Universidade Federal de Santa Catarina - UFSC
Programa de Pós-Graduação em Engenharia Elétrica
Departamento de Engenharia Elétrica
Grupo de Pesquisas em Engenharia Biomédica

Proposta de Arquitetura para Sistema Especialista Híbrido e a Correspondente Metodologia de Aquisição do Conhecimento

Tese Submetida à Universidade Federal de Santa Catarina como Parte dos Requisitos para Obtenção do Grau de Doutor em Engenharia Elétrica - Área de Concentração em Sistemas de Informação

Lourdes Mattos Brasil

Florianópolis, 1999

Proposta de Arquitetura para Sistema Especialista Híbrido e a Correspondente Metodologia de Elicitação do Conhecimento

Lourdes Mattos Brasil

ESTA TESE FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE

DOUTOR EM ENGENHARIA ELÉTRICA

ÁREA DE CONCENTRAÇÃO EM SISTEMAS DE INFORMAÇÃO, E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO

Prof. Fernando Mendes de Azevedo, Dr.
Orientador

Prof. Ildemar Cassana Decker, D.Sc.
Coordenador do Curso de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Prof. Dr. Jorge Muniz Barreto, Co-Orientador

Profa. Dra. Monique Noirhomme-Fraiture

Prof. Dr. Júlio César Nievola

Prof. Dr. Renato Garcia Ojeda

Profa. Dra. Eliana Ternes Pereira

"A Deus, pela sua infinita graça e bençãos.
A minha mãe, Helena Mattos Brasil."

Agradecimentos

Primeiramente gostaria de agradecer a Deus por ter me capacitado com discernimento e sabedoria para realizar este trabalho. Sem Ele, provavelmente, não teria finalizado esta pesquisa científica;

Quero agradecer imensamente a minha mãe que compartilhou e teve muita paciência comigo durante todo o desenvolvimento deste trabalho;

A minha família, que me auxiliou grandemente em mais esta etapa de minha vida acadêmico-profissional;

A minha família espiritual da Igreja Metodista do Estreito e Itacurubi, que através das orações, carinho, companherismo e amizade, fizeram-me crescer e perseverar na minha vida acadêmica;

Á minha madrinha Edith Long Schisler e seu esposo Pr. Willian Richard Schisler Filho (*in memoriam*) não somente pelo carinho e dedicação, mas também pelo incentivo e apoio espiritual, profissional, sentimental, etc.;

Aos meus amigos do Grupo de Pesquisas em Engenharia Biomédica (GPEB), que me auxiliaram de forma direta e indireta no desenvolvimento desta pesquisa;

Às minhas amigas Ana Regina, Maria Bernadete, Kathya e Cidinha, não somente pelo apoio científico, mas pelo carinho e dedicação para comigo;

Aos meus amigos John e Mauro, pelas respostas e sugestões a vários questionamentos sobre o meu tema de pesquisa;

Aos Professores F.M. de Azevedo e J.M. Barreto pelo crescimento intelectual e profissional, pelo incentivo, pelas sugestões e pela amizade durante todos os anos de pesquisa e desenvolvimento deste trabalho;

Aos Professores e Especialistas Médicos L.S. Min, P.C.T. Bittencourt e E.T. Pereira pelo auxílio na compreensão dos termos técnicos da área médica aqui aplicada, bem como pela motivação no desenvolvimento deste trabalho;

À Anne de Baenst-Vandenbroucke (FUNDP, Bélgica) por seus comentários, acolhimento e sua amizade sincera durante o perído que estive no seu país;

À Profa. Monique Noirhomme-Fraiture (FUNDP, Bélgica) por ter me dado condições para a minha ída ao seu país e universidade para realizar o doutorado-*sandwich*;

Por fim, à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), por ter dado o apoio financeiro para o desenvolvimento do trabalho em estudo.

Publicações

Congressos e Simpósios

1. L.M. Brasil, F.M. de Azevedo, R. Garcia Ojeda, and J.M. Barreto, Cooperation of Symbolic and Connectionist Expert System Techniques to Overcome Difficulties, *In: Proceedings do II Congresso Brasileiro de Redes Neurais*, p.177-182, Curitiba, Brazil, 1995.
2. L.M. Brasil, F.M. de Azevedo, R. Garcia Ojeda, and J.M. Barreto, A Methodology for implementing hybrid expert systems, *In: Proceedings of The IEEE Mediteranean Electrotechnical Conference (MELECON'96)*, ISBN 0-7803-3109-5, p.661–664, Bari, Italy, 1996.
3. L.M. Brasil, F.M. de Azevedo, R. Garcia Ojeda, and J.M. Barreto, Extração de regras para sistemas especialistas conexionistas. *Anales del IV Coloquio de Bioingenieria (COLOQUIO'97)*, ISBN 980-00-1112-9, p.TS-40–TS-46, Valência, Venezuela, 1997.
4. L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and R.Garcia Ojeda, A hybrid architecture for expert systems, *In: Proceedings of The World Congress On Medical Physics and Biomedical Engineering*, ISSN 0140-018/97, Nice, France, v.35, p.517, F74-PSI.08, Supplement Part I, 1997.
5. L.M. Brasil, F.M. de Azevedo, and J.M. Barreto, Uma arquitetura híbrida para sistemas especialistas, *In: Proceedings do III Congresso Brasileiro de Redes Neurais (CBRN)*, ISBN 85-900382-1-1, p.167-172, Florianópolis, Brasil, UFSC, 1997.
6. L.M. Brasil, F.M. de Azevedo, and J.M. Barreto, Uma arquitetura para sistema neuro-fuzzy-ga, *Anales del XII Congreso Chileno de Ingeniería Eléctrica*, Universidad de La Frontera, Temuco, Chile, v.II, p.712-717, 1997.
7. L.M. Brasil, F.M. de Azevedo, and J.M. Barreto, Learning algorithm for connectionist systems, *Anales del XII Congreso Chileno*

de Ingeniería Eléctrica, Universidad de La Frontera, Temuco, Chile, v.II, p.697-702, 1997.

8. L.M. Brasil, F.M. de Azevedo, and J.M. Barreto, A hybrid expert architecture for medical diagnosis, *In: Proceedings of The 3th International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'97)*, ISBN 3-211-83087-1, p.176-180, Norwich, England, April 1997.

9. L.M. Brasil, F.M. de Azevedo, J.M. Barreto and, M. Noirhomme-Fraiture, Training algorithm for neuro-fuzzy-ga systems. *In: Proceedings of The 16th IASTED International Conference on Applied Informatics (AI'98)*, ISSN 1027-2666, ISBN 0-88986-250-8, p.45–47, Garmisch-Partenkirchen, Germany, February 1998.

