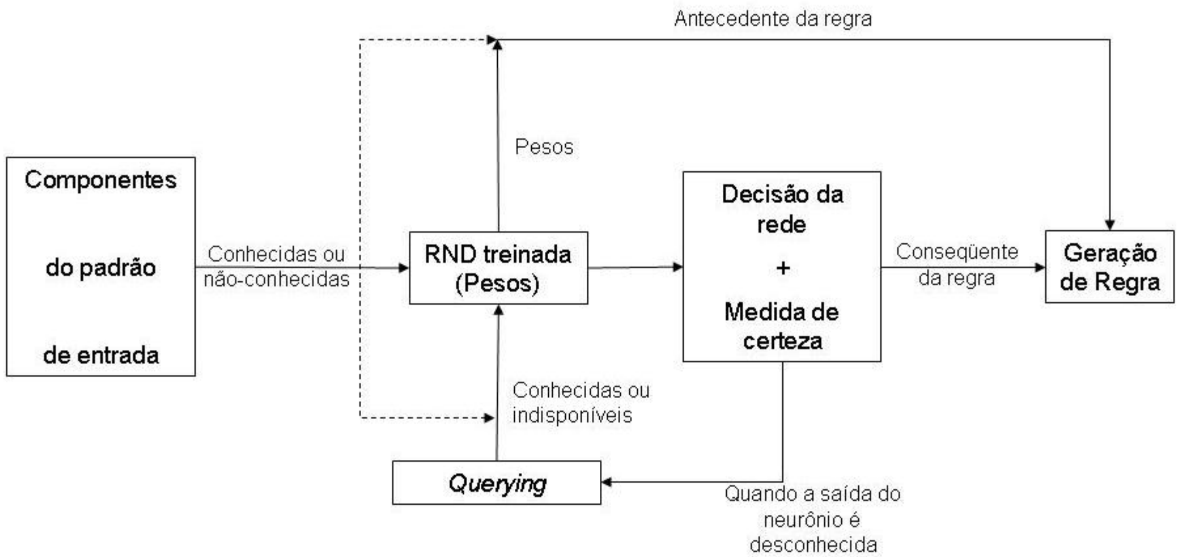

Figura 3.2: Diagrama de blocos da rede de Mitra e Pal, extraído de [MP94].

O modelo de rede RNDD, que será descrito na próxima seção, utiliza a mesma topologia de Mitra e Pal, mas com as t-normas diferenciáveis descritas na Seção 2.4 e sem o módulo de *querying* mostrado na Figura 3.2. Ambos os modelos serão comparados quanto à classificação e à geração de regras. Os resultados desta comparação serão apresentados na Seção 5.3.

3.3 Rede Neural Difusa Diferenciável (RNDD)

A rede neural da Figura 3.1 será implementada neste trabalho e os resultados com os experimentos serão descritos no Capítulo 5. Nas seções a seguir, são dadas as descrições dos processamentos realizados em cada camada da rede e do algoritmo de retropropagação usado para ajuste dos pesos das conexões sinápticas entre os neurônios da rede.

3.3.1 Pré-processamento difuso das entradas

O n-ésimo padrão de entrada m-dimensional $\overrightarrow{X}_n = [X_{n1}, X_{n2}, \ldots, X_{nm}]$ será substituído por um vetor $3m$-dimensional:

$$\overrightarrow{O}_n = [baixo(X_{n1}), médio(X_{n1}), alto(X_{n1}), \ldots, baixo(X_{nm}), médio(X_{nm}), alto(X_{nm})] \tag{3.1}$$

onde $baixo(X_{n1})$ indica o grau de pertinência do valor X_{n1} do primeiro atributo do n-ésimo padrão ao conjunto difuso *baixo*.

A função de pertinência originalmente usada em [MP94] e descrita na Seção 2.2 possuía imagem em $[0, 1]$. Para este projeto, esta função foi transformada para assumir valores no intervalo $[-1, 1]$, fazendo $2A(x) - 1$, contudo, mantendo a mesma forma:

$$A(x) = \begin{cases} 2(2(1 - \frac{|x-c|}{\lambda})^2) - 1 & \text{se } \frac{\lambda}{2} \leq |x - c| \leq \lambda \\ 2(1 - 2(\frac{|x-c|}{\lambda})^2) - 1 & \text{se } 0 \leq |x - c| \leq \frac{\lambda}{2} \\ -1 & \text{caso contrário} \end{cases} \tag{3.2}$$

onde $\lambda > 0$ é o raio da função A e c é o centro.

Os parâmetros c e λ são calculados em função dos valores máximo e mínimo do atributo X no conjunto de treinamento. Para cada um dos três conjuntos difusos: *baixo*, *médio* e *alto*, definem-se:

$$\begin{aligned}
\lambda_{médio(X)} &= 0.5(x_{max} - X_{min}) \\
c_{médio} &= X_{min} + \lambda_{médio(X)} \\
\lambda_{baixo(X)} &= \frac{1}{Sobrep}(c_{médio(X)} - X_{min}) \\
c_{baixo(X)} &= c_{médio(X)} - 0.5\lambda_{baixo(X)} \\
\lambda_{alto(X)} &= \frac{1}{Sobrep}(X_{max} - c_{médio(X)}) \\
c_{alto(X)} &= c_{médio(X)} + 0.5\lambda_{alto(X)}
\end{aligned}$$

onde $0 < Sobrep \leq 1$ é um parâmetro que controla a extensão da sobreposição das três funções.

Em resumo, o padrão de entrada m-dimensional com componentes reais é difusificado e passa a ser um vetor $3m$-dimensional, pois cada componente real terá ter graus de pertinência aos conjuntos difusos *baixo*, *médio* e *alto*. Este vetor difusificado será a entrada para a RNDD.

Na Figura 3.3, é mostrada a forma da função de pertinência.

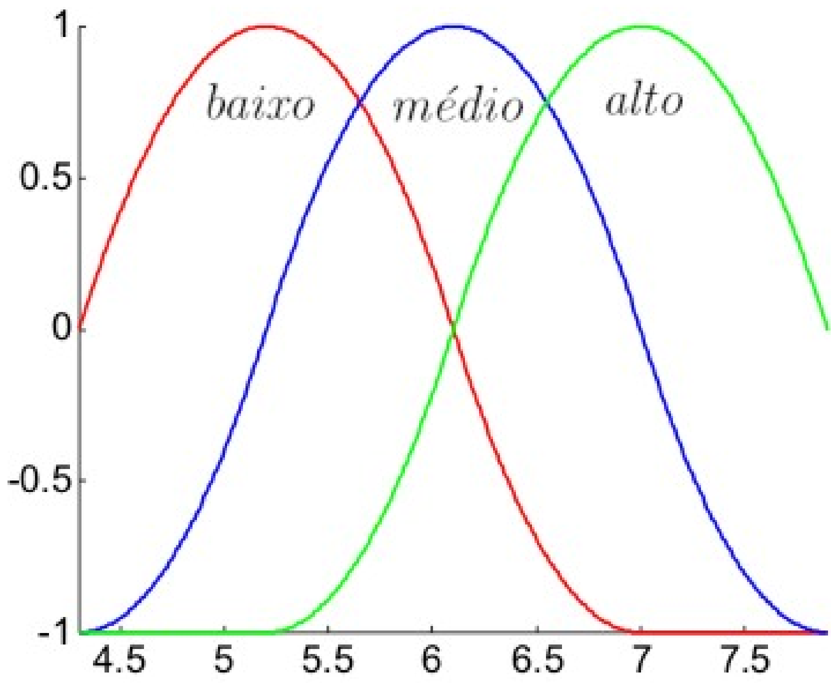

Figura 3.3: Funções de pertinência.

3.3.2 Processamento na camada intermediária *e*

Como visto na seção anterior, as entradas da rede são graus de pertinência aos três conjuntos difusos *baixo*, *médio* e *alto* correspondentes a cada um dos m atributos observados no objeto a ser classificado. Elas são denotadas por O_{nl}, em que n e l se referem ao n-ésimo padrão de treino e à l-ésima unidade de entrada, respectivamente. Considere $n = 1, \ldots, N$, em que N é o número de padrões de treino e $l = 1, \ldots, n_1$, sendo $n_1 = 3m$, o número de unidades na camada de entrada difusa. A saída das unidades de entrada são os próprios O_{nl}, observando-se que o peso da conexão de uma unidade l desta camada com uma unidade i da camada intermediária é denotada por w_{il}.

A Figura 3.4 traz um neurônio *e*.

A camada intermediária é formada por n_2 neurônios *e*. A saída das unidades desta

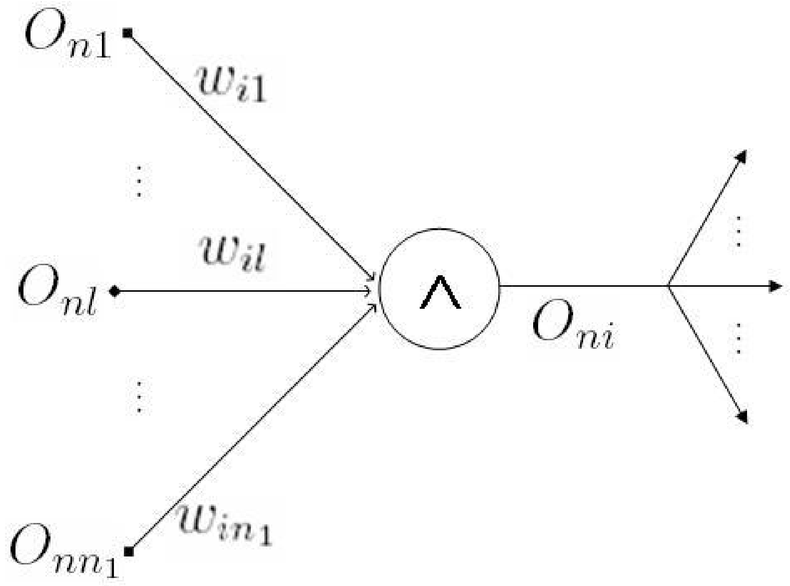

Figura 3.4: Neurônio *e*.

camada é denotada por O_{ni}, com $i = 1, \ldots, n_2$. Para facilitar a comparação deste trabalho com a dedução da regra delta generalizada dada em [RHW86], pode-se considerar a função de transferência como sendo igual à identidade I. Assim:

$$O_{ni} = I(net_{ni}) \tag{3.3}$$

$$net_{ni} = \bigwedge_{l=1}^{n_1} (O_{nl} \vee w_{il}) \tag{3.4}$$

A t-norma e a t-conorma que foram propostas na Seção 2.4 são utilizadas aqui. Empregam-se na Equação 3.4, $\bigwedge = U_c$, de onde se obtém

$$O_{ni} = U_{c\,l=1}^{\;n_1} U_d(O_{nl}, w_{il}) \tag{3.5}$$

com $i = 1, \ldots, n_2$.

Extendendo a Equação 3.5 para o índice l temos:

$$O_{ni} = U_c(U_d(O_{n1}, w_{i1}), \ldots, U_d(O_{nl}, w_{il}), \ldots, U_d(O_{nn_1}, w_{in_1})), \tag{3.6}$$

3.3.3 Processamento na camada de saída *ou*

A camada de saída é formada por n_3 neurônios *ou*, sendo este número, via de regra, igual ao número de classes em que os exemplos de treino são classificados. A saída destas unidades é denotada por O_{nj}, com $j = 1, \ldots, n_3$ e representa valores de pertinência dos padrões às classes. Para facilitar a comparação com [RHW86] será escrito:

$$O_{nj} = I(net_{nj}) \tag{3.7}$$

$$net_{nj} = \bigvee_{i=1}^{n_2} (O_{ni} \wedge w_{ji}) \tag{3.8}$$

A Figura 3.5 ilustra um neurônio *ou*. A t-norma e a t-conorma que foram propostas

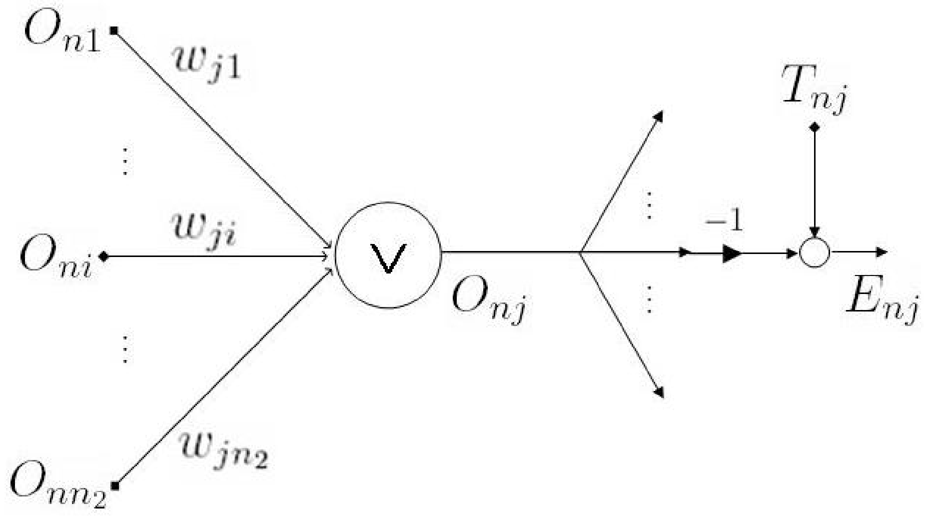

Figura 3.5: Neurônio *ou*.

na Seção 2.4 são também utilizadas aqui. Empregam-se na Equação 3.8, $\bigwedge = U_c$, de onde se obtém

$$O_{nj} = U_{d_{i=1}^{n_2}} U_c(O_{ni}, w_{ji}) \tag{3.9}$$

com $j = 1, \ldots, n_3$.

Extendendo a Equação 3.9 para o índice i temos:

$$O_{nj} = U_d(U_c(O_{n1}, w_{j1}), \ldots, U_c(O_{ni}, w_{ji}), \ldots, U_c(O_{nn_2}, w_{jn_2})) \tag{3.10}$$

3.3.4 Exemplo de cálculo da RNDD

Com o intuito de ilustrar e prover uma maior compreensão do funcionamento interno da RNDD será mostrado um exemplo do cálculo das saídas nas camadas *e* e *ou*, quando da apresentação de um exemplo de dados à RNDD treinada. Considerando um problema de classificação com $m = 2$ atributos de entrada e 2 classes, tem-se uma rede com $n_1 = 3m = 6$ neurônios na camada de entrada, $n_2 = 2$ neurônios na camada *e* e $n_3 = 2$ neurônios na camada *ou*.

Supondo o n-ésimo padrão $\overrightarrow{X}_n = \begin{bmatrix} 5.1 & 3.5 \end{bmatrix}$ e o parâmetro $Sobrep = 1.0$, será produzido o vetor de entrada difusa $\overrightarrow{O}_n = \begin{bmatrix} 0.98 & -0.20 & -0.99 & -0.75 & 0.75 & 0.75 \end{bmatrix}$.

Será apresentado abaixo, o cálculo da saída da camada *e* e, logo em seguida, o cálculo da saída da camada *ou*. Por exemplo, para o primeiro neurônio *e*, são feitas 6 operações de disjunção (*ou*) entre cada entrada difusa e o respectivo peso da rede. A saída deste neurônio, denotada por O_{n1}, será uma operação de conjunção (e) dessas 6 disjunções. Cálculo semelhante é feito para os neurônios e, com as operações de conjunção precedendo as operações de disjunção.

Na camada intermediária e, tem-se o seguinte cálculo:

$$\begin{array}{lccccccc}
O_{n1}= & (0.98\ ou\ \text{-}0.99) & e & (\text{-}0.20\ ou\ 0.99) & e & (\text{-}0.99\ ou\ 0.99) & e & \\
 & (\text{-}0.75\ ou\ 0.99) & e & (0.75\ ou\ 0.99) & e & (0.75\ ou\ \text{-}0.66) & = & 0.74 \\
O_{n2}= & (0.98\ ou\ 0.99) & e & (\text{-}0.20\ ou\ \text{-}0.94) & e & (\text{-}0.99\ ou\ 0.99) & e & \\
 & (\text{-}0.75\ ou\ \text{-}0.99) & e & (0.75\ ou\ 0.99) & e & (0.75\ ou\ 0.99) & = & \text{-}0.78
\end{array}$$

Na camada de saída *ou*, tem-se o seguinte cálculo:

$$\begin{array}{lccccc}
O_{n1}= & (0.74\ e\ 0.99) & ou & (\text{-}0.78\ e\ \text{-}0.99) & = & 0.74 \\
O_{n2}= & (0.74\ e\ \text{-}0.99) & ou & (\text{-}0.78\ e\ 0.99) & = & \text{-}0.78
\end{array}$$

A saída da rede será: $[0.74, -0.78]$. A maior saída é a do primeiro neurônio e, pelo critério de interpretação das saídas descrito na Seção 1.3, denota a pertinência à primeira classe considerada.

3.3.5 Cálculo do EQM

O objetivo do algoritmo de treinamento é minimizar a distância entre o valor de pertinência do n-ésimo padrão à classe correta j, calculado pela rede e denotado por O_{nj}, e o desejado T_{nj}. Na RNDD, a saída desejada $\overrightarrow{T}_n$ é um vetor n_3-dimensional cujo j-ésimo elemento possui o valor 1 e os demais, o valor -1.

Essa distância, também conhecida como erro instantâneo, é dada para o n-ésimo exemplo de treino pela equação

$$E_n = \frac{1}{2}\sum_{j=1}^{n_3}(T_{nj} - O_{nj})^2 \tag{3.11}$$

A função de EQM da rede é

$$EQM = \frac{1}{N}\sum_{n=1}^{N} E_n \tag{3.12}$$

que depende de todos os w_{ji}'s e w_{il}'s.

Para um dado conjunto de treinamento, o EQM representa a função custo que mede o desempenho do aprendizado. O objetivo do processo de aprendizagem é ajustar os pesos da rede para minimizar o EQM. Para fazer esta minimização, considerar-se-á o método de treino em que os pesos são atualizados no modo seqüencial até que uma época, i. é., uma apresentação completa do conjunto de treinamento, tenha sido executada e depois o processo se repete para as outras épocas. O ajuste dos pesos é feito de acordo com os respectivos erros computados para cada padrão apresentado à rede.

3.3.6 Algoritmo de retropropagação

Baseadas no cálculo do gradiente descendente, as equações de ajuste dos pesos a cada iterada do algoritmo serão:

$$w_{il}(n+1) = w_{il}(n) + \Delta_{il}(n) \tag{3.13}$$

e

$$w_{ji}(n+1) = w_{ji}(n) + \Delta_{ji}(n) \tag{3.14}$$

onde

$$\Delta_{il}(n) = \eta(\frac{-\partial E_n}{\partial w_{il}}) + \alpha(w_{il}(n) - w_{il}(n-1)) \tag{3.15}$$

e

$$\Delta_{ji}(n) = \eta(\frac{-\partial E_n}{\partial w_{ji}}) + \alpha(w_{ji}(n) - w_{ji}(n-1)) \tag{3.16}$$

sendo η a taxa de aprendizagem e α, o momento.

Por meio de derivações sucessivas, chega-se as seguintes expressões para as derivadas:

$$\begin{aligned} \frac{\partial E_n}{\partial w_{ji}} &= (T_{nj} - O_{nj}) \frac{[4a_i^2 - 8a_i + 4]}{[(3-a_i) - U_c(O_{ni}, w_{ji})(1+a_i)]^2} \\ & \frac{(4O_{ni}^2 + 8O_{ni} + 4)}{[(3+O_{ni}) + w_{ji}(1-O_{ni})]^2} \end{aligned} \tag{3.17}$$

$$\begin{aligned} \frac{\partial E_n}{\partial w_{il}} &= \frac{[4O_{nl}^2 - 8O_{nl} + 4]}{[(3-O_{nl}) - w_{il}(1+O_{nl})]^2} \frac{[4a_l^2 + 8a_l + 4]}{[(3+a_l) + U_d(O_{nl}, w_{il})(1-a_l)]^2} \\ & \sum_{j=1}^{n_3} (T_{nj} - O_{nj}) \frac{[4a_i^2 - 8a_i + 4]}{[(3-a_i) - U_c(O_{ni}, w_{ji})(1+a_i)]^2} \frac{[4w_{ji}^2 + 8w_{ji} + 4]}{[(3+w_{ji}) + O_{ni}(1-w_{ji})]^2} \end{aligned} \tag{3.18}$$

sendo

$$a_i = U_{d_{i' \neq i}}^{n_2} U_c(O_{ni'}, w_{ji'}) \tag{3.19}$$

$$a_l = U_{c_{l' \neq l}}^{n_1} U_d(O_{nl'}, w_{il'}) \tag{3.20}$$

3.3.7 Truncagem dos pesos

Os pesos da RNDD são inicializados com valores aleatórios entre $]-1,1[$, pois as t-normas e t-conormas apresentadas na Seção 2.4 do Capítulo 2 são definidas no intervalo $[-1,1]$.

Com o ajuste dos pesos a cada ciclo, ou seja, no modo padrão por padrão, os pesos podem sair do intervalo $[-1,1]$. Para que eles permanecessem neste intervalo foi realizada a truncagem, descrita a seguir, para as duas camadas de pesos.

Considerando os pesos da n-ésima iteração na primeira camada da rede, denotados por $w_{il}(n)$ para $l = 1,2,...,n_1$ e $i = 1,2,...,n_2$ e da segunda camada, denotados por $w_{ji}(n)$ para $i = 1,2,...,n_1$ e $j = 1,2,...,n_3$. São $n_1 * n_2 + n_2 * n_3$ pesos no total.

Fez-se o ajuste para a próxima iteração, utilizando as Equações 3.13 e 3.14. Depois deste ajuste com direção contrária a do gradiente, para os pesos que não ficaram em $[-1,1]$, foi feita uma truncagem dos $\Delta(n)$'s correspondentes para que estes pesos ficassem em $[-1,1]$. Isto foi feito desde o primeiro ajuste dos pesos iniciais que estavam em $]-1,1[$. Observa-se a seguir o procedimento de truncagem.

Considerando que os pesos na n-ésima iteração já estavam em $[-1,1]$, foi calculado

$$\Theta_1(n) = min\{w_{il}(n) + 1, 1 - w_{il}(n)\} \tag{3.21}$$

para todo l, $1 \leq l \leq n_1$ e todo i, $1 \leq i \leq n_2$.

$$\Theta_2(n) = min\{w_{ji}(n) + 1, 1 - w_{ji}(n)\} \tag{3.22}$$

para todo i , $1 \leq i \leq n_2$ e todo j, $1 \leq j \leq n_3$. E depois,

$$\Theta min(n) = min\{\Theta_1(n), \Theta_2(n)\}. \tag{3.23}$$

$Tetamin(n)$ é calculado levando em conta todos os pesos da n-ésima iteração. Pode-se provar que $\Theta min(n) \in]-1,1[$.

Vale lembrar que a função EQM a ser minimizada, dada pela Equação 3.12 é definida no espaço de todos os $w(n)$s, e que o vetor de todos os $\Delta(n)$'s dá a direção do ajuste.

Foi cancelado o ajuste feito pelas Equações 3.13 e 3.14 das coordenadas do vetor de pesos que não ficaram em $[-1,1]$ e feito outro ajuste com uma truncagem dos $\Delta(n)$'s correspondentes, i. é., para os índices l, i, j correspondentes:

$$w_{il}(n+1) = w_{il}(n) + \Theta min \frac{\Delta_{il}(n)}{|| \Delta(n) ||} \tag{3.24}$$

e

$$w_{ji}(n+1) = w_{ji}(n) + \Theta min \frac{\Delta_{ji}(n)}{|| \Delta(n) ||} \tag{3.25}$$

onde $\| \Delta(n) \|$ é a norma do vetor cujas componentes são todos os $\Delta(n)$'s, ou seja, é o vetor de todos os $\Delta_{il}(n)$ com $l = 1, 2, ..., n_1$ e $i = 1, 2, ..., n_2$ e de todos os $\Delta_{ji}(n)$ com $l = 1, 2, ..., n_2$ e $j = 1, 2, ..., n_3$.

O código em Português estruturado deste procedimento foi colocado no Apêndice.

3.3.8 T-norma em $[-1, 1]$

Na literatura, observa-se que, na maioria dos casos, as operações lógicas são realizadas sobre o intervalo unitário positivo $[0, 1]$, chamado de intervalo unipolar. Para a implementação da RNDD, foram redefinidas as operações lógicas para o intervalo $[-1, 1]$, chamado de intervalo bipolar. Esta redefinição foi necessária, haja vista que as t-normas utilizadas no projeto estarem definidas sobre este intervalo.

As saídas do pré-processamento, dos neurônios da camada *e*, da camada de saída *ou* e os pesos da RNDD estão definidas sobre o intervalo bipolar. Para o cálculo da medida de certeza $cert_j^H$, que será apresentada na Seção 4.1.3, as saídas da rede foram convertidas para o intervalo unitário para evitar que esta medida assumisse valores negativos. Para o cálculo dos modificadores, que também serão apresentados na Seção 4.1.3, também houve esta transformação, já que para calcular alguns deles é necessário extrair-se a raiz quadrada dos graus de pertinência.

Apontado constantemente pela literatura, o principal ponto fraco das redes neurais está na dificuldade em explicar suas decisões, já que o conhecimento apreendido quase sempre se encontra armazenado na forma de uma grande quantidade de pesos de conexões. O próximo capítulo se ocupará de uma breve introdução às técnicas e algoritmos de extração conhecimento, na forma de regras, a partir de redes neurais treinadas. Será dada uma ênfase maior no algoritmo de extração que será utilizado sobre a RNDD, o algoritmo *backtracking*.

Capítulo 4

Geração e Avaliação de Regras da RNDD

A aplicação de RNAs em diversos domínios tem sido intensificada nos últimos anos, como indicam os bons resultados e constantes pesquisas na área. Entretanto, todo o poder oferecido pelas RNAs esbarra na sua incapacidade de explicar, de uma forma diretamente compreensível, suas decisões. Este problema é fator de motivação para as várias pesquisas relacionadas ao desenvolvimento de técnicas para a geração de regras[1] a partir de RNAs treinadas. Essas técnicas têm a finalidade de conferir às RNAs uma certa capacidade de explanação[2].

Alguns domínios de aplicação, e. g., diagnóstico médico, demandam uma justificativa de como a rede produziu uma determinada conclusão. A explanação torna a decisão tomada por uma rede neural mais aceitável para os usuários. A falta da capacidade de explanação é uma das razões pelas quais muitos usuários não se interessam por esta técnica. Esta explanação, na forma de regras, serve para justificar uma saída da rede em resposta a um dado padrão de entrada ou mesmo subsidiar algum conhecimento sobre as propriedades do conjunto de treinamento usado. Existem outras razões que tornam a geração de regras a partir de RNAs uma tarefa importante, entre as quais a exploração dos dados e o melhoramento da generalização.

Para a RNDD será utilizado o algoritmo de geração de regras *backtracking*, mostrado na Seção 4.1.3, por sua natureza intuitiva e para permitir uma futura comparação com os resultados de geração de regras obtidos em [MP94].

4.1 Geração de Regras

O objetivo desta seção é apresentar três algoritmos de geração de regras: o TREPAN, o EN e o *backtracking*.

[1]Geração de regras é um caso especial de extração do conhecimento.

[2]Neste texto, os termos explanação, justificativa e explicação serão usados indistintamente com o mesmo significado.

4.1.1 O algoritmo TREPAN

O algoritmo *TREes PArroting Networks* (TREPAN) [CS96] tem como finalidade representar o conhecimento armazenado em uma RNA na forma de uma Árvore de Decisão (AD). Este algoritmo gera uma AD tomando como entrada uma RNA treinada e os dados utilizados para o seu treinamento e é aplicável à maioria dos modelos de RNAs, já que não leva em consideração a arquitetura da rede. O princípio empregado é o mesmo utilizado pelos algoritmos convencionais de indução de ADs, como CART e C4.5, que as constroem por particionamentos recursivos do conjunto de exemplos [REZ03].

A extração de conhecimento na forma de um conjunto de regras é tratada como um problema de aprendizado indutivo. Neste aprendizado, o objetivo é a função representada pela rede e a descrição do conhecimento adquirido é uma AD que a aproxima. Para evitar que as ADs extraídas fiquem muito grandes, é utilizado um parâmetro para limitar o número de seus nós internos. A construção da árvore é guiada por um oráculo, constituído pela própria RNA treinada, capaz de responder perguntas à rede, uma vez que é nela que está presente o conhecimento a ser extraído. A vantagem de aprender por meio de perguntas ao invés de por exemplos é que as perguntas podem conseguir as informações necessárias de uma forma mais precisa. Uma descrição do algoritmo pode ser encontrada em [CS96].

4.1.2 O método EN

O método *Explanation Facility*[3] (EN) [PT92] relaciona neurônios de entrada com neurônios de saída ou neurônios de saída com neurônios de entrada, sendo também capaz de mostrar a seqüência de neurônios utilizados nestes relacionamentos. Essas características auxiliam na descoberta dos neurônios de entrada ou de saída mais importantes bem como das conexões mais usadas, ajudando assim, na compreensão dos conhecimentos armazenados na RNA.

Esta técnica busca a identificação dos caminhos de propagação mais significativos da rede, uma vez que, na maioria das vezes, as redes são formadas por um grande número de conexões distribuídas em várias camadas. Em outras palavras, o método EN seleciona uma fração pré-determinada de neurônios em cada camada baseando-se nos valores absolutos de suas conexões com os demais neurônios da rede. Esses neurônios selecionados são considerados os mais significativos da rede, sendo assim, utilizados para a geração das regras.

O EN fornece dois mecanismos de explicação: o mecanismo Porque?, que relaciona as saídas de uma RNA com suas entradas e o mecanismo Como?, que relaciona as entradas de uma RNA com suas saídas. Há, ainda, a opção Trace que gera um subconjunto dos neurônios envolvidos em cada camada nas cadeias de resolução dos dois mecanismos Porque? e Como?.

[3]Foram encontrados na literatura os termos equivalentes em Português: Facilidade de Explanação e Facilidade de Explicação.

A explicação dada pelo EN segue três conceitos que devem ser analisados em conjunto para dar um significado adequado aos grupos de neurônios de entrada selecionados. Estes conceitos são:

- Pesos maiores contribuem mais, assumindo que as entradas estão normalizadas;
- Nós com entrada significativa podem ser agrupados, e estes grupos constroem e representam conceitos;
- Relacionando informações específicas da aplicação para cada neurônio, pode-se associar um significado concreto aos neurônios de entrada.

4.1.3 O algoritmo *backtracking* para RNDD

O Algoritmo *Backtracking* (AB), como descrito nesta seção, dará uma justificativa ao usuário, escrita na forma de regras Se - Então, para a explanação de como a rede classificou um determinado exemplo do conjunto de treino na j-ésima classe. Uma possível regra com a sintaxe apresentada na Seção 2.5 para as flores íris seria:

Se (*Comprimento da Pétala* é *médio ou Comprimento da Pétala* é *alto*)
e (*Largura da Pétala* é *baixo*) Então *classe* é *Íris Setosa*.

Para cada entrada do conjunto de treino será gerada uma regra pela aplicação do AB. A capacidade de representação das regras assim obtidas pode ser avaliada usando medidas que serão descritas na Seção 4.2.1.

O AB implementa uma heurística[4] garantindo que sejam selecionados os caminhos até o j-ésimo neurônio na camada de saída (que reconhece a classe j) ao longo dos quais os pares consecutivos de neurônios possuam pesos com correlação positiva significativa [MP94]. Devido à sua natureza intuitiva e por ser aplicável à RNDD este será o algoritmo utilizado para a implementação do módulo de geração de regras comentado na Seção 5.2.2.

A topologia genérica da rede considerada é mostrada na Figura 4.1. Note que esta figura é uma generalização da Figura 3.1 para considerar redes com mais de três camadas. A notação também é mais genérica. Observa-se que um neurônio j na camada H é conectado ao neurônio i na camada $H-1$ por meio do peso sináptico denotado por w_{ji}^{H-1}. Denotam-se por y_i^{H-1} e y_j^H as saídas dos neurônios i e j, respectivamente. Visando manter a generalidade desta notação para redes com mais de três camadas, são feitas as equivalências com a notação utilizada no capítulo anterior: $O_{nl} = y_l^0$, $O_{ni} = y_i^1$, $O_{nj} = y_j^2$, $w_{il} = w_{ji}^0$, $w_{ji} = w_{ji}^1$.

Supõe-se, inicialmente, que um usuário deseje saber como a rede classificou um determinado padrão de entrada como sendo pertencente à classe j. O processo inicia-se pela camada de saída $h = H$ e continua até que a camada de entrada difusa, onde $h = 0$, seja alcançada.

[4]Heurística é uma metodologia usada para resolver problemas por métodos que, embora não rigorosos, geralmente refletem o conhecimento humano e permitem obter uma solução satisfatória (Novo Dicionário Aurélio - Século XXI).

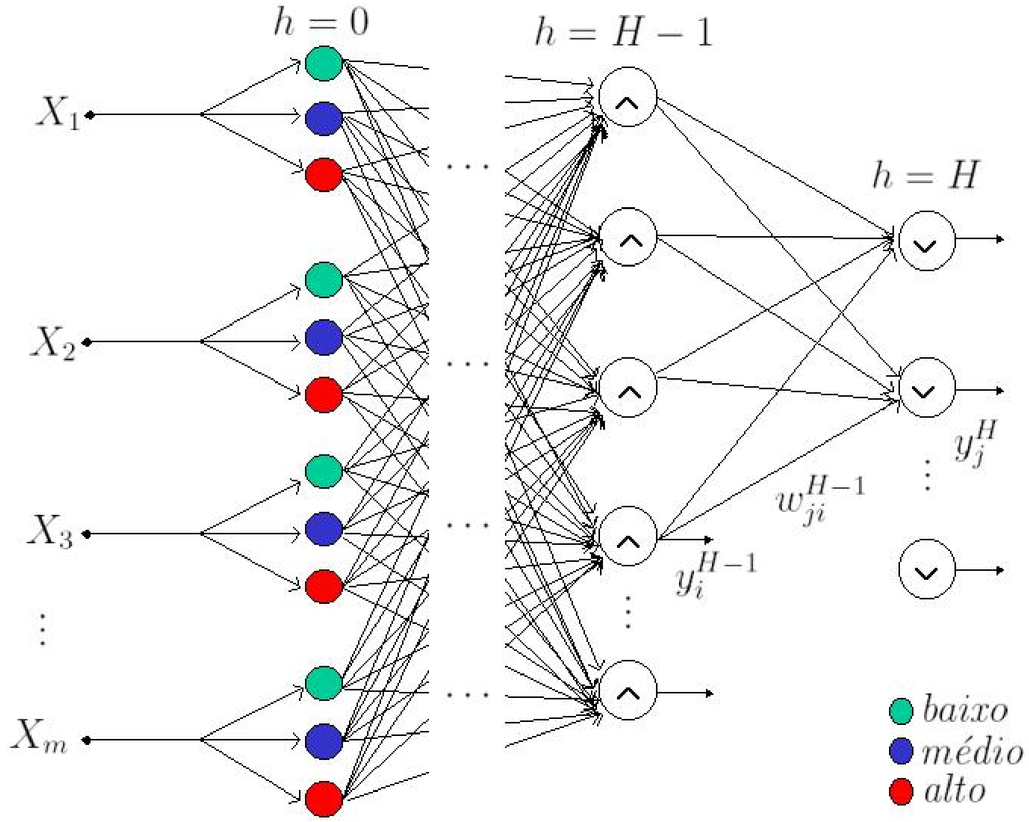

Figura 4.1: Topologia genérica de RND.

No primeiro passo, para a camada H, são selecionados os neurônios da camada precedente, $H-1$, que possuírem impacto positivo ($peso > 0$) na conclusão do neurônio de saída j. Logo, o i-ésimo neurônio da camada $H-1$ será selecionado se $w_{ji}^{H-1} > 0$. Admite-se que o conjunto de m_{H-1} neurônios da camada $H-1$, assim selecionados, seja denotado por $A^{H-1} = \{a_1^{H-1}, a_2^{H-1}, \ldots, a_{m_{H-1}}^{H-1}\}$ e o conjunto de seus respectivos pesos de conexão com o neurônio j na camada H dado por $\{wet_{a_1^{H-1}} = w_{ja_1}^{H-1}, \ldots, wet_{a_{m_{H-1}}^{H-1}} = w_{ja_{m_{H-1}}}^{H-1}\}$.

Para as camadas restantes, deverão ser obtidos os caminhos até a camada de entrada difusa, passando por esses neurônios e cujo somatório de pesos seja máximo. Com esse objetivo, o i-ésimo neurônio na camada $0 \leq h < H-1$ será selecionado, para que suas conexões façam parte de um caminho, se a saída y_i^h for positiva. Dentre os caminhos que partem do i-ésimo neurônio, serão selecionados aqueles cujo peso $w_{a_k i}^h$ seja positivo (sendo a_k o índice do neurônio a_k^{h+1}, $a_k^{h+1} \in A^{h+1}$) e a soma $wet_{a_k}^{h+1} + w_{a_k i}^h$ seja máxima. Denota-se esta soma por wet_i^h, qual seja,

$$wet_{i^h} \;=\; \max_{a_k^{h+1}} [wet_{a_k^{h+1}} + w_{a_k i}^h] \tag{4.1}$$

O conjunto de m_h neurônios, assim selecionados, será denotado por $A^h = \{a_1^h, a_2^h, \ldots, a_{m_h}^h\}$ e por $\{wet_{a_1^h}, wet_{a_2^h}, \ldots, wet_{a_{m_h}^h}\}$ o conjunto de seus respectivos pesos acumulados até o neurônio j na camada H ao longo do caminho com o somatório máximo de pesos dado pela Equação 4.1.

Do passo anterior resulta a seleção de m_0 neurônios (caminhos) partindo da camada de entrada difusa ($h = 0$), cujas saídas y_i^0 sejam graus de pertinência positivos a conjuntos difusos, correspondentes aos valores dos atributos para a entrada que foi classificada na classe j. Neste trabalho, os conjuntos difusos considerados serão: *baixo*, *médio* e *alto.* A partir dos m_0 caminhos serão geradas as cláusulas para compor o antececente das regras. Se $m_0 = 0$, para alguma entrada, não será gerada a regra correspondente que justifique sua classificação. Isto quer dizer que nenhum caminho apropriado pôde ser selecionado pelo passo anterior e o processo termina.

O próximo passo consiste em determinar as cláusulas que farão parte do antecedente da regra. Para isto, os m_0 neurônios são arranjados em ordem decrescente de seus impactos na rede. O impacto na rede ir definido para o i-ésimo neurônio como:

$$ir_i = y_i^0 wet_{i^0} \tag{4.2}$$

onde $i \in 1, 2, \ldots, m_0$, sendo este o conjunto dos índices pertencentes a A^0 e wet_i^0 definido por 4.1.

Vale dizer que, na topologia mostrada na Figura 4.1, estes m_0 neurônios são escolhidos entre os $3m$ neurônios da camada $h = 0$, sendo m o número de atributos observados nos objetos sendo classificados.

Da ordenação dos impactos na rede dados pela Equação 4.2 são escolhidos os subconjuntos de neurônios da camada de entrada difusa $I_s \subset A^0$ e $I_n = A^0 - I_s$, até que

$$\sum_{i_s} wet_{i_s^0} > 2 \sum_{i_n} wet_{i_n^0} \tag{4.3}$$

onde i_s e i_n são os índices dos neurônios escolhidos (I_s) e não-escolhidos (I_n), respectivamente e ainda $|I_s| + |I_n| = m_0$, sendo que $||$ representa a cardinalidade de um conjunto.

Tem-se determinado, até aqui, que o antecedente da regra terá $|I_s|$ cláusulas do tipo *atributo* é *conjunto*, com a garantia de que ficarão aquelas que tiveram maior influência para a classificação da entrada na classe j.

Para cada i_s, $1 \leq i_s \leq 3m$ será determinada uma cláusula. Para determinar-se o atributo da cláusula utiliza-se a fórmula a seguir:

$$t_{i_s} = (i_s - 1) \text{ div } 3 + 1 \tag{4.4}$$

onde t, $1 \leq t \leq m$, indica o índice do atributo X, $1 \leq i_s \leq 3m$ e div é o operador de divisão inteira. Para determinar-se o conjunto difuso da cláusula utiliza-se:

$$cd_{i_s} = \begin{cases} \textit{baixo} & \text{se } i_s - 3(t-1) = 1 \\ \textit{médio} & \text{se } i_s - 3(t-1) = 2 \\ \textit{alto} & \text{caso contrário} \end{cases} \tag{4.5}$$

Para compor o antecedente, usa-se o conectivo *e* para cláusulas com atributos diferentes e o conectivo *ou* para cláusulas com o mesmo atributo. No final deste processo tem-se uma regra para justificar a classificação de uma entrada na classe j.