10. M. Roisenberg, F.M. de Azevedo, J.M. Barreto and, L.M. Brasil, On a formal concept of autonomous agents. *In: Proceedings of The 16th IASTED International Conference on Applied Informatics (AI'98)*, ISSN 1027-2666, ISBN 0-88986-250-8, p.41–44, Garmisch-Partenkirchen, Germany, February 1998.

11. J.S. Dias, J.M. Barreto, S. Nassar and, L.M. Brasil, Genetic and back-propagation algorithm in hybrid training of artificial neural networks: a unimodal search procedure. *In: Proceedings of The 16th IASTED International Conference on Applied Informatics (AI'98)*, ISSN 1027-2666, ISBN 0-88986-250-8, p.37–40, Garmisch-Partenkirchen, Germany, February 1998.

12. L.M. Brasil, F.M. de Azevedo, J.M. Barreto and, M. Noirhomme-Fraiture, A new approach to classical backpropagation algorithm for neuro-fuzzy-ga systems learning, *In: Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing (ASC'98)*, ISSN 1482-7913, ISBN 0-88986-256-7, p.489–492, Cancún, Mexico, 1998.

13. L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and M. Noirhomme-Fraiture, A neuro-fuzzy-ga system architecture for helping the knowledge acquisition process, *In: Proceedings of The International IEEE Joint Symposia on Intelligence and Systems*, Washington-DC, USA, 1998 (In Press).

14. L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and M. Noirhomme-Fraiture, Knowledge acquisition using a hybrid expert system, *In:*

Proceedings of The World Multiconference on Systemics, Cybernetics and Informatics and, 4th International Conference on Information Systems, Analysis and Synthesis (SCI'98-ISAS'98), ISBN 980-07-5079-7, v.2, p.84-91, Florida, USA, 1998.

15. L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and M. Noirhomme-Fraiture, Complexity and cognitive computing, *In: Proceedings of the 11th International Conference on Industial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA-98-AIE)*, Castellon, Spain, 1998 (In Press).

16. L.M.Brasil, F.M. de Azevedo, J.M. Barreto, and M. Noirhomme-Fraiture, A new hybrid architecture for helping the knowledge acquisition processo, *In: Proceedings of The 15th International Congress on Cybernetics - Interdisciplinary Symposium in Artificial Intelligence, Cognitive Science, and Philosophy for Social Progress*, Namur, Belgium, 1998 (In Press).

Revista

1. L.M. Brasil, F.M. De Azevedo, and J.M. Barreto, Algoritmo de Aprendizado para Redes Neurais Fuzzy AND/OR, Pesquisa Naval - Suplemento Especial da Revista Marítima Brasileira, ISSN 1414-8595, n.10, p.111-132, 1997.

Sumário

Sumário vi

Lista de Figuras ix

Lista de Tabelas xi

Lista de Abreviaturas xi

Resumo xiii

Abstract xv

1 Introdução 1

1.1 Objetivos 3

1.1.1 Objetivo Geral 3

1.1.2 Objetivos Específicos 3

1.2 Justificativa 3

1.3 Sistemas Especialistas Híbridos 6

1.3.1 HES com Abordagem de L.M. Fu 9

1.3.2 HES com Abordagem de R. Sun 13

1.3.3 HES com Abordagem de R. Machado 16

1.3.4 HES com Abordagem de M. M. Gupta 18

1.3.5 HES com Abordagem de S. Mitra e S.K. Pal 22

1.3.6 HES com Abordagem de W. Pedrycz 23

1.3.7 HES com Abordagem de H.C. Fu e J.J. Shann 26

1.3.8 HES com Abordagem de Y. Hayashi 27

1.3.9 HES com Abordagem de S.K. Halgamuge e M. Glesner 28

1.3.10 HES com Abordagem de J.-S.R. Jang 31

1.3.11 HES com Abordagem de H.J. Zimmermann et al. 34

1.4 Técnicas de Extração de Regras para RNA 36

1.4.1 Refinamento do Conhecimento 38
1.4.2 Classificação 39
1.4.3 Extração de Regras Booleanas Usando Abordagens Decomposicionais 40
1.4.4 Extração de Regras Booleanas Usando Abordagens Pedagógicas 45
1.4.5 Extração de Regras Fuzzy 50

2 Fundamentação Teórica **58**
2.1 Sistemas Simbólicos 58
2.1.1 Fontes de Conhecimento 59
2.1.2 Base de Conhecimento 60
2.1.3 Aquisição de Conhecimento 61
2.1.4 Grafos E/OU 65
2.1.5 Cláusulas de Horn 68
2.2 Redes Neurais 71
2.2.1 Morfologia do Neurônio Biológico 72
2.2.2 Modelo Básico de um Neurônio 74
2.2.3 Topologias das RNA 76
2.2.4 Aprendizado em RNA 79
2.2.5 Regras de Aprendizado 81
2.3 Lógica fuzzy 90
2.4 Computação Evolucionária 95
2.4.1 Base Biológica 95
2.4.2 Algoritmos Genéticos 97
2.5 Problema do Raciocínio Médico 109

3 Sistema Proposto **116**
3.1 Fundamentação do Sistema 116
3.2 Descrição do Sistema 120

4 Metodologia da Pesquisa **128**
4.1 Aquisição de Conhecimento 128
4.2 Tradução das Regras Iniciais Fuzzy em Grafos E/OU 131
4.2.1 Exemplo 132
4.3 Tratamento das Variáveis de Entrada da RNA 135
4.4 Modelo Matemático do Neurônio 137
4.4.1 Confluência 139
4.4.2 Função de Ativação Não-Linear 140

4.4.3 Mapeamento Matemático da Entrada/Saída do NNES 142
4.4.4 Modelagem fuzzy 143
4.5 NNES Inicial 147
4.6 RBES 148
4.6.1 Descrição do Sistema Simbólico 149
4.6.2 Mapeamento em Grafos E/OU 150
4.6.3 Justificador 150

5 Algoritmo de Aprendizado (GENBACK) 152
5.1 GENBACK 153
5.1.1 Etapas do GENBACK 153
5.1.2 Algoritmo Genético 154
5.1.3 Problemas da Derivabilidade das Funções E/OU 155
5.1.4 Regra Delta para RNA E/OU 159
5.1.5 Algoritmo de Treinamento - GENBACK 161
5.2 Etapas do NNES 166
5.2.1 Atributos dos Dados Entrada/Saída 166
5.2.2 Conjunto de Exemplos 166
5.2.3 Número de Neurônios das Camadas de Entrada, Intermediária e Saída do NNES Inicial 166
5.2.4 Etapas do Algoritmo de Treinamento para o NNES 167
5.2.5 Refinamento do NNES 172