Modificadores, tais como: *muito* e *mom* (mais ou menos) podem ser anexados aos conjuntos difusos dos antecedentes. Para isso, definem-se os vetores-padrão:

$$\begin{aligned}
baixo &= [0.9500 \quad 0.6000 \quad 0.0200] \\
muito\ baixo &= [0.9025 \quad 0.3600 \quad 0.0004] \\
mom\ baixo &= [0.9025 \quad 0.7746 \quad 0.1414] \\
m\acute{e}dio &= [0.7000 \quad 0.9500 \quad 0.7000] \\
muito\ m\acute{e}dio &= [0.4900 \quad 0.9747 \quad 0.4900] \\
mom\ m\acute{e}dio &= [0.8367 \quad 0.9025 \quad 0.8367] \\
alto &= [0.0200 \quad 0.6000 \quad 0.9500] \\
muito\ alto &= [0.0004 \quad 0.3600 \quad 0.9025] \\
mom\ alto &= [0.1414 \quad 0.7746 \quad 0.9025] \qquad (4.6)
\end{aligned}$$

em que *mom* significa *mais ou menos.*

Em [MP94] é calculada a distância média quadrática entre a entrada 3-dimensional correspondente ao atributo t_{i_s} e cada um dos vetores-padrão da Equação 4.6 que possuem o mesmo conjunto cd_{i_s} dado pela Equação 4.5. O vetor-padrão cuja distância seja mínima é selecionado para compor a cláusula. Estes modificadores foram usados na geração de regras a partir do conjunto *Vogal*, mostradas nas Tabelas 5.18, 5.19 e 5.20.

Finalmente, uma medida de certeza para a regra pode ser calculada pela equação:

$$cert_j^H = \frac{y_j^H}{\sum_{i=1}^{n_3} y_i^H} \qquad (4.7)$$

onde n_3 é o número de neurônios na camada de saída e $cert_j^H$, $0 \leq cert_j^H 1$, denota o percentual segundo o qual a saída y_j^H contribuiu para a saída total da rede. Quanto maior o valor de $cert_j^H$, menor é a dificuldade da rede em decidir qual a classe de saída j e daí maior é o grau de certeza dessa decisão. Se o sistema está apto a finalizar a decisão neste estágio, então pode-se seguir para a fase de justificação que é a fase de geração de regras. Adicionalmente, na rede de Mitra e Pal, se a rede não conseguir chegar a uma conclusão, segue-se para o modo de *querying* para tentar obter mais informação de entrada. Este esquema pode ser visualizado na Figura 3.2. A fase de justificação foi implementada, mas o modo de *querying* não o foi, porque fugia do escopo deste trabalho.

Para o cálculo dos modificadores da Equação 4.6 e da medida de certeza da Equação 4.7, é considerado o intervalo $[0, 1]$. As devidas conversões das entradas e saídas da RNDD foram feitas para tornar possível a aplicação destas equações.

Nota-se que esta heurística permite a seleção daqueles neurônios ativados com a entrada atual que representam os atributos que mais contribuem para a conclusão da rede para compor as cláusulas do antecedente da regra. Em outras palavras, nem todos os atributos de entrada precisam ser necessariamente selecionados para a geração das cláusulas do antecedente. Isso permite que o exemplo atual seja levado em consideração juntamente com os pesos sinápticos ajustados durante o treino na geração de uma regra para justificar a decisão da rede em resposta à apresentação desse exemplo.

Um especialista pode ser utilizado neste estágio para verificar a lógica da regra gerada pelo modelo. Isto pode ajudar a melhorar a confiança do usuário na eficiência do sistema. Além disso, essas regras podem ser utilizadas como uma base de conhecimento para um sistema especialista de classificação para o problema em questão [MP94].

Um exemplo de geração de regra

Para fins de ilustração, na Figura 4.2 é mostrado um exemplo simples de geração de regra com conclusão em $Classe_1$, considerando o modelo de RND apresentado na Figura 4.1. Os neurônios da camada de entrada difusa, da camada intermediária e da camada de saída são numerados da esquerda para a direita. O conjunto dos pesos maiores que zero, as entradas difusas y_i^0 e seus correspondentes termos lingüísticos são mostrados. A difusificação das m entradas reais produzem $3m$ entradas difusas, i. é., para cada atributo de entrada haverá três neurônios na camada de entrada difusa aplicando na rede os graus de pertinência aos conjuntos difusos *baixo*, *médio* e *alto*.

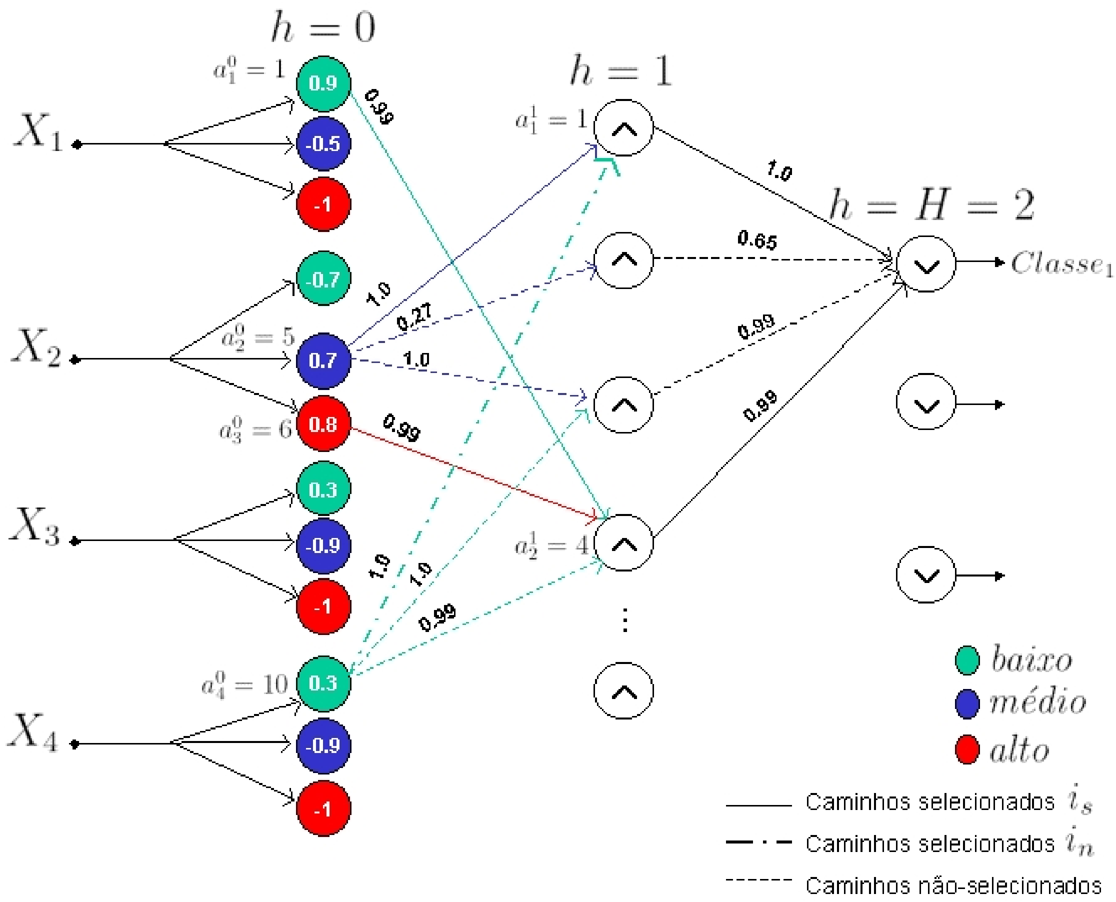

Figura 4.2: Exemplo de geração de regra por *backtracking*.

Inicialmente, são selecionados os neurônios da camada intermediária $h = 1$ cujos pesos das conexões com o neurônio $j = 1$ na camada de saída $h = 2$ sejam maiores que 0. Dessa maneira, o conjunto dos neurônios selecionados na camada $h = 1$ será $A^1 = a_1^1 = 1, a_2^1 = 4$ e $wet_{a_1^1} = 1.0, wet_{a_2^1} = 0.99$ o conjunto de suas respectivas conexões com o neurônio 1 na camada de saída:

Em seguida, são selecionados os neurônios na camada de entrada difusa $h = 0$ de acordo com a Equação 4.1. Dessa maneira, o conjunto dos neurônios selecionados na camada $h = 0$ será $A^0 = a_1^0 = 1, a_2^0 = 5, a_3^0 = 6, a_4^0 = 10$ e seus respectivos pesos acumulados até o neurônio 1 na camada de saída ao longo do caminho com o somatório máximo de pesos:

$$\begin{aligned} wet_{a_1^0} &= \max[(0.99 + 0.99)] = 1.98 \\ wet_{a_2^0} &= \max[(1.0 + 1.0), (0.65 + 0.27), (0.99 + 1.0)] = 2.0 \\ wet_{a_3^0} &= \max[(0.99 + 0.99)] = 1.98 \\ wet_{a_4^0} &= \max[(1.0 + 1.0), (0.99 + 1.0), (0.99 + 0.99)] = 2.0 \end{aligned}$$

Para cada neurônio selecionado na camada $h = 0$ é calculado o impacto na rede, tal como na Equação 4.2:

$$\begin{aligned} ir_{a_1^0} &= (0.9)(1.98) = 1.782 \\ ir_{a_2^0} &= (0.7)(2.0) = 1.4 \\ ir_{a_3^0} &= (0.8)(1.98) = 1.584 \\ ir_{a_4^0} &= (2.0)(0.3) = 0.6 \end{aligned}$$

Os caminhos representados por linhas sólidas e os representados por linhas traço-ponto terminam, respectivamente, nos neurônios dos conjuntos I_s e I_n, como determinado pela Equação 4.3. As linhas tracejadas indicam caminhos que não foram selecionados pela Equação 4.1. Ao se aplicar a desigualdade expressa na Equação 4.3 sobre os neurônios do conjunto A^0, obtém-se o subconjunto $I_s = a_1^0, a_2^0, a_3^0$ e o subconjunto $I_n = a_4^0$, I_s, já que $wet_{a_1^0} + wet_{a_2^0} + wet_{a_3^0} = 1.782 + 1.584 + 1.4 = 4.766$ é maior que $2wet_{a_4^0} = 2(0.6) = 1.2$.

Aplicando a Equação 4.4, sendo $a_1^0 = 1$, $a_2^0 = 5$ e $a_3^0 = 6$:

$$\begin{aligned} t_{a_1^0} &= (1 - 1) \text{ div } 3 + 1 = 1 \\ t_{a_2^0} &= (5 - 1) \text{ div } 3 + 1 = 2 \\ t_{a_3^0} &= (6 - 1) \text{ div } 3 + 1 = 2 \end{aligned}$$

Dessa forma, os atributos considerados serão X_1 e X_2. As cláusulas do antecedente da regra que justifica a classificação de um certo exemplo em $Classe_1$ são determinadas por:

$$\begin{aligned} conjuntodifuso_{a_1^0} &= \textit{baixo}, \text{ pois } 1 - 3(1 - 1) = 1 \\ conjuntodifuso_{a_2^0} &= \textit{médio}, \text{ pois } 5 - 3(2 - 1) = 2 \\ conjuntodifuso_{a_3^0} &= \textit{alto}, \text{ pois } 6 - 3(2 - 1) = 3 \end{aligned}$$

Nesse caso, a regra gerada pelo modelo para justificar sua conclusão como sendo $Classe_1$ seria:

Se (X_1 é *baixo*) *e* (X_2 é *médio ou* X_2 é *alto*) Então *classe* é $Classe_1$

Como pode ser notado, o conseqüente da regras se apresenta como uma conjunção de disjunções, além da cláusula simples no conseqüente. Isto é devido à topologia da rede

RNDD mostrada na Figura 3.1 e pelo critério de interpretação da saída da rede exposto na Seção 1.3.

Ao serem aplicados, os algoritmos estudados nesta seção podem gerar um grande número de regras, dificultando a análise do seu poder descritivo. Na próxima seção, serão apresentadas medidas de avaliação, bem como serão comentados dois métodos de pós-processamento de regras. O objetivo destas medidas e métodos é dar confiança ao usuário nas regras geradas e diminuir o número de regras que este deverá usar em suas decisões, focalizando sua atenção naquelas que possuam mais qualidade.

4.2 Avaliação de Regras

Geralmente, quando a RNA é orientada para tarefas descritivas de descoberta do conhecimento, o volume de regras geradas pode ser muito grande, o que dificulta a análise do poder descritivo dessas regras por parte do usuário. Logo, apesar de serem consideradas modelos de fácil compreensão e aplicação, a dificuldade de análise de um elevado número de regras pode tornar proibitivo o entendimento do conhecimento extraído. Existem várias medidas de avaliação (Piatetsky-Shapiro, 1991; Dean e Famili, 1997) *apud* [GOM02] e algoritmos de pós-processamento de regras [HLL99], [TKR$^+$95].

As medidas de avaliação utilizam técnicas estatísticas para avaliar a capacidade de representação do conhecimento adquirido sob a forma de regras. Essas medidas podem ser utilizadas para efetuar uma análise de como essas regras são sustentadas pelos exemplos, i. é., essas medidas podem ser utilizadas a fim de efetuar uma análise qualitativa das regras. Os algoritmos de pós-processamento visam reduzir as regras originais a um conjunto pequeno para que o usuário possa focalizar sua atenção sob aquelas regras melhor sustentadas pelos dados, i. é., focalizar sua atenção sob as regras de maior qualidade.

Neste projeto, a rede RNDD é treinada com um conjunto de treino representativo da população dos dados. Ao final deste processo, o conhecimento sobre os dados está codificado nos pesos de suas conexões. Estes são fixados e a etapa seguinte consiste em testar a rede usando um conjunto de teste também representativo da população dos dados e cujos exemplos não foram usados para o treino da rede. O poder de classificação da rede, para dados do mesmo domínio em que ela foi treinada, é avaliado com o conjunto de teste. A aplicação desta metodologia é explicada com mais detalhes no Capítulo 5.

No caso de o usuário desejar uma explicação da decisão da rede, ou também, se desejar ter, invés da rede, um conjunto de regras classificadoras dos dados, este deverá aplicar um algoritmo de geração de regras a partir da rede treinada, ou seja, as regras serão obtidas com exemplos de treino. Quando o conjunto de regras gerado vai ser usado como um classificador, este deve ser avaliado usando um conjunto de teste. Para cada regra, o conjunto de teste é usado para calcular uma matriz de contingência, exemplificada na Seção 4.2.1 e a partir desta serão calculadas as medidas de avaliação para cada regra. Daí pode-se selecionar as regras com as melhores medidas e usá-las para classificar dados do mesmo domínio do conjunto de treino e de teste. Esta metodologia, que também corresponde a um pós-processamento de regras, foi aplicada na Seção 5.2.

Quando uma regra gerada é usada para justificar (explicar ou explanar) a decisão da rede ao usuário, esta deve satisfazê-lo, de acordo com sua experiência ou, se não, a regra deve ser avaliada usando um conjunto de teste.

No presente trabalho, como não se tem um usuário experiente quanto aos dados em estudo, foram geradas regras para exemplos de entrada que se situam em regiões do espaço de características que provavelmente seriam classificadas pelo especialista numa certa classe e com esta previsão a regra foi julgada satisfatória ou não. Isto foi feito na Seção, na análise das Tabelas 5.20, 5.18 e 5.19.

O objetivo desta seção é apresentar diversas medidas de avaliação de regras e mostrar dois dos métodos de pós-processamento de regras existentes na literatura baseados no cálculo dessas medidas.

4.2.1 Medidas de avaliação de regras

Nesta seção, serão enumeradas 3 medidas objetivas de avaliação de regras baseadas em termos de freqüências relativas calculadas a partir da tabela da contingência de cada regra. A Tabela 4.1 traz um exemplo genérico de tabela de contingência em que H denota o conjunto de exemplos para os quais o conseqüente da regra é verdadeiro, sendo $\overline{H}$, seu complemento; B denota o conjunto de exemplos para os quais o antecedente da regra é verdadeiro, sendo $\overline{B}$, seu complemento; [5] h é o número de exemplos para os quais o conseqüente da regra é verdadeiro, sendo $\overline{h}$, seu complemento; b é o número de exemplos para os quais o antecedente da regra é verdadeiro, sendo $\overline{b}$, seu complemento; hb é o número de exemplos para os quais são verdadeiros o antecedente e o conseqüente da regra; $\overline{hb}$ é o número de exemplos para os quais são falsos o antecedente e o conseqüente da regra; $\overline{h}b$ é o número de exemplos para os quais o conseqüente da regra é falso e o antecedente é verdadeiro; $h\overline{b}$ é o número de exemplos para os quais o conseqüente da regra é verdadeiro e o antecedente é falso; e n é o número total de exemplos.

Tabela 4.1: Tabela de contingência para uma regra qualquer.

	H	$\overline{H}$	
B	hb	$\overline{h}b$	b
$\overline{B}$	$h\overline{b}$	$\overline{hb}$	$\overline{b}$
	h	$\overline{h}$	n

A seguir, serão apresentadas as seguintes medidas de avaliação de regras: confiança, suporte e novidade.

Dada uma regra Reg_1 qualquer com seu conseqüente H e seu antecedente B tem-se as seguintes definições:

[5] A notação H, comumente utilizada neste contexto, tem origem no vocábulo inglês *head* que significa cabeça, o conseqüente da regra. A notação B se origina de *body* que significa corpo, o antecedente da regra.

Definição 4.1 (Confiança) $Conf(Reg_1) = P(H|B)$

A confiança ou precisão de uma regra Reg_1 é uma medida do quanto uma regra é específica para o problema. É definida por (Lavrac *et al.*, 1999)*apud* [GOM02] como a probabilidade condicional de o conseqüente da regra ser verdadeiro dado que o antecedente também é verdadeiro. Assim:

$$Conf(Reg_1) = P(H|B) = \frac{P(HB)}{P(B)} = \frac{f_{hb}}{f_b} = \frac{hb}{b} \tag{4.8}$$

onde f denota freqüência. A confiança de uma regra destina-se a avaliar regras individuais e por essa razão tende a favorecer a precisão de exemplos positivos, ou seja, mede a corretude dos resultados retornados. Como tal, difere da precisão do conjunto de regras que constituem a hipótese (classificador), definida a partir da matriz de confusão que será mostrada na Seção 5.1. A precisão do conjunto de regras é usada como medida padrão para avaliar hipóteses.

Definição 4.2 (Suporte) $Sup(Reg_1) = P(HB)$

O suporte de uma regra é uma medida relativa do número de exemplos cobertos corretamente pela regra Reg_1. É definida como:

$$Sup(Reg_1) = P(HB) = f_{hb} = \frac{hb}{n} \tag{4.9}$$

Também conhecida como freqüência. Quanto maior o suporte, maior o número de exemplos cobertos pelo corpo e pela cabeça da regra Reg_1. Vale a pena notar que, diferentemente das medidas anteriores, o suporte é simétrico em H e B, ou seja,

$$Sup(Reg_1) = P(HB) = P(BH)$$

Definição 4.3 (Novidade) $Nov(Reg_1) = P(HB) - P(H)P(B)$

A medida de novidade tem a intenção de mostrar o quanto uma regra é interessante, nova ou fora do comum. Essa medida foi inicialmente apresentada em (Piatetsky-Shapiro, 1991) *apud* [GOM02]. A seguir é definida uma medida de novidade utilizando a estrutura da tabela de contingência.

Defini-se uma regra Reg_1 como nova se a probabilidade de B e H ocorrerem juntos não puder ser inferida pelas probabilidades de B e H isoladamente, isto é, B e H não são estatisticamente independentes.

A medida de novidade é obtida comparando o valor esperado $P(HB)$ com os valores de $P(H)$ e $P(B)$. Quanto mais o valor diferir do observado, maior é a probabilidade de que exista uma correlação verdadeira e inesperada entre B e H. Pode ser demonstrado que $-0.25 \leq Nov(Reg_1) \leq 0.25$ e que quanto maior, mais forte é a associação entre B e H enquanto que, quanto menor, mais forte é a associação entre B e $\overline{H}$.

A novidade de uma regra é então definida como:

$$Nov(Reg_1) = P(HB) - P(H)P(B) = f_{hb} - f_h \cdot f_b = \frac{1}{n}(hb - \frac{h \cdot b}{n}) \qquad (4.11)$$

As medidas aqui apresentadas foram calculadas para as regras geradas para o conjunto *Iris* na Seção 5.2.2 com intuito de selecionar aquelas com mais qualidade. O conjunto original de 75 regras foi reduzido a um pequeno subconjunto com 18 regras. Esta redução é o que traduz a idéia dos métodos de pós-processamento que serão apresentados na próxima seção.

4.2.2 Pós-processamento de regras

Em seguida, serão abordados alguns estudos com o objetivo de reduzir o número de regras, mantendo ao máximo a qualidade daquelas do conjunto original. Existem diferentes métodos, entre eles, a eliminação de regras redundantes e a identificação de regras mais ou menos interessantes.

Poda

Uma das formas encontradas para se reduzir o número de regras produzidas pelos algoritmos de geração de regras é a utilização da "poda"[6] de regras redundantes [TKR+95]. Neste método, o objetivo é eliminar regras que não oferecem nenhuma informação adicional. Dessa maneira, identificam-se as regras que descrevam os mesmos registros de uma base de dados, i. é., regras cujos antecedentes sejam mais específicos mas não possuam um valor de confiança mais elevado. Abaixo, segue um exemplo que pode ajudar na compreensão da idéia. Supondo que foram obtidas duas regras:

1. Se (X_1 é A_1) *e* (X_2 é A_2) Então *classe* é $Classe_1$
   $Conf(1) = 0.90$, $Sup(1) = 0.02$

2. Se (X_1 é A_1) *e* (X_2 é A_2) *e* (X_3 é A_2) Então *classe* é $Classe_1$
   $Conf(2) = 0.90$, $Sup(2) = 0.01$

Segundo [TKR+95], a segunda regra é redundante, sendo mais específica que a primeira, porém não introduzindo nenhuma informação adicional, uma vez que a confiança *Conf* mantém-se inalterada e o suporte da segunda reduz-se pela metade. Considera-se que a primeira "cobre" a segunda.

Estimativa da interessabilidade

Em [HLL99] são propostas medidas relativas para a estimativa do quão interessante é uma regra. Para esse pesquisador, uma regra interessante está sempre relacionada com

[6]Do Inglês: *pruning*.

o conhecimento e necessidades de quem a está analisando porque sempre há uma subjetividade inerente a este tipo de classificação. Contudo, sob o ponto de vista objetivo, também pode-se identificar uma regra interessante, recorrendo a medidas concretas.

Para a classificação de regras interessantes, [HLL99] utilizou as medidas de avaliação objetivas, isto é, passíveis de serem calculadas: confiança, suporte e novidade e uma medida subjetiva: a surpresa, que representa o quanto uma regra "vai contra" o que seria de se esperar. Uma regra pode ser inesperada face ao que o usuário pensa ou inesperada face às regras do senso comum.

No próximo capítulo, serão apresentados os conjuntos de dados *Vogal*, *Iris* e *Alea*. Também serão mostrados resultados de treino, teste e geração de regras obtidos, bem como será aplicado os métodos de pós-processamento por poda e por estimativa da interessabilidade nas regras geradas com o conjunto *Iris*.

Capítulo 5

Experimentos e Resultados

No aprendizado supervisionado, o conjunto de dados é dividido em dois subconjuntos: o conjunto de treino e o conjunto de teste. O conjunto de treino é usado para treinar a rede. Depois de treinada, a rede é avaliada com o conjunto de teste, por meio do cálculo de medidas de precisão, tais como o coeficiente Kappa e a taxa de acerto. Se obtiver um bom desempenho como classificador, a rede treinada e o conjunto de treino podem ser utilizados para gerar regras por *backtracking*. Essas regras podem ser avaliadas utilizando-se medidas de avaliação calculadas sobre o conjunto de teste. Observa-se que tanto o conjunto de treino quanto o conjunto de teste devem ser representativos da população dos dados.

O objetivo principal deste capítulo é apresentar os resultados da aplicação da RNDD sobre os conjuntos de dados *Iris*, *Vogal* e *Alea*. As próximas seções se ocuparão de apresentar estes conjuntos, bem como mostrar resultados de treino e resultados de teste. Resultados de geração de regras a partir da rede treinada também serão mostrados, bem como serão calculadas medidas de avaliação para pós-processar as regras geradas a partir do *Iris*.

5.1 Medidas de Precisão da Classificação

Um método para se avaliar o desempenho de um algoritmo de classificação consiste em comparar seus resultados de classificação com os resultados obtidos por outros classificadores para derivar sua precisão. Para padrões classificados do conjunto de teste, uma matriz de confusão[1] pode ser calculada. Na matriz de confusão, é mostrado o número de classificações corretas preditas pelo classificador em oposição ao número de classificações preditas por um especialista. Na Tabela 5.1, é mostrada uma matriz de confusão, considerando um problema de classificação com duas classes, onde Vp é o percentual de exemplos corretamente classificados como sendo da classe C_1, Fp é o percentual de exemplos da classe C_2 erroneamente classificados como sendo da classe C_1, Vn é o percentual de exemplos corretamente classificados como sendo da classe C_2 e Fn é o percentual de

[1] Neste trabalho, o termo matriz de confusão refere-se ao classificador, enquanto que a tabela de contingência, definida na Seção 4.2.1, refere-se a uma única regra.

exemplos da classe C_1 erroneamente classificados como sendo da classe C_2. A diagonal principal da matriz de confusão indica a concordância entre os dois conjuntos de dados.

Tabela 5.1: Matriz de confusão genérica.

	Classificador		Total
Referência	C_1	C_2	
C_1	Vp	Fn	$Vp + Fn$
C_2	Fp	Vn	$Fp + Vn$
Total	$Vp + Fp$	$Fn + Vn$	1.0

A matriz de confusão permite que várias medidas de precisão sejam calculadas, entre elas, a taxa de acerto e o coeficiente Kappa. A soma da diagonal da matriz é a taxa de acerto. O coeficiente Kappa foi dado por [COH60]. Esse coeficiente foi recomendado como uma medida de precisão aceitável na temática da classificação por representar a matriz de confusão como um todo. Ele leva em consideração todos os elementos da matriz de confusão, ao invés de apenas os elementos diagonais, como ocorre com o cálculo da taxa de acerto.

O coeficiente Kappa k é definido por:

$$k = \frac{p_o - p_c}{1 - p_c} \tag{5.1}$$

onde $p_o = Vp + Vn$ e $p_c = (Vp + Fp)(Vp + Fn) + (Fn + Vn)(Fp + Vn)$ indicam, respectivamente, a proporção de unidades que concordam[2] e a proporção de unidades para a concordância esperada.

5.2 Conjunto de Dados *Iris*

Em [AND35] foram observadas 150 flores íris quanto aos atributos: comprimento e largura da sépala (CS e LS) e comprimento e largura da pétala (CP e LP), em centímetros. Foram medidas 50 plantas de cada sub-espécie: Íris Setosa, Íris Versicolor e Íris Virgínica que, para fins de simplificação, serão aqui denominadas pelas siglas *SE*, *VC* e *VG*, respectivamente. Assim, o conjunto de dados *Iris*[3] é formado por 150 linhas com 4 colunas.

Este conjunto de dados tornou-se favorito para a análise de agrupamento[4] e classificação porque é pequeno e apresenta uma classe bastante separada (*SE*) e duas com um certo grau de sobreposição (*VC* e *VG*), o que facilita uma avaliação inicial para algoritmos de aprendizagem. As características da pétala são as que melhor discriminam as três classes. A Tabela 5.2 traz uma amostra dos dados de cada espécie.

[2] Taxa de acerto.

[3] Disponível no endereço eletrônico *http://www.ics.uci.edu/~mlearn.*

[4] O sentido aqui é do vocábulo inglês *cluster*, amplamente usado na literatura e do qual deriva o termo "clusterização", sinônimo de aglomeração.

Tabela 5.2: Amostra do conjunto de flores *Iris*.

Espécie	*CS*	*LS*	*CP*	*LP*
SE	4.40	3.20	1.30	0.20
...	...	...	...	...
SE	5.10	3.30	1.70	0.50
VC	5.10	2.50	3.00	1.10
...	...	...	...	...
VC	5.20	2.70	3.90	1.40
VG	5.80	2.70	5.10	1.90
...	...	...	...	...
VG	5.90	3.00	5.10	1.80

Na Tabela 5.3, são fornecidos os intervalos dos valores das medidas dos 4 atributos sobre as 3 espécies.

Tabela 5.3: Intervalos das medidas para as flores *Iris*.

Espécie	*CS*	*LS*	*CP*	*LP*
SE	[4.3, 5.8]	[2.3, 4.4]	[1.0, 1.9]	[0.1, 0.6]
VC	[4.9, 7.0]	[2.0, 3.4]	[3.0, 5.1]	[1.0, 1.8]
VG	[4.9, 7.9]	[2.2, 3.8]	[4.5, 6.9]	[1.4, 2.5]

A classificação do conjunto *Iris* é um problema conhecido e explorado na literatura [WKB97], razão pela qual foi um dos escolhidos para testar a rede implementada.

5.2.1 Classificação

Método *Holdout*

No teste *Holdout*[5] $perc\% - (100 - perc)\%$, proposto por Fukunaga [FUK90], o conjunto de dados é dividido aleatoriamente em $perc\%$ dos padrões para treinamento e $(100 - perc)\%$ dos padrões para teste. Em ambos os conjuntos, os exemplos são distribuídos proporcionalmente em cada classe.

O primeiro teste foi realizado com $perc = 70$ (*Holdout* 70% − 30%). A rede foi treinada com os seguintes parâmetros: taxa de aprendizagem $\eta = 0.01$ (fixo), momento $\alpha = 0.1$ (fixo), número máximo de épocas $maxepoca = 700$, número de neurônios na camada intermediária $n_2 = 8$ e parâmetro de sobreposição das funções de pertinência $Sobrep = 0.55$.

[5] Optou-se pelo uso do termo em Inglês amplamente utilizado pela comunidade acadêmica, pois não possui uma tradução adequada para a língua portuguesa.

Vale lembrar que os valores máximos e mínimos de cada atributo do conjunto de dados são usados para calcular os parâmetros λ e c das funções de pertinência descritas na Seção 3.3.1.

Em todos os testes, os últimos pesos do treino são fixados na rede, os exemplos de teste são submetidos e pré-processados pelas mesmas funções de pertinência do treino com os mesmos parâmetros λ, c e *Sobrep*. Os demais parâmetros utilizados no treino também são mantidos para o cálculo da saída da rede.

Na Tabela 5.4, é mostrada a evolução do EQM durante o treino para as 700 épocas. Pode ser visto que o erro decresce a cada época. No entanto, para a maioria das redes neurais é esperado um EQM de 0.05 ou menos. Como poderá ser observado adiante, apesar de ter atingido um EQM relativamente alto, a RNDD obteve bons resultados de classificação.

Tabela 5.4: Evolução do EQM por época - RNDD.

Época	EQM
1	1.5773
2	1.5682
3	1.5595
121	1.2812
126	1.2802
480	0.6806
698	0.3943
699	0.3936
700	0.3930

Das 45 amostras do teste, apenas 1 foi classificada incorretamente segundo a classificação de Anderson [AND35] utilizada como referência, conferindo à RNDD uma taxa de acerto de 97.77%. Essa amostra classificada de maneira incorreta pela RNDD como pertencente à classe *VG* é, segundo a classificação do taxionomista Anderson, pertence à classe *VC*. A Figura 5.1 traz a plotagem das amostras de teste com a classificação feita pela RNDD e a indicação do padrão incorretamente classificado. Como pode ser notado, este padrão encontra-se muito próximo à superfície de decisão que separa as classes sobrepostas *VC* e *VG*.

A matriz de confusão para o teste *Holdout* $70\% - 30\%$ é mostrada na Tabela 5.5. O coeficiente Kappa [COH60] calculado foi de 0.9667 e a taxa de acerto foi de 97.77%.

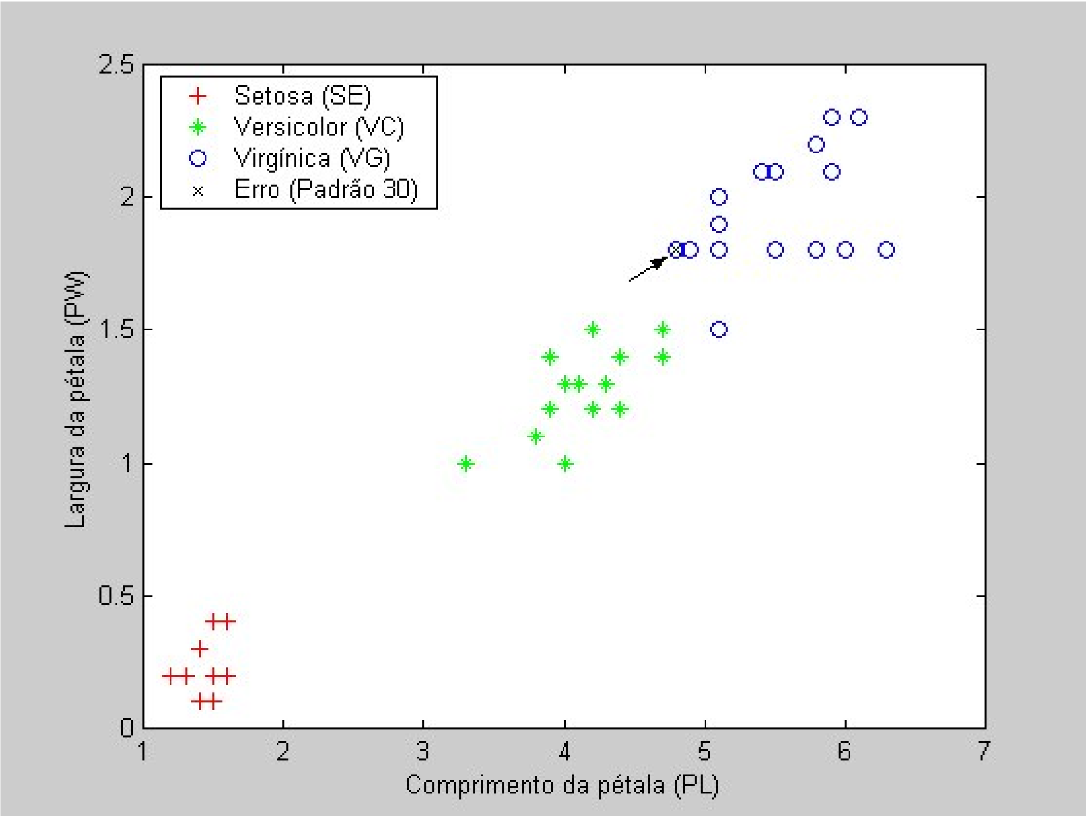

Figura 5.1: O padrão 30, incorretamente classificado.

Tabela 5.5: Matriz de confusão - RNDD (*Holdout* $70\% - 30\%$).

	Classificador		
Referência	*SE*	*VC*	*VG*
SE	0.33	0	0
VC	0	0.31	0.02
VG	0	0	0.33

O segundo teste foi realizado com $perc = 50$ (*Holdout* $50\% - 50\%$). A rede foi treinada com os seguintes parâmetros: $\eta = 0.01$ (fixo), $\alpha = 0.1$ (fixo), $maxepoca = 3000$, $n_2 = 6$ e $Sobrep = 1.0$. O coeficiente Kappa calculado foi de 0.94 e a taxa de acerto foi de 96.00%.

Método Rotação 2-dobras

No teste Rotação 2-dobras, também proposto por Fukunaga [FUK90], o conjunto de dados é dividido aleatoriamente: metade para treino e metade para teste. O teste é feito e as medidas de precisão são calculadas. Depois, invertem-se os conjuntos de treino e de teste e os cálculos das medidas de precisão são refeitos. A medida de desempenho final é dada pela média dos índices obtidos nas duas etapas.

Para o teste de Rotação 2-dobras, foi utilizado o conjunto de dados *Iris_CR*, uma variação do *Iris* resultante da centralização e redução por atributo da base de dados original, o que significa dizer que estes dados foram submetidos a uma rotina que normaliza cada atributo para possuir variância unitária. Este método foi aplicado para permitir uma comparação com os resultados obtidos por diversos classificadores reportados por Woods e colaboradores. Importante frisar que, nesse artigo, os classificadores foram utilizados com seus respectivos parâmetros otimizados.

A rede foi treinada com os seguintes parâmetros: $\eta = 0.01$ (fixo), $\alpha = 0.1$ (fixo), $maxepoca = 1000$, $n_2 = 5$ e $Sobrep = 0.55$. Na Tabela 5.6, são mostradas as matrizes de confusão para o teste. O coeficiente Kappa médio calculado foi de 0.94 e a taxa de acerto média foi de 96.00%.

Tabela 5.6: Matrizes de confusão - RNDD (Rotação 2-dobras *Iris_CR*).

Rotação 1	Classificador		
Referência	*SE*	*VC*	*VG*
SE	0.33	0	0
VC	0	0.32	0.01
VG	0	0.03	0.31

Rotação 2	Classificador		
Referência	*SE*	*VC*	*VG*
SE	0.33	0	0
VC	0	0.32	0.01
VG	0	0.03	0.31

Na Tabela 5.7, são sumarizados os resultados de classificação obtidos pela RNDD sobre os dados *Iris* utilizando as metodologias de teste apresentadas na Seção **??**. O teste *Holdout* $50\% - 50\%$ será utilizado para a geração das regras apresentadas na Seção 5.2.2.

Na Tabela 5.8, é mostrada uma comparação entre a precisão obtida por diversos classificadores sobre o *Iris_CR* e a precisão da RNDD. Conforme pode ser observado, o

Tabela 5.7: Resultados de classificação da RNDD.

Método de teste	Kappa	Taxa de acerto	Dados
Holdout 70% – 30%	0.9667	97.77%	*Iris*
Holdout 50% – 50%	0.9200	94.67%	*Iris*
Rotação 2-dobras	0.9400	96.00%	*Iris_CR*

desempenho de classificação obtido pela RNDD foi somente inferior àquele obtido pelo método de classificação Bayes Linear.

Tabela 5.8: Comparativo da precisão de classificação.

Método de classificação	Acerto
K-Nearest Neighbor	92.00%
Rede Neural	95.33%
Árvore de Decisão C4.5	92.67%
Bayes Quadrático	95.33%
Bayes Linear	97.33%
RNDD	96.00%

Como esperado, os resultados de classificação dos dados *Iris* foram muito bons, já que este conjunto apresenta pouca sobreposição. Em um conjunto de dados onde há muita sobreposição ou em situações em que é difícil para um especialista classificar pela sua experiência, conseguir uma rede neural que classifique corretamente 40% ou mais das observações pode ser muito vantajoso em relação ao que o especialista poderia fazer.

A classificação da RNDD será ainda avaliada com outros dois conjuntos: o *Vogal* e o *Alea*. A próxima seção trará os resultados de geração de regras sobre o *Iris*.

5.2.2 Geração de regras

Dada a RNDD treinada e um conjunto de exemplos, é possível gerar regras mediante o uso de um algoritmo específico. Particularmente nesta seção foi utilizado o conjunto de treino para geração das regras para possibilitar que essas mesmas regras fossem avaliadas com o conjunto de teste. A idéia é eliminar redundâncias na avaliação das regras.

Na Tabela 5.9, é mostrada uma seleção das 75 regras difusas obtidas pela aplicação do algoritmo *backtracking* à RNDD treinada, a partir da apresentação de 75 exemplos do treino da divisão feita pelo método *Holdout* 50% – 50%, cujos resultados foram apresentados na Seção 5.2.1. Este foi o teste escolhido para a geração das regras porque dividiu a base de dados em dois conjuntos com o mesmo número de exemplos: metade para a geração de regras e a outra metade para avaliação das mesmas na Seção 5.2.3.

A coluna Id denota a identificação da regra. Na coluna Regra, é mostrada a regra difusa obtida pelo módulo de geração de regras. A coluna Certeza traz a medida $cert_j^H$ calculada segundo a Equação 4.7 para o j-ésimo neurônio com a saída máxima. Para evitar que fossem produzidas medidas de certeza negativas no cálculo de $cert_j^H$, as saídas da RNDD foram transformadas para o intervalo unitário $[0, 1]$.

Tabela 5.9: Regras difusas do *Iris* obtidas por *backtracking*.