6 Algoritmo de Extração de Regras Fuzzy para RNA Fuzzy (FUZZY-RULEXT) 173
6.1 Descrição da Técnica de Extração de Regras para o NNES Fuzzy . . 174
6.1.1 Grafo E/OU 176
6.1.2 Geração do Caminho por Backtracking 176
6.1.3 Geração das Cláusulas 180
6.2 Algoritmo de Extração de Regras Proposto para RNA Fuzzy 187

7 Simulações e Resultados 190
7.1 Implementação do Sistema Proposto 191
7.2 Simulações para o NNES Fuzzy 197
7.2.1 Exemplo 1: Classificação dos Números de 0-15 em Pares, Primos e Ímpares 198
7.2.2 Resultados do Exemplo 1 203
7.2.3 Exemplo 2: Classificação das Vogais em Código ASCII 206

7.2.4 Resultados do Exemplo 2 209
7.2.5 Exemplo 3: Classificação de Crises Epilépticas 211
7.2.6 Resultados do Exemplo 3 219
7.3 Simulações para o RBES 223
7.3.1 Exemplos 225
7.3.2 Resultados das Simulações para o RBES 227

8 Epílogo **231**
8.1 Contribuições deste Trabalho 231
8.2 Conclusões 233
8.3 Trabalhos Futuros 240

Referências Bibliográficas **242**

Lista de Figuras

1.1 Trajetória básica do sistema visual do olho ao córtex visual 8
1.2 Nível semântico de uma rede *neuro-fuzzy* 9
1.3 Grafo E/OU 10
1.4 Uma rede conexionista baseada em regras 11
1.5 Substituindo valores de saídas múltiplas com nós intermediários múltiplos 15
1.6 Implementando um automata de estado finito em RNA 16
1.7 SE conexionista *fuzzy* 17
1.8 RNA com neurônios OU 19
1.9 Dois modelos de sistemas neurais *fuzzy* usando a topologia direta . . . 20
1.10 Uma RNA de três camadas com funções lógicas E e OU 22
1.11 Diagrama de bloco da fase de entrada do modelo proposto 23
1.12 Arquitetura do neurônio E/OU 24
1.13 Arquitetura do neurônio OU/E 25
1.14 A estrutura da RNA *fuzzy* 27
1.15 Uma RNA com grupos de células *abruptas* e *fuzzy* 28
1.16 A arquitetura de um sistema FuNe-I 29
1.17 Estrutura do ANFIS 33
1.18 Modelo *neuro-fuzzy*: sistemas *fuzzy* tipo Sugeno simples com regras simples 35
1.19 Exemplo do algoritmo SUBSET 42
1.20 Exemplo do algoritmo MofN 44

2.1 Grafo E/OU 67
2.2 Esquema simplificado de um neurônio 72
2.3 Neurônio artificial 75
2.4 Diagrama de blocos de um neurônio dinâmico 76
2.5 RNA com uma camada intermediária 77
2.6 Rede direta 79
2.7 Rede com ciclos 80

2.8 Uma rede multicamada 83
2.9 Pertinência *fuzzy* versus pertinência abrupta 92
2.10 Operador *crossover* de um ponto 104
2.11 Exemplo de mutação (troca simples) 105
2.12 Elementos do processo de diagnóstico médico baseado em heurística . 113

3.1 Diagrama de blocos de um SE simbólico básico 117
3.2 Diagrama de blocos de um NNES típico 118
3.3 Diagrama de blocos do HES 124

4.1 Diagrama de bloco do sistema geral 129
4.2 Relação especialista/engenheiro de conhecimento 130
4.3 Organização da base de regras como uma RNA 131
4.4 Grafo de inferência - varicela 133
4.5 Grafo de inferência - catapora 134
4.6 Grafo de inferência - sarampo 135
4.7 Grafo de inferência - caxumba 135
4.8 Tipos de variáveis de entrada 136
4.9 Exemplo de variável quantitativa 138
4.10 Modelo matemático de um neurônio estático 139
4.11 Medida de similaridade $(u(t) = X_a(t) \copyright W_a(t))$ 141
4.12 Diagrama de bloco das camadas da RNA 142
4.13 Neurônio *fuzzy* OU 146
4.14 Neurônio *fuzzy* E 146
4.15 Configuração inicial da RNA 148

5.1 Exemplo de código genético do número decimal 98 para codificar uma cadeia de cromossomos 154
5.2 Diagrama de fluxo de informação para minimização da pesquisa univariável 160

6.1 Uma visão geral do sistema proposto 175
6.2 Rede neural *fuzzy* 177
6.3 Exemplo 1 demonstrando a geração de regra por *backtracking* 183
6.4 Exemplo 2 demonstrando a geração de regra por *backtracking* 186
6.5 Exemplo 3 demonstrando a geração de regra por *backtracking* 188

7.1 Tela principal do sistema proposto 191
7.2 Tela do menu principal: *file* 192
7.3 Tela do menu principal: *edit* 193

7.4 Tela do menu principal: *run* 193
7.5 Tela do menu principal: *graphics* 194
7.6 Tela para o treinamento e refinamento do NNES *fuzzy* 194
7.7 Tela de criação e eliminação de redundância de regras 195
7.8 Tela do RBES: conversão e geração das regras 196
7.9 Tela do RBES: consulta feita à base de conhecimento 196
7.10 Tela do RBES: resultados 197
7.11 Tela do RBES: formalização das regras 198
7.12 Análise do desvio padrão: aptidão menor 214
7.13 Análise do desvio padrão: aptidão maior 215

Lista de Tabelas

2.1 Roleta Ponderada 101

3.1 Características gerais de algumas abordagens para HES 121
3.2 Características gerais de algumas abordagens para HES 122

7.1 Simulação 1: Exemplo 1 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 199
7.2 Simulação 1: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$. 200
7.3 Simulação 2: Exemplo 1 com $FA = \frac{N_{CI}}{Erro\_Total}$ 201
7.4 Simulação 2: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$ 202
7.5 Simulação 3: Exemplo 1 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 203
7.6 Simulação 3: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$. 203
7.7 Simulação 4: Exemplo 1 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ e *bias* 204
7.8 Simulação 4: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$. 204
7.9 Simulação 5: Exemplo 1 com $FA = \frac{N_{CI}}{Erro\_Total}$ e *bias* 205
7.10 Simulação 5: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$ e *bias* 205
7.11 Simulação 6: Exemplo 2 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 207
7.12 Simulação 6: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$. 208
7.13 Simulação 7: Exemplo 2 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 208
7.14 Simulação 7: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$. 209
7.15 Simulação 8: Exemplo 2 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ e *bias* 209
7.16 Simulação 8: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$ e *bias* 210
7.17 Simulação 9: Exemplo 2 com $FA = \frac{N_{CI}}{Erro\_Total}$ 210
7.18 Simulação 9: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$ 211
7.19 Simulação 10: Exemplo 2 com $FA = \frac{N_{CI}}{Erro\_Total}$ 211
7.20 Simulação 11: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$ . . . 212
7.21 Simulação 12: Exemplo 2 com $FA = \frac{N_{CI}}{Erro\_Total}$ e *bias* 212
7.22 Simulação 12:Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$ e *bias* 212
7.23 Simulação 13: Exemplo 3 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 213
7.24 Simulação 13: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$. 214