Id	Regra	$cert_j^H$
8	Se (*CS* é *baixo*) *e* (*LP* é *baixo*) *e* (*LS* é *médio*) Então *classe* é *SE*	0.95
9	Se (*LP* é *baixo*) *e* (*LS* é *baixo ou LS* é *médio*) Então *classe* é *SE*	0.92
10	Se (*CS* é *baixo ou CS* é *médio*) *e* (*LS* é *médio*) Então *classe* é *SE*	0.95
16	Se (*CS* é *baixo ou CS* é *médio*) *e* (*LP* é *baixo*) Então *classe* é *SE*	0.90
22	Se (*CS* é *baixo*) *e* (*LP* é *baixo*) Então *classe* é *SE*	0.96
25	Se (*CS* é *baixo*) *e* (*LS* é *médio*) Então *classe* é *SE*	0.96
26	Se (*CP* é *alto*) *e* (*CS* é *baixo ou CS* é *médio*) *e* (*LS* é *médio*) Então *classe* é *VC*	0.58
39	Se (*CP* é *alto*) *e* (*CS* é *médio*) *e* (*LS* é *baixo ou LS* é *médio*) Então *classe* é *VC*	0.58
40	Se (*CS* é *baixo ou CS* é *médio*) *e* (*LP* é *baixo*) *e* (*LS* é *baixo*) Então *classe* é *VC*	0.71
42	Se (*CP* é *alto*) *e* (*CS* é *médio ou CS* é *alto*) *e* (*LS* é *baixo ou LS* é *médio*) Então *classe* é *VC*	0.55
48	Se (*CP* é *alto*) *e* (*CS* é *médio ou CS* é *alto*) *e* (*LS* é *baixo*) Então *classe* é *VC*	0.63
49	Se (*CP* é *alto*) *e* (*CS* é *médio ou CS* é *alto*) *e* (*LS* é *médio*) Então *classe* é *VC*	0.54
58	Se (*LP* é *alto*) *e* (*LS* é *baixo ou LS* é *médio*) Então *classe* é *VG*	0.95
60	Se (*CS* é *alto*) *e* (*LP* é *alto*) *e* (*LS* é *baixo ou LS* é *médio*) Então *classe* é *VG*	0.88
63	Se (*CS* é *médio*) *e* (*LP* é *alto*) *e* (*LS* é *baixo ou LS* é *médio*) Então *classe* é *VG*	0.70
64	Se (*CS* é *alto*) *e* (*LP* é *alto*) *e* (*LS* é *médio*) Então *classe* é *VG*	0.87
65	Se (*CS* é *médio ou CS* é *alto*) *e* (*LP* é *alto*) *e* (*LS* é *baixo*) Então *classe* é *VG*	0.78
66	Se (*CS* é *alto*) *e* (*LP* é *alto*) *e* (*LS* é *alto*) Então *classe* é *VG*	0.97
74	Se (*LP* é *alto*) *e* (*LS* é *baixo*) Então *classe* é *VG*	0.98

O critério usado para a geração do conseqüente[6] foi o do "vencedor-leva-tudo" como explicado na Seção 1.3. Somente 2 regras foram geradas com o conseqüente incorreto para os exemplos de treino que as originaram.

Pode-se notar que as regras para a classe *SE* são as que possuem os valores mais altos de $cert_j^H$. Isto se justifica, já que esta é a classe mais separada das outras e é um

[6]Nesse caso, o conseqüente será a classe à qual pertence o exemplo.

indicativo de que a decisão foi tomada mais facilmente pela RNDD. Os modificadores e os qualificadores citados na Seção 4.1.3 não foram utilizados aqui para facilitar a aplicação da metodologia de avaliação das regras geradas.

5.2.3 Avaliação das regras geradas

Para as regras apresentadas na Tabela 5.9, são mostradas as seguintes medidas calculadas a partir das respectivas matrizes de contingência: confiança (*Conf*), suporte (*Sup*) e novidade (*Nov*). Os valores das medidas são mostrados na Tabela 5.10 em ordem decrescente da soma das colunas *Conf* e *Sup*. Vale lembrar que, para a coluna *Sup*, o valor máximo esperado é 0.33, já que os exemplos estão equitativamente distribuídos nas 3 classes consideradas.

Tabela 5.10: Medidas de avaliação das regras originais.

Id	Classe	*Conf*	*Sup*	*Nov*
22	*SE*	0.96	0.32	0.21
16	*SE*	0.89	0.33	0.21
25	*SE*	1.00	0.21	0.14
8	*SE*	1.00	0.21	0.14
58	*VG*	0.85	0.29	0.18
9	*SE*	0.86	0.25	0.16
74	*VG*	0.88	0.20	0.12
64	*VG*	1.00	0.07	0.04
65	*VG*	0.87	0.17	0.11
60	*VG*	0.90	0.12	0.08
10	*SE*	0.80	0.21	0.12
66	*VG*	1.00	0.01	0.01
63	*VG*	0.79	0.15	0.08
40	*VC*	0.50	0.04	0.01
26	*VC*	0.50	0.03	0.01
42	*VC*	0.35	0.16	0.01
49	*VC*	0.42	0.07	0.01
48	*VC*	0.32	0.09	0.00
39	*VC*	0.32	0.08	0.00

Essas medidas foram escolhidas por serem medidas objetivas utilizadas nos métodos de pós-processamento por poda e por estimativa da interessabilidade comentados na Seção 4.2.2. Estas regras foram selecionadas do conjunto das 75 regras originais por apresentarem os maiores valores destas medidas. Dessa forma, o especialista recebe uma indicação para analisar mais cuidadosamente esse pequeno conjunto de regras, ao invés de fazê-lo com o conjunto todo.

Quanto à surpresa, por ser uma medida de caráter subjetivo do pós-processamento por estimativa da interessabilidade, não é passível de ser calculada, sendo possível apenas estimá-la. A surpresa se manifesta no fato de todas as regras selecionadas possuírem em seus antecedentes os atributos da sépala (LS e CS), ainda que os atributos que mais discriminem sejam os da pétala (LP e CP). Outro ponto a considerar é a presença de determinadas cláusulas nas regras da Tabela 5.9 que predizem as classes consideradas. Naquelas que predizem a classe *SE*, destaca-se a presença da cláusula *CS* é *baixo*. Nas que predizem a classe *VC*, nota-se em todas, a presença da cláusula *CS* é *médio*. Finalmente, a cláusula *LP* é *alto* está presente em todas as regras que predizem a classe *VG*.

Submetendo-se esse conjunto reduzido de regras ao método de pós-processamento por poda, identifica-se que a regra 8 é coberta pela regra 25, sendo a primeira mais específica do que a segunda, não aumentando a confiança nem o suporte, portanto passível de ser eliminada. Dessa maneira, elimina-se do conjunto pós-processado a regra 8.

Numa situação real, se a rede tem uma boa taxa de acertos e as regras geradas são representativas da população dos dados, então as regras, depois de pós-processadas, poderão ser usadas como um classificador para o domínio de dados em estudo. Neste trabalho, esta avaliação do conjunto das regras originalmente geradas e do conjunto de regras pós-processadas como classificadores não foi feita, mas poderia ser levada em conta para estimar o quanto as regras pós-processadas representam o conjunto de regras originais.

5.3 Conjunto de Dados *Vogal*

O conjunto de dados *Vogal*[7] [PM77] é formado por 871 linhas com 4 colunas cada (3 atributos e a classe), representando os sons de 6 vogais (∂, a, i, u, e, o) do idioma indiano Telugu. Os 3 atributos: F_1, F_2 e F_3 correspondem às freqüências formantes, obtidas por meio de uma análise espectral da voz de três locutores masculinos na faixa etária de 30 a 35 anos. Este foi um dos conjuntos de dados utilizados em [MP94] cujos resultados serão usados para comparação nesta seção. As 6 classes deste conjunto apresentam sobreposição como pode ser visto na projeção bidimensional mostrada na Figura 5.2.

Na Tabela 5.11 é fornecida uma amostra dos dados de cada classe.

Tabela 5.11: Amostra do conjunto de vogais.

Classe	F_1	F_2	F_3	Classe	F_1	F_2	F_3
∂	550	2000	2700	u	350	1000	2600
...	...	...	...	...	...	...	...
a	650	1050	2400	e	400	1300	2400
...	...	...	...	...	...	...	...
i	350	2250	2700	o	450	1100	2300
...	...	...	...	...	...	...	...

[7]Disponível no endereço eletrônico *http://www.isical.ac.in/˜sushmita*.

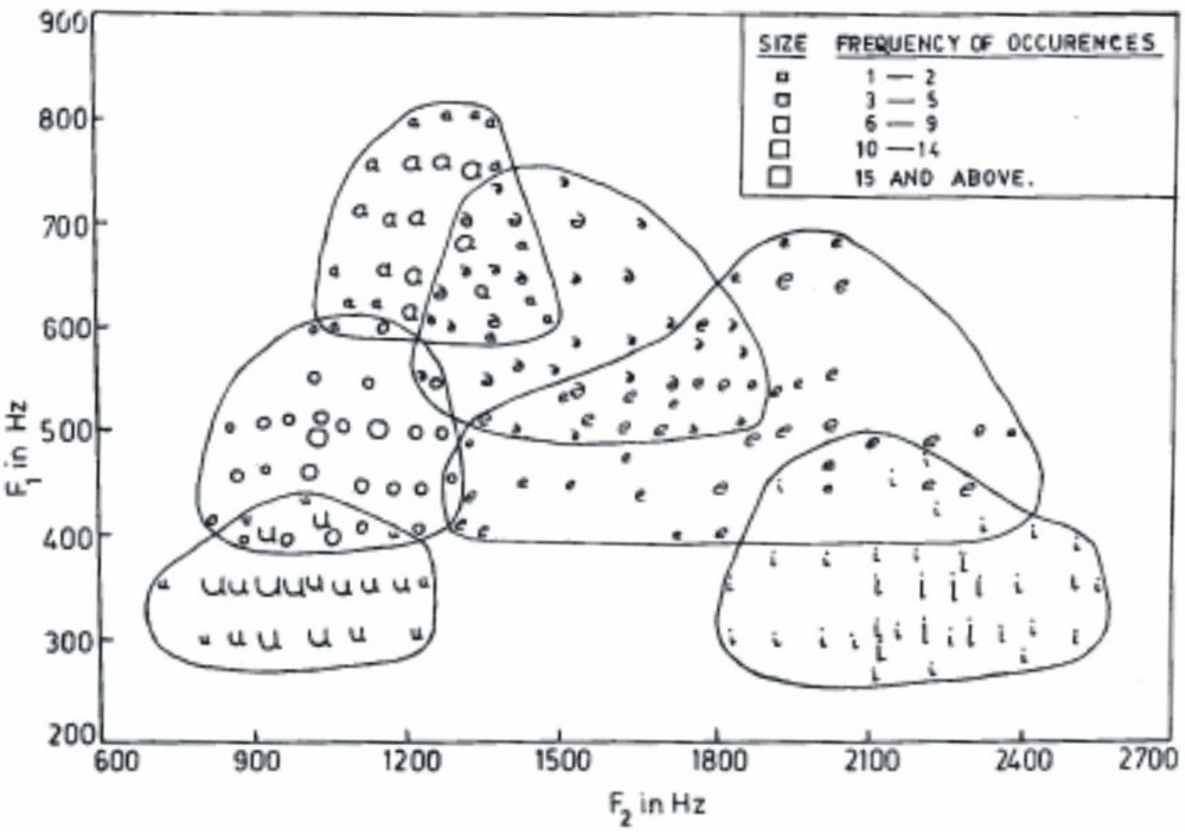

Figura 5.2: Os dados *Vogal*.

Na Tabela 5.12, são fornecidos os intervalos dos valores das medidas dos 3 atributos sobre as 6 classes, bem como a quantidade de exemplos em cada classe.

Tabela 5.12: Intervalos das medidas das vogais.

Classe	F_1	F_2	F_3	Qtde
∂	[400, 750]	[1200, 2000]	[1900, 2700]	72
a	[550, 800]	[1000, 1600]	[2100, 2600]	89
i	[250, 550]	[1500, 2550]	[2300, 3200]	172
u	[300, 500]	[700, 1400]	[1800, 2800]	151
e	[350, 900]	[1200, 2400]	[2000, 3100]	207
o	[400, 600]	[800, 1900]	[2100, 3000]	180

A classificação e a geração de regras a partir do conjunto de dados *Vogal* foi explorada em [MP94] e a sua escolha foi motivada visando uma comparação com os resultados obtidos pela RNDD.

5.3.1 Classificação

Abaixo, na Tabela 5.13 são apresentadas saídas do modelo P de Mitra e Pal [MP94] em resposta a apresentações de exemplos do conjunto *Vogal*. Este modelo utiliza como t-norma e t-conorma, respectivamente, o produto e a soma probabilística mostrados na Seção 2.3.1. As colunas F_1, F_2 e F_3 trazem os valores dos atributos de cada exemplo apresentado à rede para a geração de regras.

A marcação *indis* (indisponível), em um determinado valor de atributo, denota que seus respectivos graus de pertinência aos conjuntos *baixo*, *médio* e *alto* serão definitivamente iguais ao valor mais ambíguo do intervalo considerado: 0.5, para o intervalo unitário (usado nos modelos de [MP94]) e 0.0, para o intervalo bipolar (usado na RNDD). A marcação $falt$ (faltante) denota a mesma coisa, mas em caráter temporário: os valores podem ser mudados subseqüentemente, bastando que a rede não consiga definir conclusões significativas para todos os neurônios da camada de saída e que pergunte ao usuário o valor daquele atributo. Esta funcionalidade de consulta[8] está presente nos modelos utilizados em [MP94], mas não foi implementada na RNDD.

Na coluna Classe j, é mostrada a classe correspondente ao neurônio j com a saída máxima e, na coluna y_j^H, sua saída. Na coluna $cert_j^H$, é mostrado o valor da medida de certeza apresentada na Equação 4.7. Na coluna Classe i, é mostrada a classe correspondente ao neurônio com a segunda maior saída significativa e que possua $cert_i^H > 0.1$.

Tabela 5.13: Resposta do modelo P para o conjunto *Vogal*, extraído de [MP94]

	Atributos de Entrada			Saída Máxima		2ª Maior Saída Sig.		Certeza
#	F_1	F_2	F_3	Classe j	y_j^H	Classe i	y_i^H	$cert_j^H$
1	700	1000	2600	a	0.96	-	-	0.99
2	400	800	*indis*	o	0.27	-	-	0.89
3	400	*indis*	*indis*	e	0.17	i	0.17	0.40
4	700	1300	*indis*	a	0.73	-	-	0.98
5	700	1000	*indis*	a	0.84	-	-	1.00
6	450	2400	*indis*	i	0.38	e	0.20	0.65
7	900	1400	*indis*	a	0.44	-	-	1.00
8	600	1200	*indis*	a	0.40	o	0.30	0.57

Na Tabela 5.14, são mostradas as saídas da RNDD em resposta à apresentação de exemplos semelhantes aos utilizados para a construção da Tabela 5.13. Como pode ser observado, as linhas de 1 a 8 possuem valores de atributos próximos aos utilizados na construção da Tabela 5.13. Pode-se notar que, apesar dos valores da medida de certeza serem inferiores àqueles produzidos pelo modelo P, as decisões da RNDD mantiveram-se coerentes com os da tabela anterior e razoáveis quando analisados quanto aos intervalos da Tabela 5.12 e da Figura 5.2.

[8]Do Inglês, *querying*.

As baixas medidas de certeza obtidas pela RNDD em resposta aos padrões utilizados no modelo P deram-se por conta da particular posição da fronteira de decisão da rede que não privilegiou tantos esses exemplos em particular. No entanto, as linhas 9 e 10 foram inseridas com o intuito de exemplificar entradas que, quando submetidas à RNDD, produzem um alto grau de certeza.

Tabela 5.14: Resposta da RNDD para o conjunto *Vogal*

	Atributos de Entrada			Maior Saída		2ª Maior Saída Sig.		Certeza
#	F_1	F_2	F_3	Classe j	y_j^H	Classe i	y_i^H	$cert_j^H$
1	700	1000	2600	a	-0.32	-	-	0.62
2	400	800	2700	u	0.01	-	-	0.40
3	400	*indis*	*indis*	e	-0.02	-	-	0.34
4	700	1300	*indis*	a	-0.03	e	-0.69	0.48
5	700	1000	*indis*	a	-0.04	-	-	0.68
6	450	2400	*indis*	e	0.43	i	-0.13	0.61
7	900	1400	*indis*	a	-0.36	-	-	0.42
8	600	1200	*indis*	a	-0.21	o	-0.30	0.34
9	300	900	2400	u	-0.31	o	-0.90	0.84
10	700	2300	3100	e	0.34	-	-	0.94

Para os testes com o *Vogal*, utilizou-se a metodologia *Holdout* com $perc = 10$ (*Holdout* $10\% - 90\%$), a mesma usada em [MP94].

A rede foi treinada com os seguintes parâmetros: $\eta = 0.01$ (fixo), $\alpha = 0.01$ (fixo), $maxepoca = 4500$, $n_2 = 17$ e $Sobrep = 1.0$. Os últimos pesos do treino são fixados na rede e os exemplos de teste são submetidos e pré-processados pelas mesmas funções de pertinência. Os demais parâmetros utilizados no treino também são mantidos para o cálculo da saída da rede.

Na Tabela 5.15, é mostrada a matriz de confusão para o teste. A taxa de acerto obtida foi de 80.3% e o Kappa calculado foi de 0.771.

Tabela 5.15: Matriz de confusão - RNDD (*Holdout* $10\% - 90\%$ *Vogal*).

	Classificador					
Referência	∂	a	i	u	e	o
∂	0.004	0.032	0.000	0.000	0.027	0.020
a	0.001	0.097	0.000	0.000	0.001	0.004
i	0.000	0.000	0.174	0.001	0.022	0.000
u	0.000	0.000	0.000	0.158	0.000	0.015
e	0.001	0.000	0.023	0.003	0.182	0.029
o	0.000	0.005	0.000	0.011	0.001	0.188

Na Tabela 5.16, são mostradas as taxas de acerto de classificação dos três modelos de RND utilizados em [MP94] e do modelo RNDD. O modelo *KDL* utiliza os operadores *min* e *max*, com o operador de implicação Kleene-Dienes-Lukasiewicz ($||X \subset Y|| = 1 - X + XY$). Na linha n_2 da tabela, são especificados os números de neurônios na camada intermediária de cada modelo. A coluna Modelo especifica o modelo de RND utilizado no teste. As linhas especificam as taxas de acerto obtidas pelos diferentes modelos em cada classe do conjunto *Vogal* e a taxa de acerto geral, além do EQM.

Tabela 5.16: Desempenho de classificação - *Vogal*).

Modelo	*P*		*KDL*	*Fuzzy O*		RNDD	
n_2	20	22	20	20	22	15	17
∂(%)	31.2	24.6	9.3	44.6	69.8	4.6	4.6
a(%)	76.5	93.8	97.5	65.4	72.8	95.1	93.8
i(%)	90.2	92.2	96.7	79.2	81.8	90.3	88.4
u(%)	80.0	93.5	85.9	88.3	85.9	91.2	91.2
e(%)	75.1	62.3	15.8	75.0	75.0	73.3	76.5
o(%)	86.5	77.9	3.0	85.7	87.2	91.4	91.4
Geral(%)	77.8	77.1	48.8	76.8	80.1	80.0	80.3
EQM	0.042	0.047	0.072	0.046	0.04	0.84	0.82

Apesar de a taxa de acerto para os exemplos da classe ∂ ter sido pobre e o EQM ser relativamente alto quando comparado com os demais modelos, os resultados de classificação geral obtidos pela RNDD foram ligeiramente superiores àqueles reportados em [MP94]. Importante notar, ainda, que os modelos usados por Mitra e Pal possuíam 20 e 22 neurônios na camada intermediária. Os resultados mostrados na Tabela 5.16 foram alcançados pela RNDD com a utilização de 15 e de 17 neurônios na camada intermediária. Desta forma, pode-se concluir que a RNDD conseguiu resultados superiores com redes mais simples, i. é., com menos parâmetros internos (pesos).

Outro ponto a considerar é que os resultados apresentados na Tabela 5.16 são para os modelos de RND que não utilizam a difusificação nas saídas desejadas. Em [MP94] também foram apresentados resultados de classificação geral do modelo P com $n_2 = 20$ e utilizando a referida difusificação. Estes resultados variaram de 72.2% a 86.1%. Esta difusificação poderá ser incorporada à RNDD para permitir novas comparações. Esta é, inclusive, uma sugestão para aperfeiçoamentos futuros.

5.3.2 Geração de regras

Segundo Mitra e Pal [MP94], os valores dos atributos de entrada influenciam na geração de regras a partir dos conjuntos de pesos sinápticos da rede treinada. Isso ajuda a extrair regras relevantes para a região do espaço de características que fica em torno do exemplo de dados utilizado.

Em [MP94], foram utilizados diferentes qualificadores, de acordo com o valor da medida de certeza da Equação 4.7, para os conseqüentes das regras, conforme mostrado na Tabela 5.17.

Tabela 5.17: Qualificadores dos conseqüentes, utilizados em [MP94]

Qualificador	Intervalo
muito provavelmente	$0.8 \leq cert_j^H \leq 1.0$
provavelmente	$0.6 \leq cert_j^H < 0.8$
mom provavelmente	$0.4 \leq cert_j^H < 0.6$
não-improvavelmente	$0.1 \leq cert_j^H < 0.4$
impossível de reconhecer	$cert_j^H < 0.1$

Nas Tabelas 5.18 e 5.19, são mostradas as regras difusas geradas pelos modelos *P* e *Fuzzy O* [MP94]. Neste último modelo, somas ponderadas e funções sigmóides são empregadas no lugar das t-normas. Observa-se a presença de mais de um conseqüente nessas regras. Isto se deu porque as cláusulas do antecedente foram geradas para todos os neurônios de saída possuindo $cert_j^H > 0.1$. Isto também foi feito para a RNDD. As regras 2a, 2b e 4a, 4b, apesar de serem geradas a partir do mesmo exemplo de entrada, foram colocadas em linhas distintas para ressaltar que, além dos conseqüentes de ambas possuírem medidas de certeza similares, seus antecedentes são distintos.

Tabela 5.18: Regras geradas pelo modelo P.

#	F_1	F_2	F_3	Regra
1	900	1400	*indis*	Se (F_2 é *muito médio*) *ou* (F_1 é *muito alto*) Então *classe* é *muito provavelmente a*
2	700	1300	*indis*	Se (F_2 é *muito médio*) *e* (F_1 é *muito médio*) *ou* (F_1 é *mom alto*)Então *classe* é *muito provavelmente a*
3	700	1000	2600	Se (F_3 é *muito médio*) *e* (F_1 é *muito médio*) *ou* (F_1 é *mom alto*) *ou* (F_2 é *muito baixo*) Então *classe* é *muito provavelmente a*
4	700	*indis*	*falt*	Se (F_1 é *muito médio*) *ou* (F_1 é *mom alto*) Então *classe* é *provavelmente a*, mas *não-improvavelmente e*
5	600	1200	*indis*	Se (F_1 é *muito médio*) *e* (F_2 é *muito médio*) *ou* (F_2 é *baixo*) Então *classe* é *mom provavelmente a ou o*
6	450	2400	*indis*	Se (F_2 é *muito alto*) *e* (F_1 é *mom baixo*) *ou* (F_1 é *muito médio*) Então *classe* é *provavelmente i*, mas *não-improvavelmente e*
7	400	800	*indis*	Se (F_1 é *muito baixo*) *e* (F_2 é *muito baixo*) Então *classe* é *muito provavelmente o*, mas *não-improvavelmente u*
8	400	*indis*	*indis*	Se (F_1 é *muito baixo*) Então *classe* é *mom provavelmente e ou i*, mas *não-improvavelmente* classe *o*
9	300	900	*indis*	Se (F_1 é *muito baixo*) *ou* (F_2 é *muito baixo*) Então *classe* é *muito provavelmente u*
10	500-600	1600	*falt*	Se (F_2 é *muito médio*) *ou* (F_1 é *muito médio*) *ou* (F_1 é *mom baixo*) Então *classe* é *provavelmente e*, mas *não-improvavelmente o*

Tabela 5.19: Regras geradas pelo modelo *Fuzzy O*.

#	F_1	F_2	F_3	Regra
1	700	2300	$falt$	Se (F_2 é *muito alto*) *e* (F_1 é *muito médio*) Então *classe* é *muito provavelmente e*
2	700	1300	*indis*	Se (F_2 é *muito médio*) *e* (F_1 é *mom alto*) Então *classe* é *provavelmente a*
3	700	1000	2600	Se (F_3 é *muito médio*) *e* (F_1 é *mom alto*) Então *classe* é *muito provavelmente a*
4a	700	*indis*	*indis*	Se (F_1 é *muito médio*) Então *classe* é *não-improvavelmente* ∂
4b	700	*indis*	*indis*	Se (F_1 é *mom alto*) Então *classe* é *não-improvavelmente a*
5	600	1200	$falt$	Se (F_1 é *muito médio*) Então *classe* é *muito provavelmente o*
6	400	800	$falt$	Se (F_2 é *muito baixo*) *e* (F_1 é *muito baixo*) Então *classe* é *provavelmente o*, mas *não-improvavelmente u*
7	400	*indis*	*indis*	Se (F_1 é *muito baixo*) Então *classe* é *não-improvavelmente e ou o ou i ou u ou* ∂
8	300	900	*indis*	Se (F_1 é *muito baixo*) *e* (F_2 é *muito baixo*) Então *classe* é *muito provavelmente u*, mas *não-improvavelmente o*
9a	250	1550	*indis*	Se (F_1 é *muito baixo*) *e* (F_2 é *mom baixo*) Então *classe* é *mom provavelmente u*
9b	250	1550	*indis*	Se (F_2 é *muito médio*) *e* (F_1 é *muito baixo*) Então *classe* é *mom provavelmente e*
10	*indis*	1000	*indis*	Se (F_2 é *muito baixo*) Então *classe* é *mom provavelmente o*, mas *não-improvavelmente u ou a*

Na Tabela 5.20, são mostradas as regras obtidas pela RNDD. Observe que alguns exemplos de entrada são ligeiramente diferentes daqueles utilizados nas Tabelas 5.18 e 5.19. Isto ocorreu, por dois motivos: a RNDD não conseguiu gerar regra para o padrão considerado ou não se soube ao certo o valor definitivo para os valores $falt$.

Tabela 5.20: Regras geradas pela RNDD.

#	F_1	F_2	F_3	Regra
1	900	1400	*indis*	Se (F_2 é *muito médio*) Então *classe* é *mom provavelmente a*
2	700	2300	3100	Se (F_1 é *alto*) *e* (F_3 é *muito alto*) Então *classe* é *muito provavelmente e*
3	700	1300	*indis*	Se (F_2 é *muito médio*) *e* (F_1 é *muito médio*) *ou* (F_1 é *alto*) *e* (F_2 é *mom baixo*) Então *classe* é *mom provavelmente a*, mas *não-improvavelmente e*
4	700	1100	2200	Se (F_1 é *muito médio*) *e* (F_3 é *muito médio*) Então *classe* é *provavelmente a*
5	700	1000	2600	Se (F_3 é *muito médio ou* F_3 é *mom alto*) *e* (F_1 é *muito médio*) Então *classe* é *provavelmente a*
6	700	1000	*indis*	Se (F_1 é *muito médio*) Então *classe* é *provavelmente a*
7	700	*indis*	3100	Se (F_1 é *alto*) *e* (F_3 é *muito alto*) *ou* (F_3 é *muito alto*) *e* (F_1 é *muito médio*) Então *classe* é *mom provavelmente e*, mas *não-improvavelmente a*
8a	700	*indis*	*indis*	Se (F_1 é *alto*) Então *classe* é *não-improvavelmente e*
8b	700	*indis*	*indis*	Se (F_1 é *muito médio*) Então *classe* é *não-improvavelmente a*
9a	600	1600	*indis*	Se (F_2 é *muito médio*) *e* (F_1 é *mom alto*) *e* (F_2 é *mom baixo*) Então *classe* é *não-improvavelmente e ou o*
9b	600	1600	*indis*	Se (F_2 é *muito médio*) *e* (F_1 é *muito médio*) Então *classe* é *não-improvavelmente a*
10a	600	1200	*indis*	Se (F_1 é *muito médio*) *e* (F_2 é *muito médio*) Então *classe* é *não-improvavelmente a*
10b	600	1200	*indis*	Se (F_1 é *mom alto*) *e* (F_2 é *muito médio*) Então *classe* é *não-improvavelmente o*
10c	600	1200	*indis*	Se (F_1 é *muito médio*) *e* (F_1 é *mom alto*) *e* (F_2 é *muito médio*) Então *classe* é *não-improvavelmente u*
11	450	2400	*indis*	Se (F_1 é *baixo*) *ou* (F_1 é *muito médio*) Então classe é *provavelmente e*, mas *não-improvavelmente i*
12	400	800	2700	Se (F_3 é *mom alto*) *ou* (F_3 é *muito médio*) Então *classe* é *mom provavelmente o*
13	400	*indis*	*indis*	Se (F_1 é *muito baixo*) Então *classe* é *não-improvavelmente e*
14	300	900	2400	Se (F_3 é *mom baixo*) *ou* (F_3 é *muito médio*) Então *classe* é *muito provavelmente u*, mas *não-improvavelmente o*
15a	250	1550	*indis*	Se (F_2 é *muito médio*) Então *classe* é *mom provavelmente u*
15b	250	1550	*indis*	Se (F_2 é *muito médio*) *e* (F_2 é *mom baixo*) Então *classe* é *mom provavelmente i*

5.3.3 Comparação de resultados

Como pôde ser visto na Tabela 5.16, o desempenho de classificação da RNDD mostrou-se comparável àqueles obtidos pelos pelos modelos *P*, *KDL* e *Fuzzy O*. Importante ressaltar que este desempenho de classificação da RNDD foi obtido utilizando uma rede com menos neurônios na camada intermediária.

Quanto à geração de regras, segundo [MP94], os modelos de RND com t-normas quando comparadas ao modelo *Fuzzy O* apresentam uma vantagem. Esses modelos permitem a geração de regras mais apropriadas, expressas como disjunções de cláusulas conjuntivas. Isto é devido a estrutura *e/ou* do modelo lógico. Por outro lado, os neurônios do modelo *Fuzzy O* com funções sigmoidais aproximam as funções *e/ou* em casos especiais dependendo do aprendizado da rede. Logo, as cláusulas das regras na forma *e/ou* não podem ser geradas com uma eficiência comparável.

As regras geradas pela RNDD mostraram-se coerentes com as regras geradas pelos modelos *P* e *Fuzzy O*. Na Tabela 5.21, é feito um comparativo das decisões de classificação desses três modelos, mostradas nas Tabelas 5.20 (RNDD), 5.18 (*P*) e 5.19 (*Fuzzy O*).

Tabela 5.21: Razoabilidade das regras geradas pela RNDD.

#	RNDD	*P*	*Fuzzy O*	Razoável
1	a	$= P_{(1)}$	-	sim
3	a, e	$= P_{(2)}$	$= O_{(2)}$	sim
5	a	$= P_{(3)}$	$= O_{(3)}$	sim
11	e, i	$\neq P_{(6)}$	-	não
13	e	$= P_{(8)}$	$= O_{(7)}$	sim
10a	a	$= P_{(5)}$	-	sim
10b	o	$= P_{(5)}$	-	sim
10c	u	$\neq P_{(5)}$	-	não
15a	u	-	$= O_{(9a)}$	sim
15b	i	-	$\neq O_{(9b)}$	não
8a	e	-	$\neq O_{(4a)}$	não
8b	a	-	$= O_{(4b)}$	sim
2	e	-	-	-
4	a	-	-	-
6	a	-	-	-
7	e, a	-	-	-
12	o	-	-	-
14	u, o	-	-	-
9a	e, o	-	-	-
9b	a	-	-	-

Por exemplo, na primeira linha da tabela, a decisão de classificação da rede RNDD em resposta ao exemplo de entrada 1 foi igual àquela produzida pelo modelo *P*. O traço na coluna *Fuzzy O* indica que não houve uma entrada comparável na Tabela 5.19. A coluna Razoável mostra se a decisão tomada pela RNDD foi concordante ou não com os dois outros modelos.

Como pode ser observado, na maioria das entradas comparáveis, houve concordância com os conseqüentes produzidos pelos modelos *P* e *Fuzzy O*. As decisões da RNDD também foram razoáveis em todas as regras, considerando os intervalos dos valores de atributo do conjunto Vogal, mostrados na Tabela 5.12 e a projeção da Figura 5.2.

5.4 Conjunto de Dados *Alea*

Foram criados outros três conjuntos de dados aleatórios denominados Alea1, Alea2 e Alea3. Cada conjunto composto por 900 linhas com 5 colunas (4 atributos + 1 classe). As linhas são distribuídas equitativamente em três classes. Cada valor de atributo foi obtido a partir da geração de números aleatórios com distribuição normal. Os conjuntos possuem todos a mesma variância. As médias foram alteradas para prover diferentes graus de sobreposição entre as classes. A Figura 5.3 traz uma projeção tridimensional dos atributos dos três conjuntos.

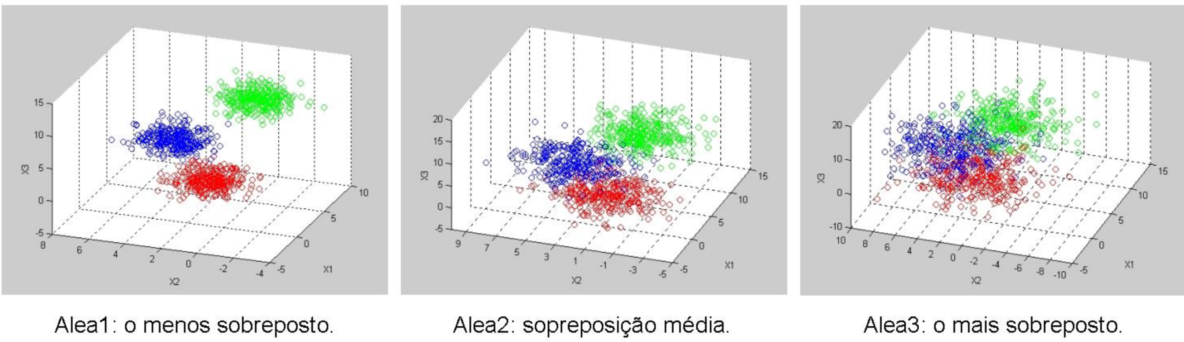

Figura 5.3: Sobreposição dos conjuntos Alea1, Alea2 e Alea3.

Estes conjuntos de dados serão utilizados para se avaliar o desempenho de classificação da RNDD conforme se aumenta a sobreposição entre as classes. Um pré-processamento semelhante ao mostrado na Seção 3.3.1 foi feito também para os conjuntos *Alea*.

Foram feitos 12 testes *Holdout* $70\% - 30\%$ sobre os conjuntos *Alea* com os seguintes parâmetros de treino: $\eta = 0.01$ (fixo), $\alpha = 0.1$ (fixo), $maxepoca = 500$, $n_2 = 8$. O parâmetro *Sobrep* das funções de pertinência foi variável a cada teste.

As Figuras 5.4 e 5.5 mostram a variação do índice Kappa e da taxa de acerto nos testes com os três conjuntos e variando o parâmetro *Sobrep* de 0.1 a 2.0.

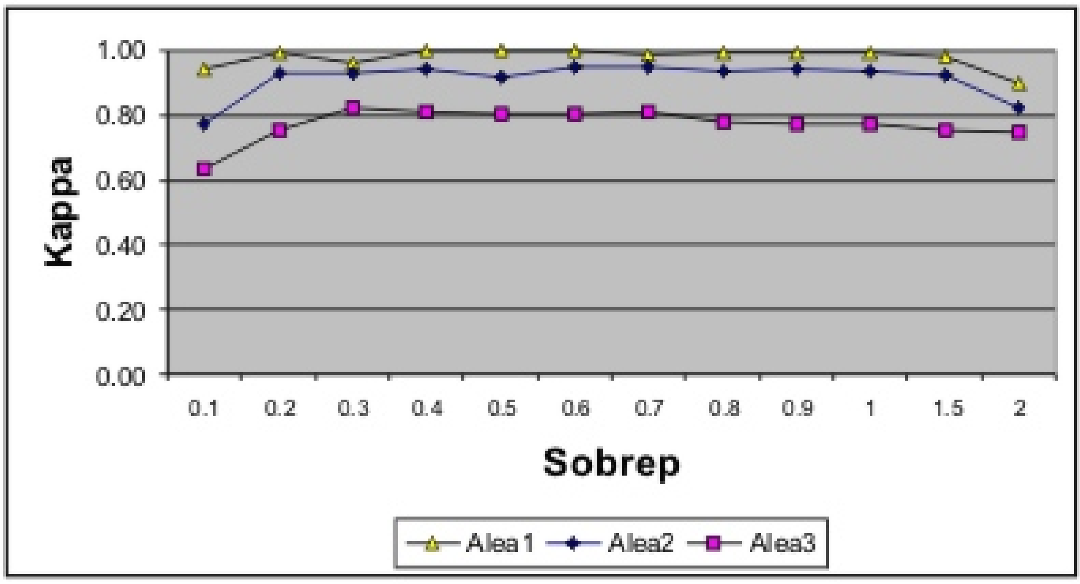

Figura 5.4: Variação de Kappa sobre o parâmetro *Sobrep*.

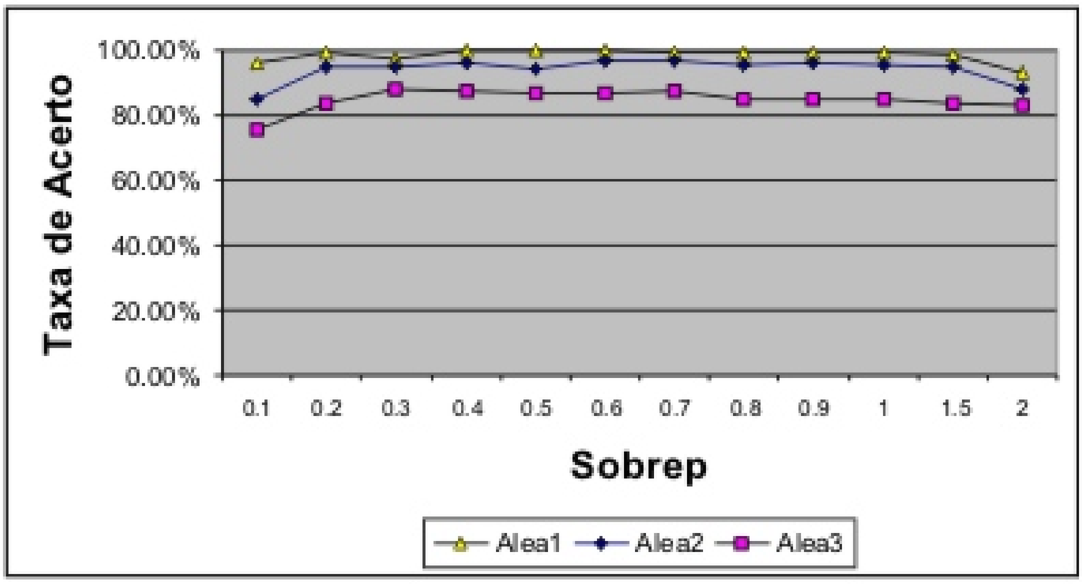

Figura 5.5: Variação da taxa de acerto sobre o parâmetro *Sobrep*.

As figuras mostram que, mesmo no conjunto AleaV6, em que a sobreposição entre as três classes é maior, o índice Kappa e a taxa de acerto mantiveram-se altos. O parâmetro *Sobrep* não teve influência significativa nos maiores índices de precisão alcançados pela rede nos testes com os três conjuntos *Alea*, contudo, não foi feito um controle rigoroso da variação deste parâmetro. Pode-se notar que um valor de *Sobrep* muito pequeno (0.1) ou muito grande (2.0) pode prejudicar o desempenho de classificação da rede, dada a curvatura dos gráficos apresentados nas Figuras 5.4 e 5.5.

Os resultados apresentados neste capítulo mostraram-se promissores. No entanto, é importante destacar a necessidade de mais comprovações empíricas e a demanda por investigações subseqüentes para que a metodologia proposta neste trabalho seja efetivamente utilizada como uma ferramenta de aquisição de conhecimento a partir de exemplos de dados.

Capítulo 6

Conclusão

Os esforços de pesquisa nas áreas de conjuntos difusos, redes neurais e sistemas híbridos compostos por esses dois paradigmas têm contribuído de forma significativa para o desenvolvimento de tecnologias voltadas para a mineração de dados. Uma dessas contribuições, tratada nesta dissertação, refere-se à utilização de redes neurais difusas para extração de conhecimento a partir de grandes bases de dados.

O objetivo principal deste projeto foi implementar uma rede neural difusa utilizando as t-normas e t-conormas diferenciáveis e interativas definidas em [ZAN97] para realizar as tarefas de classificação de padrões e de geração de regras. Considerando este objetivo, foram apresentados os conceitos básicos relativos à teoria dos conjuntos difusos e as t-normas que foram utilizadas na implementação deste trabalho. O uso de t-normas diferenciáveis permitiu a derivação da função de erro, além de evitar o uso de operadores de implicação.

A característica da interatividade da rede não foi avaliada. Para esta avaliação, seria necessário trabalhar com o conceito de granularidade. Introduzida por Zadeh, a granularidade pode ser vista metaforicamente como o processo de filtrar dados através de peneiras com orifícios de diferentes tamanhos. Conjuntos difusos são grânulos de informação que possuem um número de elementos e um nome associado. Tal avaliação é sugerida como um dos aperfeiçoamentos futuros.

Foram revisados alguns conceitos acerca das RNAs, sendo introduzidos alguns modelos de RNDs, particularmente a rede reural difusa diferenciável, objeto de estudo deste trabalho. A principal dificuldade em se entender os conceitos representados nessas redes está no fato de o conhecimento quase sempre estar em uma maneira não-inteligível, armazenado na forma de uma grande quantidade de pesos de conexões.