7.25 Simulação 14: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$ 215
7.26 Simulação 14: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$ . . . 216
7.27 Simulação 15: Exemplo 3 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 219
7.28 Simulação 15: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$. 220
7.29 Simulação 16: Exemplo 3 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 221
7.30 Simulação 16: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$. 222
7.31 Simulação 17: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$ 223
7.32 Simulação 17: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$ . . . 224
7.33 Simulação 18: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$ 225
7.34 Simulação 18: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$ . . . 226
7.35 Simulação 19: Exemplo 1 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 226
7.36 Simulação 20: Exemplo 1 com $FA = \frac{N_{CI}}{Erro\_Total}$ 227
7.37 Simulação 21: Exemplo 2 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 227
7.38 Simulação 22: Exemplo 2 com $FA = \frac{N_{CI}}{Erro\_Total}$ 227
7.39 Simulação 23: Exemplo 3 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 228
7.40 Simulação 24: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$ 228
7.41 Simulação 25: Exemplo 3 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 229
7.42 Simulação 26: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$ 229
7.43 Simulação 27: Exemplo 3 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ 229
7.44 Simulação 28: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$ 230

Lista de Abreviaturas

AC - Aquisição de Conhecimento
ADN - Ácido Deóxi-Ribonuclêico AE - Algoritmo Evolucionário
AG - Algoritmo Genético
ANFIS - Adaptive-Network-based Fuzzy Inference System
BAM - Bidirectional Associative Memory
BCC - Base de Conhecimento Conexionista
CEBP - Constrained Error Backpropagation
CRBR - Connectionist Rule-Based Reasoner
DN - Discrete Neural Model
EC - Elicitação de Conhecimento
EES - Explanation Expert System
FC - Fator de Certeza
FEL - Fuzzy Evidencial Logic
FNES - Fuzzy Neural Expert System
FA - Função de Ativação
FUZZYRULEXT - Fuzzy Rule Extraction Algorithm
GENBACK - Genetic-Backpropagation Based Learning Algorithm
GPEB - Grupo de Pesquisas em Engenharia Biomédica
HES - Hybrid Expert System
HU/UFSC - Hospital Universitário da Universidade Federal de Santa Catarina
IA - Inteligência Artificial
IAC - Interactive Activation and Competition
KBANN - Knowledge Based Artificial Neural Network
KBCNN - Knowledge Based Conceptional Neural Network
LD - Linearmente Dependente
LI - Linearmente Independente
LIA - Laboratório de Inteligência Artificial
LMS - Least Mean Square
LRU - Locally Responsive Units

LSE - Least Square Estimation
MNC - Modelo Neural Combinatorial
NNES - Neural Network Based Expert System
NEFCLASS - Neuro-Fuzzy Classification
NEFCON - Neuro-Fuzzy Controller
OG - Operador Genético
PDN - Probabilistic Discrete Neural Model
RBES - Rule-Based Expert System
RBF - Radial Basis Functions
RN - Rule Net
RNA - Rede Neural Artificial
SBC - Sistema Baseado no Conhecimento
SE - Sistema Especialista
SINTA - Sistemas Inteligentes Aplicados

Resumo

Este trabalho tem como meta propor uma metodologia para o desenvolvimento de um Sistema Especialista (SE) usando uma arquitetura híbrida. Este SE tem, como característica principal, a capacidade de aprender a extrair conhecimento a partir de uma base de conhecimento inicial e de um conjunto de exemplos. Desta forma, espera-se contribuir para a solução de um dos problemas da Inteligência Artificial (IA), que consiste na extração de conhecimento do especialista de domínio e que, nos sistemas simbólicos, é conhecido como uma das etapas da Aquisição de Conhecimento (AC).

A implementação do SE proposto conduz a um Sistema Especialista Híbrido (HES). Este sistema é formado por um Sistema Especialista Baseado em Redes Neurais (NNES) e de um Sistema Especialista Baseado em Regras (RBES). A idéia principal é de que o engenheiro de conhecimento consiga, num pequeno espaço de tempo, extrair algumas regras iniciais do especialista de domínio que, por sua vez, servirão para a definição da estrutura inicial de um NNES a ser refinado, posteriormente, através do conjunto de exemplos. Esta etapa de refinamento implica na existência de um algoritmo de aprendizado, que permite mudar a estrutura da rede, incluindo ou excluindo neurônios e/ou incluindo ou excluindo ligações. Esta abordagem conduz a uma representação de conhecimento localizada, isto é, o conhecimento somente é localizado no sistema inicial, onde neurônios representam conceitos e conexões representam relações entre conceitos. A partir do NNES refinado, quando o conhecimento deve estar distribuído, regras podem ser deduzidas. Geralmente, não é muito fácil se obter a explicação de como uma RNA chegou a uma conclusão. Portanto, propõe-se aqui também um algoritmo para a extração de regras. Além disso, estas regras podem ser usadas para formar um RBES, que poderá eventualmente dar explicações do resultado obtido na saída do NNES. Nesta etapa, o conhecimento volta a estar localizado.

A metodologia foi desenvolvida tendo em vista sua aplicação em Sistemas de

Apoio à Decisão na Área Médica, onde as tarefas de Elicitação e Representação de Conhecimento são das mais difíceis, devido às características particulares desse tipo de problema. No entanto, espera-se que o sistema possa possuir um razoável grau de generalidade de forma a que seja aplicado a outras áreas.

Palavras-Chaves: Algoritmo Genético, Redes Neurais, Sistemas Especialistas, Lógica *Fuzzy* e Inteligência Artificial.

Abstract

A Proposal of Architecture for Hybrid Expert System and a Corresponding Elicitation/Representation Methodology of Knowledge

The main goal of this work is to propose a new methodology for developing an Expert System (ES) using an hybrid architecture. The main characteristic of this ES is the ability of learning to elicit knowledge by means of a basic knowledge base and a set of examples. With this approach, it is expected to reduce one of the problems of the Artificial Intelligence (AI), i.e., the extraction task of the domain expert knowledge. In the symbolic systems this task is one of the stages of the Knowledge Acquisiton (KA).

Implementing this methodology leads to a Hybrid Expert System (HES). It consists of a Neural Network based Expert System (NNES) and a Rules Based Expert System (RBES). The main idea is that if the knowledge engineer has conditions to obtain some basic rules, and a set of examples, from the domain expert then it is possible to define the basic structure of the NNES using those basic rules. Next, the NNES can be refined using the set of examples. Follows a stage of refinement that consists of a learning algorithm to allow structural changes of the network by inclusion or exclusion of connections and/or neurons. This approach leads to a localized knowledge representation, i.e., knowledge is only localized in the basic system, where neurons are representing concepts and connections are representing relations among concepts. Rules can be deduced after the NNES had been refined, when knowledge should be distributed, because generally to obtain the explanation in the output of an ANN is not straighforward. So, we also propose an algorithm for rules extraction. Moreover, the NNES can be used to form a RBES that it will give an explanation of the obtained result in output of the NNES. In this stage, knowledge comes back to be localized.

The methodology developed to HES was used in implementing Decision Support

Systems in the Medical Area. In this case, Elicitation and Representation Knowledge tasks are one of the most hard processes because of the private characteristics of this kind of problem. However, we hope that the system can own a reasonable degree of generality. So it will be applied to other areas.

Key-words: Genetic Algorithm, Neural Networks, Expert Systems, Fuzzy Logic and Artificial Intelligence.

Capítulo 1

Introdução

Desenvolver um SE (simbólico ou conexionista), principalmente para uma área de domínio médico, trata-se de uma árdua tarefa. Uma das razões deve-se ao fato que, as informações que os médicos dispõem sobre seus pacientes, em geral, são caracterizadas pela imprecisão de seu conteúdo. Porém, ainda assim, os médicos são capazes de chegar à conclusões a respeito desses dados [1]. Outra razão, diz respeito ao processo de extração do conhecimento de um especialista. Sabe-se que este tipo de processo é considerado como o *gargalo* da área de Inteligência Artificial (IA) no que diz respeito a implementação de SE. Existem muitos trabalhos onde os pesquisadores estão preocupados em minimizar este tipo de problema, por exemplo, [18][11]. Contudo, ainda não se chegou a um processo adequado, onde todo, ou quase todo o conhecimento de um determinado especialista possa ser administrado por uma ferramenta computacional que seja plausível, no sentido de alcançar resultados próximos ao que se encontra no mundo real.

O conhecimento do mundo real, caracteriza-se por ser:

- Incompleto;
- Impreciso;
- Inconsistente.

Entretanto, o homem acostumou-se a buscar a precisão e a exatidão nos dados, esquecendo-se que nem sempre essas características podem ser encontradas nas informações usadas para tomar decisões. A tomada de decisão clínica é um claro exemplo disto onde o médico, durante suas horas de trabalho, enfrenta casos

caracterizados por dados imprecisos, incompletos e, em alguns casos, de natureza contraditória nos sintomas relatados pelo paciente. Entre estes dados, tem-se [116]:

1. História do Paciente: Durante a anamnese, a informação dada pelo paciente é altamente subjetiva. Pode-se encontrar exageros ou sintomas subestimados. Ignorância de doenças anteriores dele ou de sua família e erros na descrição de intervenções cirúrgicas já realizadas. No entanto, a informação que finalmente leva ao diagnóstico correto é muitas vezes encontrada na história do paciente.
2. Exame Físico: Durante o exame físico, os médicos poderiam cometer erros ao não considerar indicações importantes, ou enganar-se ao realizar o exame completo. Além disso, eles poderiam interpretar inadequadamente outras indicações, porque o limite entre o normal e o patológico não é claramente definido.
3. Resultados de Testes Laboratoriais: Os resultados de laboratório são considerados como dados objetivos. Porém, deve-se lembrar que erros nas medições, na colocação de etiquetas, troca de exames, ou comportamento inadequado do paciente no instante prévio ao teste, poderiam levar a imprecisões e, algumas vezes, a dados incorretos. Por outro lado, os limites entre resultados normais e patológicos são geralmente imprecisos, isto é, existem valores de fronteira dos quais não se pode dizer se são normais ou não.
4. Resultados Histológicos: Assim como os resultados obtidos em exames de raios X, ultrasom, tomografia computadorizada, ressonância magnética e outros exames clínicos, os exames histológicos dependem de interpretações corretas pelo médico. Tais descobertas são cruciais, porque elas freqüentemente indicam terapia invasiva. Em muitos casos, as considerações de imprecisão são parte dos procedimentos de avaliação, por exemplo, contagem de células, determinação do tipo de células, análise descritiva, etc.

Todavia pode-se afirmar que, em geral, os dados com os quais o médico tem que lidar são, na sua grande maioria, informações imprecisas, inexatas e incompletas.

Nas próximas seções, estará se tratando dos objetivos, das justificativas, dos sistemas especialistas híbridos e das técnicas de extração de conhecimento, abrangendo a linha dos problemas acima mencionados.

1.1 Objetivos

1.1.1 Objetivo Geral

Propor uma metodologia para uma arquitetura de sistema especialista híbrido, combinando as características favoráveis apresentadas pelos paradigmas conexionista e simbólico, visando facilitar a tarefa de Aquisição de Conhecimento (AC), que é um dos principais problemas em IA.

1.1.2 Objetivos Específicos

1. Propor uma arquitetura para Sistema Especialista Híbrido (HES), que ofereça condições para uma solução e explicar o porquê da mesma;
2. Propor uma metodologia para o processo de AC;
3. Propor uma metodologia para a definição da topologia da Rede Neural Artificial (RNA), do Sistema Especialista Baseado em Redes Neurais (NNES), constituinte do Sistema Especialista Híbrido (HES);
4. Desenvolver um algoritmo de aprendizado que permita a modificação da estrutura da RNA, de forma que possa criar ou deletar neurônios, representando conceitos, e/ou criar ou deletar relações entre neurônios;
5. Estudar o desempenho do HES, quando aplicado à Sistemas de Apoio à Decisão na Área Médica.

1.2 Justificativa

Os Sistemas Especialistas (SE) surgiram a cerca de trinta anos atrás, como uma aplicação do paradigma de manipulação simbólica da IA. Tradicionalmente, os SE têm usado técnicas de IA e bases de conhecimentos para simular a ação de especialistas humanos, com o propósito de solucionar problemas específicos em um dado domínio [37].

Nos dias de hoje, os SE são um dos produtos mais conhecidos da IA. Há várias áreas do conhecimento onde os SE têm provado sua eficiência, independentemente do paradigma de implementação adotado: manipulação simbólica ou conexionista. Por exemplo, no domínio de diagnóstico (MYCIN, PUFF, DARF), de predição (PROSPECTOR), de interpretação (DENDRAL), etc. [148][154][11].

Construir um SE, utilizando-se do paradigma de manipulação simbólica, leva a vários problemas: o primeiro deles, é o processo de AC. De fato, a elicitação de conhecimento de um especialista de domínio - a qual é um dos estágios da AC - é uma tarefa árdua e consome um tempo razoável, não só do engenheiro de conhecimento como também do especialista em questão. Além disso, seres humanos, mesmo que conhecedores da solução de um problema, têm dificuldade em explicar de que forma a encontram [18]. A diversidade no modo de representar o conhecimento de forma a codificar o raciocínio do especialista humano, representa mais uma dificuldade.