Descrito sob a forma de regras, o conhecimento extraído pode ser analisado de diferentes maneiras pelo usuário. Alguns algoritmos de geração de regras a partir de redes treinadas foram apresentados, entre eles, o algoritmo *backtracking*. Geralmente, quando uma RNA é orientada para tarefas descritivas de descoberta do conhecimento, o volume de regras geradas pode ser muito grande, o que dificulta a análise do poder descritivo dessas regras por parte do usuário.

Algumas medidas de avaliação de regras foram mostradas com o objetivo de saber quais dessas regras são melhor sustentadas pelos dados. Além disso, foram brevemente apresentados dois métodos de pós-processamento de regras que têm como objetivo reduzir o número de regras a um pequeno conjunto, mantendo ao máximo a qualidade das regras originais.

No intuito de obter um método automático de aquisição de conhecimento a partir de exemplos de um domínio de dados qualquer, foi implementada a RNDD, juntamente com o módulo de geração de regras. Uma heurística consistindo de diminuições sucessivas na taxa de aprendizado e momento, bem como a apresentação aleatória dos exemplos de treino também foram codificadas.

Testes de classificação de dados e geração de regras foram realizados utilizando várias metodologias de teste. Os resultados de classificação obtidos com a aplicação da RNDD sobre o conjunto *Iris*, *Vogal* e sobre os conjuntos *Alea* mostraram-se promissores. Na classificação do *Iris*, a maior taxa de acerto foi de 97.77% e o maior Kappa foi de 0.9667, obtidos no teste *Holdout* $70\% - 30\%$. Visando uma comparação com os resultados de vários classificadores reportados por Woods [WKB97], foi realizado um teste de mesma metodologia sobre o conjunto *Iris_CR*. A taxa de acerto média alcançada foi de 96.00%, inferior somente ao método de classificação Bayes Linear. Sobre o conjunto de dados *Vogal*, a RNDD teve um desempenho de classificação superior quando comparado aos resultados dos modelos de RND reportados em [MP94], utilizando menos neurônios na camada intermediária.

Nos testes de classificação sobre os conjuntos aleatoriamente gerados, o índice Kappa e a taxa de acerto mantiveram-se altos, mesmo no conjunto em que as classes estão mais sobrepostas. Foram atingidos índices máximos de precisão na classificação do conjunto com menor sobreposição. No conjunto com maior sobreposição, a taxa de acerto alcançada foi de 88.15% e o Kappa de 0.8220. Esses resultados parecem tornar a RNDD apropriada para problemas de classificação de dados que apresentam sobreposição entre as classes.

O parâmetro *Sobrep*, que controla a extensão da sobreposição das funções de pertinência, influenciou significativamente os resultados de classificação produzidos na aplicação sobre o *Iris*. No entanto, não teve influência significativa nos maiores índices de precisão alcançados pela rede nos testes com os três conjuntos *Alea* e no conjunto *Vogal*. Os melhores resultados de classificação destes conjuntos foram alcançados utilizando $Sobrep = 1.0$, contudo não foi feito um controle rigoroso da variação deste parâmetro. Em [HAY99], na seção 6.9 *Experimento Computacional*, há uma metodologia para a padronização da variação de parâmetros nos testes.

O algoritmo *backtracking* foi aplicado à RNDD treinada com os dados *Vogal*. Foram geradas regras com exemplos de dados semelhantes aos utilizados em [MP94]. A RNDD permite a geração de regras de uma forma mais apropriada quando comparada a modelos sem operadores lógicos como o *Fuzzy O* de Mitra e Pal. Essas regras são expressas como disjunções de cláusulas conjuntivas. Isto é devido à estrutura *e/ou* da RNDD e das demais redes lógicas.

Foram ainda geradas 75 regras difusas dos dados *Iris* a partir dos pesos da rede treinada e da apresentação dos respectivos 75 exemplos de treino. Ainda que considerando o critério de interpretação da saída da rede como sendo o do "vencedor-leva-tudo", houve uma grande quantidade de regras geradas: 75, uma para cada padrão de treino utilizado. Foram calculadas algumas medidas de avaliação das regras geradas pelo *Iris*, para permitir a seleção daquelas mais promissoras.

A avaliação empregada sobre os resultados de geração de regras utilizando medidas de avaliação pode ser vista como uma análise objetiva do conhecimento extraído. Nada impede que, na análise subjetiva do conhecimento, feita por um especialista de domínio, esta análise objetiva possa ser usada. Assim, o especialista pode focalizar sua atenção sobre as regras que mais se destacam evitando que se tenha de examinar uma a uma cada regra do conjunto de regras.

Contribuições

A bibliografia existente foi examinada, mostrando um contexto geral das redes neurais difusas com geração de regras. Além disso, foram apresentadas medidas de avaliação e métodos de pós-processamento de regras.

Uma rede neural difusa com t-normas diferenciáveis foi definida e implementada. Essa rede mostrou-se capaz de classificar bem instâncias de uma base de dados e de gerar regras na forma Se - Então que explicassem as decisões de classificação tomadas pela rede. O fato de a implementação da rede possuir a funcionalidade de salvar as matrizes de pesos possibilita que seu estado seja preservado entre sucessivas execuções. Desta forma, tornando possível a execução do módulo de geração de regras sem a necessidade de treinar novamente a rede.

A metodologia aqui tratada poderá ser utilizada em outras bases de dados representando problemas reais, por exemplo, os de relevância financeira, tais como: detecção de fraudes em cartões de crédito ou em unidades consumidoras de energia. Para tanto, um conjunto de instâncias de dados de clientes fraudadores e não-fraudadores seria submetida à RNDD. Depois de treinada e verificada a qualidade de sua generalização, regras representando o conhecimento adquirido seriam geradas. Caso o número de regras seja muito grande dificultando o seu entendimento, estas poderiam ser submetidas a um dos dois métodos de pós-processamento apresentados.

Outra contribuição deste trabalho, foi a publicação do artigo *A Fuzzy Neural Network with Differentiable T-Norms: Classification and Rule Generation*, de Oliveira e Zanusso [dOZ05b] aceito no *International Conference of Artificial Intelligence* ICAI-2005.

Além disso, a presente dissertação servirá de base para trabalhos subseqüentes que se ocuparão de avaliar outros aspectos da metodologia proposta.

Limitações do trabalho

Na implementação da RNDD e do módulo de geração de regras foram utilizadas linguagens de programação distintas. A RNDD foi implementada em linguagem C, devido à sua característica de gerar um código executável rápido. Já para a implementação do módulo de geração de regras, foi utilizada a linguagem de scripts *Matlab*, porque esta facilita a manipulação, ordenação e alocação de matrizes.

Esta heterogeneidade entre as linguagens utilizadas nas implementações dos módulos deu-se por conta de uma escolha pessoal, particular, na tentativa de explorar as melhores características de cada linguagem. No entanto, a implementação utilizando uma mesma linguagem de programação em todos os módulos favoreceria a criação de uma interface amigável, possibilitando a utilização efetiva da RNDD como uma ferramenta de aquisição de conhecimento a partir de dados brutos. Além do mais, poder-se-ia ter toda essa funcionalidade disponibilizada em um componente, acoplável a um sistema de banco de dados qualquer. Esta é, inclusive, uma sugestão para aperfeiçoamentos futuros.

Idealmente, diversos conjuntos de dados devem ser utilizados e diversos testes devem ser feitos para aferir a qualidade da hipótese produzida pelo classificador. Nesse sentido, em [RN04], seção *Avaliação de desempenho do algoritmo de aprendizagem*, é sugerida a seguinte metodologia:

1. Coletar um grande conjunto de amostras de um domínio qualquer;

2. Dividi-lo em dois conjuntos distintos;

3. Treinar o classificador com um dos conjuntos para gerar a hipótese;

4. Medir o percentual de exemplos do outro conjunto corretamente classificados pela hipótese gerada;

5. Repetir todos os passos acima para diferentes tamanhos de conjuntos aleatoriamente selecionados.

O resultado desta metodologia é uma série de dados que podem ser processados para dar a qualidade média da classificação como uma função do tamanho do conjunto usado no treinamento do classificador. Isto pode ser colocado em um gráfico denominado de curva de aprendizagem. Normalmente, quando o tamanho do conjunto de treino cresce, a qualidade da classificação aumenta. Mais testes, com diferentes tamanhos e diferentes conjuntos de dados devem ser feitos para a RNDD para avaliar sua qualidade de classificação.

Uma outra limitação diz respeito ao tamanho das matrizes que devem ser mantidas simultaneamente na memória principal. A manipulação de conjuntos de dados maiores que o *Vogal* pela RNDD só será possível mediante a utilização de arquivos em disco para simular essas matrizes.

Aperfeiçoamentos futuros

Diversas questões podem ser abordadas em trabalhos futuros e, dentre elas, pode-se enumerar:

1. Aplicação da RNDD a outros conjuntos de dados e posterior avaliação quanto ao desempenho de classificação e geração de regras;
2. Avaliação do tempo de treinamento da RNDD, expresso em número de épocas, mediante comparação com outros modelos de RND;
3. Implementação da RNDD, dos módulos de geração e avaliação de regras em uma única linguagem, além do desenvolvimento de uma interface amigável. Dessa forma, possibilitando a utilização efetiva da RNDD como uma ferramenta de aquisição de conhecimento a partir de dados brutos;
4. Disponibilização das principais funções implementadas em um componente, acoplável a um sistema de banco de dados qualquer;
5. Implementação da RNDD em linguagem Java, sob o mesmo padrão da ferramenta de mineração de dados *Weka*, para permitir a automação de testes e comparação com os demais classificadores já implementados;
6. Introdução de outras técnicas como a aceleração do treinamento de redes neurais por algoritmos genéticos [REZ03] e a utilização de *Rough Sets* [XXL05], com intuito de conferir maior robustez ao sistema proposto.
7. Utilização de outras funções de pertinência como as triangulares e trapezoidais, muito utilizadas na literatura e com bons resultados;
8. Utilização de difusificação nas saídas desejadas da rede para permitir uma comparação com os outros resultados de classificação reportados por [MP94];
9. Emprego de outros métodos para geração de regras tais como o EN;
10. Submissão das regras geradas a uma máquina de inferência visando determinar até que ponto podem figurar como uma base de conhecimento;
11. Implementação de rotina que torne possível a utilização de entradas lingüísticas na rede;
12. Substituição das matrizes de memória principal utilizadas pela rede por arquivos em disco;
13. Avaliação da característica de interatividade da rede;

Hibridizações, geralmente, devem ser consideradas como alternativas para se resolver um problema ou mesmo aumentar o desempenho ou a qualidade de uma tarefa em particular. O foco da investigação científica de mineração de dados, em particular da tarefa de classificação, não está na eficiência, mas na eficácia. Busca-se, em um primeiro momento, fazer, para depois, em um segundo momento, fazer melhor.

Apêndice: Implementação

O objetivo deste capítulo é descrever a implementação da RNDD e de seus módulos. Um diagrama simplificado da rede pode ser visto na Figura 1.1. Este diagrama será útil para entender alguns aspectos básicos do sistema. Como pode ser visto, existe um pré-processador difuso responsável pela conversão de entradas numéricas em dados difusos que serão aplicados à rede. A saída da rede será um vetor n_3-dimensional de graus de pertinência, onde n_3 é o número de classes consideradas. Observa-se, neste sistema, a inexistência do passo de desdifusificação.

A implementação da RNDD foi realizada na linguagem C. Abaixo são mostrados a lista de parâmetros de treino da rede e pseudo-códigos do arquivo-fonte principal e das rotinas que participam da truncagem de pesos. O código foi reescrito em Português estruturado, usando a mesma notação que [FBF+99], para que o leitor se familiarize com o programa. Uma descrição textual de cada função também é dada.

Parâmetro	**Comentário**
$MAXEPOCA$	// número máximo de épocas
$NTREINO$	// número de linhas da matriz treino
$NDES = NTREINO$	// número de linhas da matriz desejada
$NTESTE$	// número de linhas da matriz teste
$NDEST = NTESTE$	// número de linhas da matriz desejada de teste
$NUMATRIBUTOS$	// número de atributos observados
$N1 = (NUMATRIBUTOS * 3)$	// número de unidades na primeira camada
$N2$	// número de unidades na segunda camada
$N3$	// número de unidades na terceira camada
$SOBREP$	// sobreposição das funções de pertinência

Variável Global	**Comentário**
ETHAV	// vetor para as diminuições sucessivas da taxa de aprendizado
ALPHAV	// vetor para as diminuições sucessivas do momento
ETALP	// índice dos vetores de diminuições
INTEREP	// período para as diminuições entre as épocas
DELTAPEQUENO	// tolerância máxima do erro
TREINO	// matriz de treino
DES	// matriz desejada de treino
TESTE	// matriz de teste
DEST	// matriz desejada de teste
MINIMO	// vetor com os valores mínimos do conjunto de treino
MAXIMO	// vetor com os valores máximos do conjunto de treino
LAMBDA	// matriz dos valores *baixo*, *médio* e *alto* para LAMBDA
CE	// matriz dos valores *baixo*, *médio* e *alto* para CE
SAIDAS1	// vetor com os valores de entrada difusificados
SAIDAS2	// vetor com os valores de saída da camada intermediária *e*
SAIDAS3	// vetor com os valores da camada de saída *ou*
ERROEP	// vetor de erros para cada época
ERRO	// variável acumuladora de erro para cada padrao de treino
DELTAANTERIOR	// matriz com os valores de delta anteriores
PESOS1	// vetor de pesos para os neurônios da camada intermediária
PESOS2	// vetor de pesos para os neurônios da camada de saída
PESOS1ANTERIOR	// vetor com os valores anteriores de PESOS1
PESOS2ANTERIOR	// vetor com os valores anteriores de PESOS2
ERROSSAIDA	// vetor com os erros de cada saída na época atual
DIFERENCAS	// vetor com o somatório das diferenças (desejado - saída)
GRAD1	// vetor com os valores do GRAD1
GRAD2	// vetor com os valores do GRAD2
DELTA1	// vetor com os valores para DELTA1
DELTA2	// vetor com os valores para DELTA2
NORMADEDELTA	// truncagem dos pesos
NORMA1	// truncagem dos pesos
NORMA2	// truncagem dos pesos
TETAMIN	// truncagem dos pesos
TETA	// valor para teta

.<u>declare</u> ETHAV[]={ETHA1,ETHA2,ETHA3,ETHA4,ETHA5} <u>numérico</u>
.<u>declare</u> ALPHAV[]={ALPHA1,ALPHA2,ALPHA3,ALPHA4,ALPHA5} <u>numérico</u>
.<u>declare</u> ETALP <u>numérico</u>
.<u>declare</u> INTEREP <u>numérico</u>
.<u>declare</u> DELTAPEQUENO <u>numérico</u>
.<u>declare</u> TREINO[$NTREINO$][$NUMATRIBUTOS$] <u>numérico</u>
.<u>declare</u> DES[$NTREINO$][$N3$] <u>numérico</u>
.<u>declare</u> TESTE[$NTESTE$][$NUMATRIBUTOS$] <u>numérico</u>
.<u>declare</u> DEST[$NTESTE$][$N3$] <u>numérico</u>
.<u>declare</u> MINIMO[$NUMATRIBUTOS$] <u>numérico</u>

.declare MAXIMO[$NUMATRIBUTOS$] numérico
.declare LAMBDA[$N3$][$NUMATRIBUTOS$] numérico
.declare CE[$N3$][$NUMATRIBUTOS$] numérico
.declare SAIDAS1[$N1$] numérico
.declare SAIDAS2[$N2$] numérico
.declare SAIDAS3[$N3$] numérico
.declare ERROEP[$MAXEPOCA$] numérico
.declare ERRO[$NTREINO$] numérico
.declare DELTAANTERIOR[$N3$][$N2$] numérico
.declare PESOS1[$N1$*$N2$] numérico
.declare PESOS2[$N2$*$N3$] numérico
.declare PESOS1ANTERIOR[$N1$*$N2$] numérico
.declare PESOS2ANTERIOR[$N2$*$N3$] numérico
.declare ERROSSAIDA[$N3$] numérico
.declare DIFERENCAS[$N3$] numérico
.declare GRAD1[$N2$*$N1$] numérico
.declare GRAD2[$N3$*$N2$] numérico
.declare DELTA1[$N1$*$N2$] numérico
.declare DELTA2[$N2$*$N3$] numérico
.declare NORMADEDELTA numérico
.declare NORMA1 numérico
.declare NORMA2 numérico
.declare TETAMIN numérico
.declare TETA numérico

```
Algoritmo RNDD
.    declare N, EPOCA numérico
.    GERAMATRIZ()
.    GERAPESOSALEATORIOS()
.    MAXIMOSEMINIMOS(ARQUIVO, MAXIMO, MINIMO)
.    CALCULALAMBDAECE()
.    EPOCA ← 1
.    repita
.         se EPOCA>MAXEPOCA
.              então interrompa
.         fim se
.         REALEATORIZAMATRIZTREINO()
.         ERROEP[EPOCA] ← 0.0
.         N ← 1
.         repita
.              se N> NTREINO
.                   então interrompa
.              fim se
.              FUSIFICAPADRAO(N)
.              CALCULASAIDAE()
.              CALCULASAIDAOU()
.              CALCULAERROPADRAO(N)
.              CALCULAGRAD2()
.              CALCULAGRAD1()
.              CALCULATETAMIN()
.              ATUALIZAPESOS2()
.              ATUALIZAPESOS1()
.              NORMADEDELTA ← SQRT(NORMA1 + NORMA2)
.              TRUNCAPESOS2()
.              TRUNCAPESOS1()
.              ERROEP[EPOCA] ← ERROEP[EPOCA] + ERRO[N]
.              se EPOCA MOD INTEREP = 0
.                   então CALCULAETALP(EPOCA)
.              fim se
.              N ← N + 1
.         fim repita
.         ERROEP[EPOCA] ← ERROEP[EPOCA]/NTREINO
.         EPOCA ← EPOCA + 1
.    fim repita
.    SALVAPESOS2()
.    SALVAPESOS1()
fim algoritmo.
```

```
subrotina CALCULATETETAMIN()
.       TETAMIN ← 99999
.       J ← 1
.       repita
.               se J>N3
.                       então interrompa
.               fim se
.               I ← 1
.               repita
.                       se I>N2
.                               então interrompa
.                       fim se
.                       K ← I * N3
.                       K ← K + J
.                       se (PESOS2[K]+1.0)<(1.0-PESOS2[K])
.                               então TETA ← PESOS2[K]+1
.                               senão TETA ← 1.0-PESOS2[K]
.                       fim se
.                       se TETA<TETAMIN
.                               então TETAMIN ← TETA
.                       fim se
.                       I ← I + 1
.               fim repita
.               J ← J + 1
.       fim repita

.       I ← 1
.       repita
.               se I>N2
.                       então interrompa
.               fim se
.               L ← 1
.               repita
.                       se L>N1
.                               então interrompa
.                       fim se
.                       K ← L * N2
.                       K ← K + I
.                       se (PESOS1[K]+1.0)<(1.0-PESOS1[K])
.                               então TETA ← PESOS1[K]+1
.                               senão TETA ← 1.0-PESOS1[K]
.                       fim se
.                       se TETA<TETAMIN
.                               então TETAMIN ← TETA
.                       fim se
.                       L ← L + 1
```

```
.                  fim repita
.                  I ← I + 1
.       fim repita
fim subrotina.

subrotina ATUALIZAPESOS2()
.       NORMA2 ← 0
.       J ← 1
.       repita
.                  se J>N3
.                           então interrompa
.                  fim se
.                  I ← 1
.                  repita
.                           se I>N2
.                                    então interrompa
.                           fim se
.                           K ← I * N3
.                           K ← K + J
.                           DELTA2[K] ← ETHAV[ETALP] * (GRAD2[K]) +
.                              ALPHAV[ETALP] * (PESOS2[K] - PESOS2ANTERIOR[K])
.                           PESOS2ANTERIOR[K] ← PESOS2[K]
.                           PESOS2[K] ← PESOS2[K] + DELTA2[K]
.                           NORMA2 ← NORMA2 + quad(DELTA2[K])
.                           I ← I + 1
.                  fim repita
.                  J ← J + 1
.       fim repita
fim subrotina.

subrotina ATUALIZAPESOS1()
.       NORMA1 ← 0
.       I ← 1
.       repita
.                  se I>N2
.                           então interrompa
.                  fim se
.                  L ← 1
.                  repita
.                           se L>N1
.                                    então interrompa
.                           fim se
.                           K ← L * N2
.                           K ← K + I
.                           DELTA1[K] ← ETHAV[ETALP] * (GRAD1[K]) +
.                              ALPHAV[ETALP] * (PESOS1[K] - PESOS1ANTERIOR[K])
.                           PESOS1ANTERIOR[K] ← PESOS1[K]
```

```
.                    PESOS1[K] ← PESOS1[K] + DELTA1[K]
.                    NORMA1 ← NORMA1 + quad(DELTA1[K])
.                    L ← L + 1
.            fim repita
.            I ← I + 1
.    fim repita
fim subrotina.

subrotina TRUNCAPESOS2()
.    J ← 1
.    repita
.            se J>N3
.                    então interrompa
.            fim se
.            I ← 1
.            repita
.                    se I>N2
.                            então interrompa
.                    fim se
.                    K ← I * N3
.                    K ← K + J
.                    se (PESOS2[K]<=-1 ou PESOS2[K]>=1
.                            então PESOS2[K] ← PESOS2ANTERIOR[K] +
.                              TETAMIN * (DELTA2[K]/NORMADEDELTA)
.                    fim se
.                    I ← I + 1
.            fim repita
.            J ← J + 1
.    fim repita
fim subrotina.

subrotina TRUNCAPESOS1()
.    I ← 1
.    repita
.            se I>N2
.                    então interrompa
.            fim se
.            L ← 1
.            repita
.                    se L>N1
.                            então interrompa
.                    fim se
.                    K ← L * N2
.                    K ← K + I
.                    se (PESOS1[K]<=-1 ou PESOS1[K]>=1
.                            então PESOS1[K] ← PESOS1ANTERIOR[K] +
.                              TETAMIN * (DELTA1[K]/NORMADEDELTA)
```

. fim se
. I ← I + 1
. fim repita
. J ← J + 1
. fim repita
fim subrotina.

subrotina CALCULAETALP(N numérico)
. se (ETALP < TAMANHO(ALPHAV) e
. (ERRO[N]-ERRO[N-INTEREP] < DELTAPEQUENO)
. então ETALP ← ETALP + 1
. fim se
fim subrotina.

Abaixo, seguem as funções com as suas respectivas descrições: subrotina GERAMATRIZ()
Descrição: lê o arquivo de dados e preenche as matrizes de treino $TREINO$ e desejada DES com seus valores. No caso do conjunto de dados *Iris*, este arquivo corresponderá a uma matriz de 150 linhas por 5 colunas. As 150 linhas correspondem ao conjunto de dados das amostras das flores íris, as quais foram objeto de classificação da RNDD. A primeira coluna do conjunto de dados corresponderá ao comprimento da sépala (CS), a segunda, à largura da sépala (LS), a terceira, ao comprimento da pétala (CP) e a quarta, à largura da pétala (LP). A quinta coluna corresponderá à sub-espécie de íris a que pertencem os dados de cada linha, sendo 0 para Íris Setosa, 1 para Íris Versicolor e 2 para Íris Virgínica. Após a leitura dos dados, é efetuada uma aleatorização da ordem das linhas para que se evite o agrupamento das classes. Ao fim de sua execução, a matriz $TREINO$ conterá os quatro valores dos atributos e a matriz DES conterá em suas três colunas os valores referentes à classe da respectiva linha na matriz $TREINO$, sendo a tripla $(1, -1, -1)$ para Íris Setosa, $(-1, 1, -1)$ para Íris Versicolor e $(-1, -1, 1)$ para Íris Virgínica. Ver Seção 3.1, sobre o treinamento de uma rede neural.

subrotina GERAPESOSALEATORIOS()
Descrição: preenche as matrizes de pesos $PESOS1$ e $PESOS2$ com valores aleatórios no intervalo $]-1, 1[$, com até seis casas decimais. Ver Seção 3.3.7.

subrotina MAXIMOSEMINIMOS(ARQUIVO,MAXIMO,MINIMO)
Descrição: obtém o valor máximo e mínimo de cada atributo, com base na matriz $TREINO$. Cada posição do vetor $MINIMO$ irá conter o valor mínimo para o atributo correspondente à posição indicada. O mesmo vale para o vetor $MAXIMO$. Ver Seção 3.3.1.

subrotina CALCULALAMBDAECE()
Descrição: calcula e devolve, nas matrizes de parâmetros $LAMBDA$ e CE os valores obtidos, aplicando a função descrita na Seção 3.3.1 sobre os valores mínimos e máximos dos atributos da matriz $TREINO$ em execução. As colunas da referida matriz correspondem aos atributos e as linhas, aos valores *baixo*, *médio* e *alto* para cada atributo. Ver Seção 3.3.1.

subrotina REALEATORIZAMATRIZTREINO()

Descrição: faz a permutação das linhas das matrizes $TREINO$ e DES de forma a aleatorizar sua seqüência a cada época. O processo é feito de forma que para cada linha que for trocada na matriz $TREINO$ sua correspondente na matriz DES seja trocada também. Ver Seção 1.3, sobre a heurística de apresentação aleatória dos padrões de treino a cada época.

subrotina FUSIFICAPADRAO(N)
Descrição: preenche a matriz de saídas da camada de entrada difusa $SAIDAS1$ com o resultado da difusificação dos valores do padrão que acabou de ser apresentado à rede. As linhas correspondem aos atributos e as colunas, aos graus de pertinência aos conjuntos difusos *baixo*, *médio* e *alto*. Ver Seção 3.3.1.

subrotina CALCULASAIDAE()
Descrição: calcula o valor das saídas dos neurônios da camada intermediária e os armazena na matriz de saídas da camada intermediária $SAIDAS2$ (conjunção de disjunções). Estes valores são calculados com base na matriz $SAIDAS1$ e nos valores da primeira matriz de pesos $PESOS1$ (conexões entre a camada de entrada e a camada de intermediária). Ver Seção 3.3.2.

subrotina CALCULASAIDAOU()
Descrição: calcula o valor das saídas dos neurônios da camada de saída e os armazena na matriz de saídas da camada de saída $SAIDAS3$ (disjunção de conjunções). Estes valores são calculados com base na matriz $SAIDAS2$ e nos valores da segunda matriz de pesos $PESOS2$ (conexões entre a camada intermediária e a camada de saída). No caso do *Iris*, cada posição na matriz de saídas da camada de saída corresponde a uma sub-espécie do conjunto de dados *Iris*. Ver Seção 3.3.3.

subrotina CALCULAERROPADRAO(N)
Descrição: calcula o erro de cada saída do padrão atual ($ERRO$) em processamento na rede ($ERRO$) e atualiza o valor do EQM, armazenando-o na matriz de erro $ERROSSAIDA$. Ver Seção 3.3.5.

subrotina CALCULAGRAD2()
Descrição: calcula a derivada dos erros (gradiente) em relação à $PESOS2$ e armazena em $GRAD2$. Ver Seção 3.3.6.

subrotina CALCULAGRAD1()
Descrição: calcula a derivada dos erros (gradiente) em relação à $PESOS1$ e armazena em $GRAD1$. Ver Seção 3.3.6.

subrotina CALCULATETAMIN()
Descrição: calcula, a cada apresentação de padrão à rede, o $TETAMIN$ das matrizes $PESOS1$ e $PESOS2$ para ser utilizado na função de truncagem. Ver Seção 3.3.7.

subrotina ATUALIZAPESOS2()
Descrição: reajusta $PESOS2$, utilizando-se da fórmula para reajuste de pesos descrita na Seção 3.3.6, além de calcular o valor de $NORMA2$ para a truncagem dos pesos. A matriz com os pesos anteriores $PESOS2ANTERIOR$ receberá os valores dos pesos antes da atualização.

<u>subrotina</u> ATUALIZAPESOS1()
Descrição: reajusta $PESOS1$, utilizando-se da fórmula de reajuste dos pesos descrita na Seção 3.3.6, além de calcular o valor de $NORMA1$ para a truncagem dos pesos. A matriz com os pesos anteriores $PESOS1ANTERIOR$ receberá os valores dos pesos antes da atualização.

<u>subrotina</u> TRUNCAPESOS2()
Descrição: trunca os pesos da matriz $PESOS2$ para que os mesmos permaneçam no intervalo $]-1, 1[$. Para maiores detalhes, consultar Seção 3.3.7.

<u>subrotina</u> TRUNCAPESOS1()
Descrição: trunca os pesos da matriz $PESOS1$ para que os mesmos permaneçam no intervalo $]-1, 1[$. Para maiores detalhes, consultar Seção 3.3.7.

<u>subrotina</u> SALVARPESOS2()
Descrição: salva, no arquivo **w2final.dat**, os valores da matriz $PESOS2$, após a passagem de todos os padrões em todas as épocas.

<u>subrotina</u> SALVARPESOS1()
Descrição: salva, no arquivo **w1final.dat**, os valores da matriz $PESOS1$, após a passagem de todos os padrões em todas as épocas.

<u>subrotina</u> CALCULAETALP()
Descrição: é chamada a cada $INTEREP$ épocas para incrementar o índice $ETALP$, com o objetivo de utilizar os $ETALP$-ésimos valores dos vetores $ETHAV$ e $ALPHAV$. Ver Seção 1.3, sobre a heurística de diminuição da taxa de aprendizagem e do momento.

Na Figura 7.1, é mostrada a estrutura de diretórios dos arquivos-fontes e executáveis que compõem os módulos do sistema RNDD.

A descrição de cada arquivo é dada a seguir:

- **RNDD.cpp**: arquivo-fonte em C com a implementação da RNDD;
- **RNDD.exe**: arquivo binário executável que lê os arquivos **treino.dat**, **teste.dat** e **maxmin.dat** e gera os arquivos **w1final.dat** e **w2final.dat**;
- **treino.dat**: arquivo-texto com os exemplos de treino;
- **teste.dat**: arquivo-texto com os exemplos de teste;
- **maxmin.dat**: arquivo-texto com os valores máximos e mínimos de cada atributo;
- **w1final.dat** e **w2final.dat**: arquivos-texto com os pesos após o término do treino da rede;
- **RNDD.bat**: arquivo em lote com os parâmetros utilizados no teste final com o conjunto *Vogal*;
- **backtrack.m**: script MatLab que implementa o algoritmo de *backtracking*. Este script lê os arquivos **maxmin.dat**, **w1final.dat**, **w2final.dat** e **entradas.dat** para gerar as regras que serão gravadas no arquivo **regras.dat**;

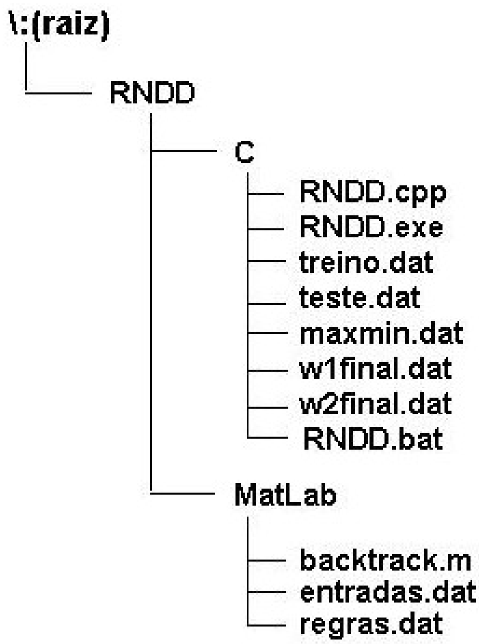

Figura 7.1: Estrutura de diretórios para o sistema RNDD.

- **entradas.dat**: arquivo-texto com as entradas a serem utilizadas pelo arquivo **backtrack.m**;
- **regras.dat**: arquivo-texto com as regras geradas a partir do arquivo **entradas.dat**.

A RNDD foi implementada em C. O módulo de geração de regras por *backtracking* foi implementado por meio de scripts *MatLab*, que fazem a leitura seqüencial dos arquivos de pesos e do arquivo com os exemplos de teste. Os programas se comunicam utilizando rotinas de importação e de exportação de arquivos-texto.

Referências Bibliográficas

[AFAC+05] R. ANDONIE, L. FABRY-ASZTALOS, C. COLLAR, S. ABDUL-WAHID, e N. SALIM. Neuro-fuzzy prediction of biological activity and rule extraction for hiv-1 protease inhibitors. *Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, páginas 113–120, 2005.

[AG95] R. ANDREWS e S. GEVA. Rulex e cebp networks as the basis for a rule refinement system. *Hybrid Problems Hybrid Solutions*, páginas 1–12, 1995.

[AND35] E. ANDERSON. The irises of the gaspe peninsula. *Bull. Amer. Iris Soc.*, páginas 2–5, 1935.

[AZM97] A. L. A. ARAÚJO, M. B. ZANUSSO, e L. F. J. MAIA. Novos operadores conjuntivos e disjuntivos para conjuntos difusos. *XVII-Congresso da Sociedade Brasileira de Computação - I ENIA*, páginas 214–223, 1997.

[BABZ99] L. M. BRASIL, F. M. AZEVEDO, J. M. BARRETO, e M. B. ZANUSSO. Técnica de extração de regras para sistemas especialistas conexionistas crisp e fuzzy. *Proceedings of IV Brazilian Conference on Neural Networks*, páginas 988–999, 1999.

[BKP05] P. BOSC, D. KRAFT, e F. PETRY. Fuzzy sets in database and information systems: Status and opportunities. *Fuzzy Sets and Systems*, páginas 418–426, 2005.

[CF00] G. CASTELLANO e A. M. FANELLI. Fuzzy inference and rule extraction using a neural network. *Neural Network World Journal*, páginas 361–371, 2000.

[CM03] S. CHOUDHURY e S. MITRA. Neurofuzzy classification and rule generation of modes of radiowave propagation. *IEEE Transactions on Antennas and Propagation*, páginas 862–871, 2003.

[COH60] J. A. COHEN. A coefficient of agreement for nominal scales. *Educational and Measurement*, páginas 37–46, 1960.

[CPS98] K. CIOS, W. PEDRYCZ, e R. W. SWINIARSKI. *Data Mining Methods for Knowledge Discovery*. Kluwer Academic Publishers, 1998.

[CS96] M. W. CRAVEN e J. W. SHAVLIK. Extracting tree-structured representations of trained networks. *Advances in Neural Information Processing Systems*, páginas 24–30, 1996.

[DOM82] J. DOMBI. A general class of fuzzy operators, the morgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy Sets and Systems*, páginas 149–163, 1982.

[dOZ05a] F. R. de OLIVEIRA e M. B. ZANUSSO. Clusterização de ocorrências policiais utilizando k-means e um mapa auto-organizável. *CBRN 2005*, páginas 338–353, 2005.

[dOZ05b] F. R. de OLIVEIRA e M. B. ZANUSSO. A fuzzy neural network with differentiable t-norms: Classification and rule generation. *IC-AI*, páginas 195–201, 2005.

[DP88] D. DUBOIS e H. PRADE. New results about properties and semantics of fuzzy set-theoretic operators. *Fuzzy Sets - Theory and Applications to Policy Analysis and Information Systems*, páginas 59–75, 1988.

[FBF+99] H. FARRER, C. G. BECKER, E. C. FARIA, H. F. MATOS, M. A. SANTOS, e M. L. MAIA. *Algoritmos Estruturados*. LTC, 1999.

[FU94] L. M. FU. Rule generation from neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, páginas 1114–1124, 1994.

[FUK90] K. FUKUNAGA. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[GAL88] S. I. GALLANT. Connectionist expert systems. *ACM Press*, páginas 152–169, 1988.

[GFP93] F. A. C. GOMIDE, M. FIGUEIREDO, e W. PEDRYCZ. A fuzzy neural network: Structure and learning. *Científico Internacional, 5th IFSA World Congress*, páginas 1171–1174, 1993.

[GOG68] J. A. GOGUEN. The logic of inexact concepts. *Syntese*, páginas 325–373, 1968.

[GOM02] A. K. GOMES. *Análise do Conhecimento Extraído de Classificadores Simbólicos utilizando Medidas de Avaliação e de Interessabilidade*. Tese de Doutoramento, ICMC - USP, 2002.

[GR94] M. M. GUPTA e D. H. RAO. On the principles of fuzzy neural networks. *Fuzzy Sets and Systems*, páginas 1–18, 1994.

[HAY99] S. HAYKIN. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[HEB49] D. HEBB. *The Organization of Behavior*. Wiley Press, 1949.

[HLL99] F. HUSSAIN, H. LIU, e H. LU. On relative measure for mining interesting rules. *Principles of Data Mining and Knowledge Discovery*, páginas 15–23, 1999.

[HUL05] E. HULLERMEIER. Fuzzy methods in machine learning and data mining: Status and prospects. *Fuzzy Sets and Systems*, páginas 387–406, 2005.

[JAN93] J.-S. R. JANG. Anfis: Adaptive network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, páginas 665–685, 1993.

[KEE94] J. D. KEELER. Predictive control in practice. *Jornal A*, páginas 54–57, 1994.

[KOV97] Z. L. KOVÁCS. *O Cérebro e a sua Mente: Uma Introdução à Neurociência Computacional*. Edição Acadêmica, 1997.

[KY95] G. KLIR e B. YUAN. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall, 1995.

[KZ02] W KLOSGEN e J. M. ZYTKOW. *Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, 2002.

[LAK73] G. LAKOFF. Hedges: a study in meaning criteria and the logic of fuzzy concepts. *Journal of Philosophical Logic*, páginas 458–508, 1973.

[MAI91] L. F. J. MAIA. *Caracterização e Reconhecimento de Conceitos*. Tese de Doutoramento, UNICAMP - FEE, 1991.

[MEN42] K. MENGER. Statistical metrics. *Proc. Nat. Acad. Sci.*, páginas 535–537, 1942.

[MP94] S. MITRA e S. K. PAL. Logical operation based fuzzy mlp for classification and rule generation. *Neural Networks*, páginas 353–373, 1994.

[MP02] S. MITRA e S. K. PAL. Data mining in soft computing framework: a survey. *IEEE Transactions on Neural Networks*, páginas 3–14, 2002.

[NKK97] D. NAUCK, F. KLAWONN, e R. KRUSE. *Foundations of Neuro-Fuzzy Systems*. John Wiley & Sons, Inc., 1997.

[PG98] W. PEDRYCZ e F. GOMIDE. *An Introduction to Fuzzy Sets: Analysis and Design*. MIT Press, 1998.

[PM77] S. K. PAL e D. D. MAJUMDER. Fuzzy sets and decision making approaches in vowel and speaker recognition. *IEEE Transactions on Systems, Man and Cybernetics*, páginas 625–629, 1977.

[PM92] S. K. PAL e S. MITRA. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on Neural Networks*, páginas 683–697, 1992.

[PT92] L. F. PAU e G. TZCHE. Explanation facility for neural networks. *Journal of Intelligent and Robotic Systems*, páginas 193–206, 1992.

[REZ03] S. O. REZENDE. *Sistemas Inteligentes*. Editora Manole, 2003.

[RHW86] D. E. RUMELHART, G. E. HINTON, e R. J. WILLIAMS. *Learning Internal Representation by Error Propagation*. MIT Press, 1986.

[RN04] S. RUSSELL e P. NORVIG. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2004.

[SS63] B. SCHWEIZER e A. SKLAR. Associative functions and abstract semigroups. *Publicationes Mathematicae*, páginas 69–81, 1963.

[TKR+95] H. TOIVONEN, M. KLEMETTINEN, P. RONKAINEN, K. HATONEN, e H. MANNILA. Pruning and grouping discovered association rules. *Machine Learning and Discovery in Databases*, páginas 47–52, 1995.

[TS85] T. TAKAGI e M. SUGENO. Fuzzy identification of systems and its application to modelling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, páginas 116–132, 1985.

[WIN72] T. WINOGRAD. *Understanding Natural Language*. Academic, New York, 1972.

[WKB97] K. WOODS, W. P. KEGELMEYER, e K. BOWYER. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, páginas 405–410, 1997.

[XXL05] K. XIE, G. XIE, e X. LI. An aproach to generate fuzzy rules based on rough set and fuzzy neural network. *IEEE Transactions on Neural Networks*, páginas 349–352, 2005.

[YRXY01] G. YINGJIE, H. RUNAN, K. XIANGDONG, e W. YIQUN. Fuzzy model in predictive control for hydraulic pressure system. *Fifth International Conference On Fluid Power Transmission And Control - ICFP2001*, páginas 40–42, 2001.

[ZAD65] L. A. ZADEH. Fuzzy sets. *Information and Control*, páginas 338–353, 1965.

[ZAM97] M. B. ZANUSSO, A. L. A. ARAÚJO, e L. F. J. MAIA. T-normas e t-conormas - projeto de tese. *XVII-Congresso da Sociedade Brasileira de Computação - I ENIA*, 1997.

[ZAN97] M. B. ZANUSSO. *Famílias de T-normas Diferenciáveis, Funções de Pertinência Relacionadas e Aplicações*. Tese de Doutoramento, UFSC, 1997.

[ZTR04] M. H. F. ZARANDI, I. B. TURKSEN, e B. REZAEE. A systematic approach to fuzzy modeling for rule generation from numerical data. *IEEE Annual Meeting of the North American Fuzzy Information Processing Society*, páginas 768–773, 2004.