A representação de conhecimento é uma das áreas mais ativas da IA, envolvendo grandes desafios. A literatura de IA apresenta, aproximadamente, uma dezena de formas de representação do conhecimento, sendo que, o sistema de regras de produção é a mais comum. Ele se constitui de um conjunto de regras que reúne condições e ações. A condição é constituída por um padrão que determina a aplicabilidade da regra, enquanto que, a ação indica o que será realizado quando a mesma for aplicada. Assim, um sistema é formado por uma ou mais bases de regras, separadas segundo as conveniências de processamento e complementada, ainda, por uma estratégia de controle. Desta forma, estabelecendo as prioridades em que as mesmas serão aplicadas, bem como, critérios de desempate, quando houver possibilidade de aplicação de mais de uma regra ao mesmo tempo - isto é, resolução de conflito -, tem-se um SE conhecido como um Sistema Especialista Baseado em Regras (RBES).

Contudo, quando se deseja obter um conjunto de regras de especialistas humanos, tem-se várias dificuldades. Uma delas, diz respeito a maneira como eles articulam o conhecimento para atingir a solução de um dado problema. A multiplicidade de especialistas é outro problema neste sentido, pois muitas vezes, apresentam diferentes explicações para suas decisões, chegando alguns casos a divergirem na conclusão final. Além disso, quando a manipulação simbólica se utiliza de regras de produção, a opacidade é uma das características deste tipo de sistema, cuja completa realização é difícil de se constatar, bem como, a verificação dos possíveis fluxos de processamento. A ineficiência é outro fator que contribui para este tipo de problema, particularmente, quanto ao número de regras a combinar e, também, do esforço de *matching*[1] necessário ao suporte de execução das regras [154]. Além disso, o tempo de processamento e atualização da base de conhecimento são mais alguns fatores que interferem na resolução do problema em estudo.

[1] *Matching*, no sentido deste trabalho, é a verificação das regras que se aplicam ao estado do problema, bem como, a verificação de quais regras antecedem ou sucedem outra regra.

No caso de Sistemas de Suporte à Decisão, no Domínio Médico, em particular, as dificuldades já mencionadas são de tal envergadura que este domínio poderia ser considerado como o pior caso na construção de SE. O diagnóstico médico não é o mesmo que, por exemplo, configurar um Sistema de Computador. No segundo caso, o funcionamento do sistema é considerado conhecido, porque ele é *man made.* No outro caso, o conhecimento é parcial, como resultado de muita pesquisa, além do que, a maioria, deriva de casos particulares observados. Isto é, os sistemas de suporte à decisão médica tratam com um fenômeno natural (uma doença), não com um sistema feito pelo homem - o conhecimento no campo médico é derivado de observação de causas e efeitos [42]. Os médicos são o exemplo mais flagrante de especialistas, que apresentam dificuldades em articular o conhecimento adquirido para resolver um caso. Eles costumam utilizar muito a experiência e o *feeling* para resolver problemas, além de que aprendem a tratar com as relações de causas/efeitos, ainda, na fase de aprendizado no curso de medicina. Assim, algumas regras iniciais podem ser extraídas, mesmo que eles tenham dificuldades de articulá-las. Deste modo, tem-se um problema ao se construir um SE para suporte à decisão na área médica. Se, todavia, os especialistas não podem articular regras apropriadas, no entanto podem seguramente fornecer exemplos de suas próprias decisões, o que nos levaria ao Paradigma Conexionista da IA.

O paradigma de RNA tem sido utilizado, ultimamente, para implementar SE, como um complemento aos RBES. As RNA são feitas através de um grande número de unidades. Estas unidades têm algumas propriedades dos verdadeiros neurônios. Isto significa que cada unidade apresenta diversas entradas, sendo algumas excitatórias e outras inibitórias. Estas unidades levam em consideração os valores de todas as entradas e geram uma saída, que é função daquelas entradas. Portanto, uma rede é caracterizada pelas unidades (neurônios) e pela maneira como elas são conectadas (topologia). Algoritmos são usados para mudar os pesos das conexões (regras de aprendizado). O efeito geral destes três aspectos constitui o Paradigma de Redes Neurais da IA. Os SE implementados, desta maneira, são chamados de Sistemas Especialistas Baseados em Redes Neurais (NNES) ou Sistemas Especialistas Conexionistas [42][49].

No entanto, na construção de NNES o primeiro problema que surge é a escolha da topologia a ser utilizada. Ou seja, como mapear uma descrição (do sistema), normalmente baseada em regras, em uma RNA? Como mapear atributos de dados,

variáveis ou conceitos às unidades de entrada, conceitos ou hipóteses às unidades de saída e conceitos ou hipóteses intermediárias às unidades intermediárias? Outra dificuldade em um sistema conexionista, diz respeito a necessidade de se explicar como este sistema chegou a uma dada solução.

No que se refere as redes recursivas, uma metodologia foi proposta por F.M. de Azevedo [42]. Já no caso de redes *feedforward* (direta), diversas metodologias têm sido propostas, como é o caso de [64][76][77][122][120][134].

Aqui, propõe-se que, ao invés de usar regras visando construir um SE e depois arranjá-las em função das diferenças de erros, procurar extrair as regras que forem possíveis (do especialista de domínio) e construir, a partir destas, uma RNA. Exemplos de casos bem sucedidos, solucionados pelo especialista de domínio, seriam utilizados, então, para refinar a rede, no sentido de traduzir tanto as regras, as quais ele não foi capaz de articular, bem como, sua experiência e seu *feeling*. Desta forma, a rede original seria tratada através de um algoritmo de aprendizado que, utilizando-se dos exemplos, modificaria não só o peso de cada conexão como também, eventualmente, a estrutura da rede, incluindo ou excluindo ligações entre neurônios, e/ou incluindo ou excluindo os neurônios, propriamente ditos, da camada[2] intermediaria.

Assim, um HES é proposto para lidar com os problemas aqui relacionados. Este sistema consiste basicamente de um NNES e um RBES [33][20].

1.3 Sistemas Especialistas Híbridos

As arquiteturas híbridas, para sistemas inteligentes, são um novo campo da pesquisa de IA. Estudos recentes focalizam a integração de paradigmas de SE e RNA, ambos com valores *crisp* (abrupto) e *fuzzy*, explorando as similaridades das estruturas básicas destes dois métodos de manipulação de conhecimento, bem como, as várias aplicações nas quais os sistemas híbridos inteligentes podem ser usados para solucionar tarefas importantes.

As RNA e a lógica *fuzzy* podem ser combinadas para formarem sistemas híbridos, os quais absorvem as melhores características de ambas. Sendo assim, RNA e lógica

[2]Durante todo este trabalho quando se referenciar a palavra *camada* será tratada como sendo a *camada de neurônios*, salvo quando se especificar no texto em questão que será a de *pesos*.

fuzzy provêm dois modos complementares de modelagem da organização complexa e dicotômica do cérebro humano. Ao nível de RNA, estas suprem um meio para modelar os processos de ordem celular, isto é, a fisiologia do cérebro. Enquanto que, ao nível de lógica *fuzzy*, esta supre propriedades, ou seja, ela pode ser vista como uma modelagem psicológica da mente [104]. Além disso, para prover um mecanismo que represente o processo de raciocínio, de ordem macroscópica no cérebro, a linguagem da lógica *fuzzy* auxilia na descrição e compreensão da conduta básica da célula nervosa. Contudo, a noção básica da célula como um sistema *fuzzy*, faz com que haja consideráveis correlações entre o processo de informação no cérebro e a teoria de conjunto *fuzzy*. Deste modo, um conjunto *fuzzy* é uma representação distribuída do conhecimento. Por exemplo, um número maior que 10 pode ser visto como sendo definido por uma coleção de graus de pertinências. Entretanto, a representação do conhecimento em uma RNA é suposta ser codificada por relações estabelecidas entre uma coleção de neurônios. Outra conexão entre conjuntos *fuzzy* e RNA é o isomorfismo entre ambas as abordagens. As operações mais conhecidas em conjuntos *fuzzy* são interseção, união e complemento, estas são paralelas, enquanto as RNA, fornecem um meio ideal para estas computações.

A interpretação do significado de uma rede é uma tarefa bastante difícil e crítica, tendo em vista, a falta de uma linguagem apropriada para transcrever o conhecimento representado nas conexões entre os neurônios da RNA. A teoria de conjuntos *fuzzy* pode prover uma ferramenta para superar este problema. Como conseqüência da correlação íntima entre conjunto *fuzzy* e RNA, a lógica *fuzzy* pode fornecer uma semântica, bem como, uma sintaxe forte para auxiliar a descrever o conhecimento codificado na rede. Se uma linguagem adequada é fornecida para uma RNA, pode também ser utilizada para programá-la.

A nível sintático, ou seja, biológico, a RNA pode ser descrita de forma a mostrar um mapeamento *fuzzy*. Por exemplo, a retina - órgão sensorial do sistema visual humano - é a região fotossensível do olho que contém os cones, principais responsáveis pela visão cromática, e os bastonetes, principais responsáveis pela visão da intensidade luminosa. Além das células fotossensíveis, que aparecem na retina, há as células bipolares e as células ganglionares. As células fotossensíveis estabelecem sinapses com as células bipolares que, por sua vez, fazem sinapses com as células ganglionares, cujos axônios constituem o nervo óptico. Os prolongamentos periféricos das células fotossensíveis são os receptores da visão, cones e bastonetes. Assim, os raios

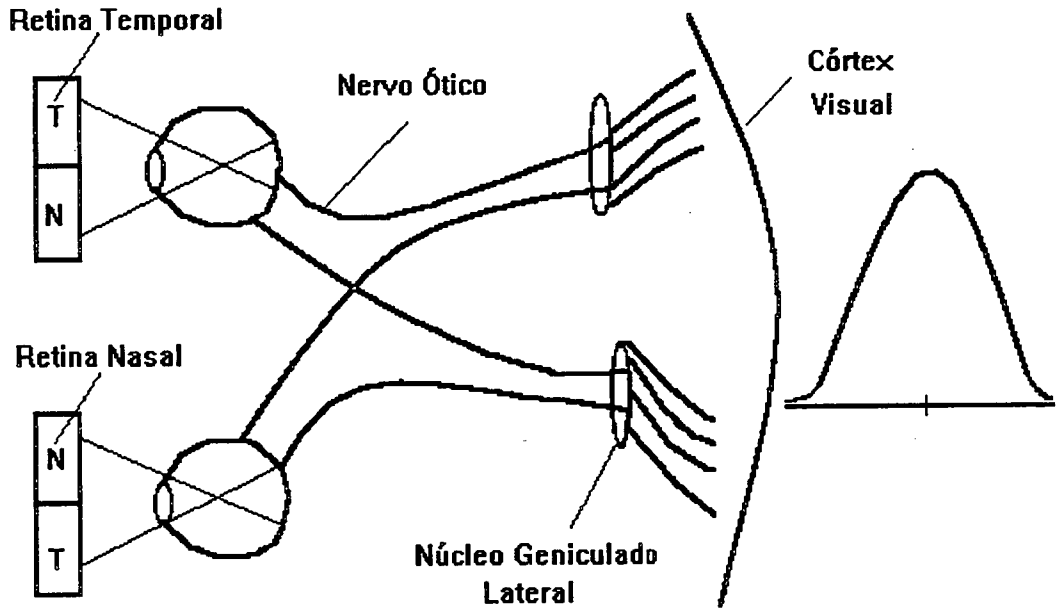

Figura 1.1: Trajetória básica do sistema visual do olho ao córtex visual

luminosos, que incidem sobre a retina, atravessam suas nove camadas internas para atingirem os fotoreceptores, cones e bastonetes. A membrana destes é sensível à luz e esta provoca a despolarização da membrana, que caminha para o soma, ou seja, das células fotossensíveis para as células bipolares, e destas para as células ganglionares, cujos axônios constituem o nervo óptico. Então, quando os cones e bastonetes são excitados, os sinais são transmitidos através de neurônios sucessivos na própria retina e, finalmente, pelas fibras do nervo óptico até o córtex cerebral [119]. Neste ponto, o cérebro faz um mapeamento do campo visual, de forma que se obtém uma forma de onda, similar a uma função de pertinência usada na lógica *fuzzy*, do sinal de entrada apresentado na retina para o córtex cerebral (Figura 1.1).

A nível semântico mostra-se que, seja na forma de função de pertinência ou operações com os conectivos *fuzzy*, há uma correspondência entre a RNA e a lógica *fuzzy*. O paradigma simbólico *fuzzy* pode ser representado nas conexões de uma RNA representando uma função de pertinência. Pode ser representado ainda, em termos das unidades de processamento (neurônios), na função de ativação substituindo, por exemplo, a função sigmóide, como também, na operação de confluência, onde, dependendo do tipo de neurônio pode ser substituído por um dos operadores de agregação da lógica *fuzzy*. Assim, pode-se observar que há uma equivalência

semântica entre estes dois paradigmas, conforme é mostrado na Figura 1.2.

Desta forma, com os princípios anteriormente mencionados entre os paradigmas simbólicos e sub-simbólicos, são descritos a seguir, sucintamente, alguns HES mais significativos nesta área de pesquisa.

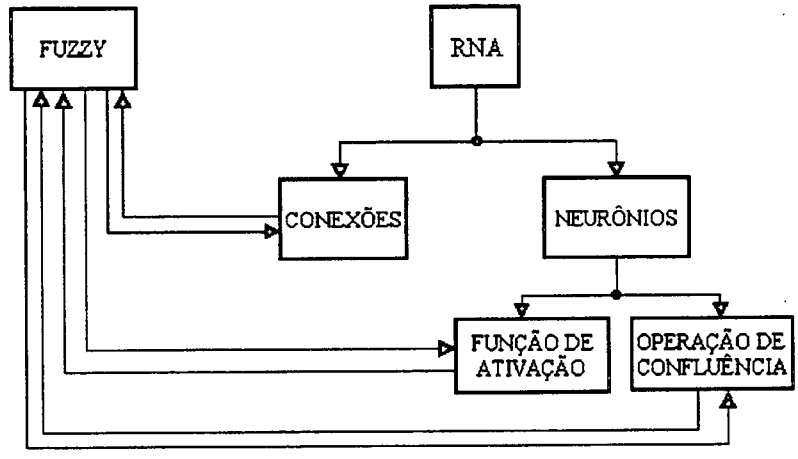

Figura 1.2: Nível semântico de uma rede *neuro-fuzzy*

1.3.1 HES com Abordagem de L.M. Fu

L. M. Fu [58][66][59][61][60][62][63][64], propôs um modelo híbrido, no qual traduz um RBES em uma RNA juntamente com um conjunto de exemplos, por mapear a base de conhecimento e a máquina de inferência em um tipo de RNA, referenciada como Rede Neural Conceitual Baseada no Conhecimento (*Knowledge Based Conceptional Neural Network* - KBCNN). Sob tal prisma, um conceito de domínio ou um valor de atributo, na base de conhecimento, é mapeado em um nó da KBCNN. Uma regra é implementada por um grupo de nós e conexões, no qual é acompanhada pelo uso de uma das ferramentas de tratamento de incertezas. Recomendados pela literatura, os Fatores de Certeza (FC), têm como objetivo o de aproximar-se com um certo grau de similaridade ao da tomada de decisão humana. Porém, antes da montagem da KBCNN, propriadamente dita, executam-se duas tarefas: a primeira, é o uso de uma rede de inferência, ou seja, um grafo E/OU, e a segunda é a colocação das regras na forma de cláusulas de Horn.

A rede de inferência serve como uma etapa intermediária do processo de AC para a montagem do KBCNN. Esta rede permite saber quantos neurônios são possíveis, tanto para a camada de entrada, quanto para as camadas de saída e intermediária desta rede (Figura 1.3).

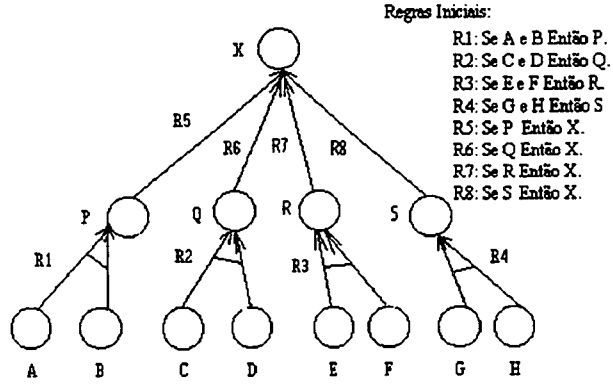

Figura 1.3: Grafo E/OU

As regras obtidas no processo de AC são usadas no formato de cláusulas de Horn, porque se deseja, exemplificando, para o caso de diagnóstico médico, apenas uma saída, ou seja, somente uma conclusão. Por exemplo, caso se tenha obtido a seguinte regra:

Se p *Então* q *E* r.

Como não está no formato desejado, ela é transformada para a seguinte forma:

Se p *Então* q e *Se* p *Então* r.

A arquitetura da rede é caracterizada por alternar camadas conjuntivas (E) e disjuntivas (OU), uma maneira pela qual preserva-se muito das semânticas do sistema original (Figura 1.4). A conduta de inferência é caracterizada por propagar e combinar ativações recursivamente através da rede e, pode envolver pesquisa iterativa para um estado estável. A conduta de aprendizado é baseada em um mecanismo conhecido como retropropagação, o qual tem mostrado ser efetivo na construção de KBCNN. Em contraste a outras RNA baseadas em conhecimento, a KBCNN provê

conexão bidirecional entre RNA e RBES. De um modo geral, um RBES pode ser mapeado em uma RNA. Por outro lado, o conhecimento da RNA pode ser transferido para trás para o RBES. Desta forma, este é considerado como um tipo de modelo conexionista baseado em regras, incorporando aprendizado específico e algoritmos de processamento de informações.

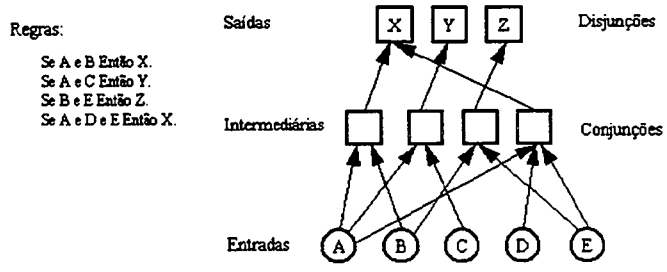

Figura 1.4: Uma rede conexionista baseada em regras

O algoritmo de aprendizado, usado para este tipo de RNA, é desenvolvido da seguinte forma:

1. Os pesos são inicializados com a força das regras, caso elas sejam existentes e fortes assumindo o valor de 0,5 e caso sejam existentes e fracas serão 0,1. Porém, para as regras inexistente os pesos passam a ser -0,1;

2. A função de ativação para a KBCNN é dada pelo modelo do Fator de Certeza (FC), isto é:

$$f(X,Y) = X + Y - XY$$

   Assim, formalmente, é definida uma função de ativação F baseada no modelo FC, como a seguir:

$$F(x_1, x_2, \ldots, y_1, y_2, \ldots) = F^+(x_1, x_2, \ldots) + F^-(y_1, y_2, \ldots) \qquad (1.1)$$

3. O procedimento de aprendizado para a KBCNN é dado por:

   a - Retropropagação adaptada para o modelo da KBCNN;

   Nesta etapa, a RNA é treinada com o algoritmo de retropropagação, porém, com algumas modificações, como:

- Tratamento das variáveis de entrada utilizando FC;
- Número de neurônios em cada camada da RNA determinado pelos grafos de inferência;
- Uso de pesos aleatórios para ambas as camadas da RNA.

b - Transformação esparsa da RNA treinada por anular os pesos pequenos;

- Os pesos da entrada para cada unidade intermediária são simplificados de tal forma que, cerca de 30% dos pesos maiores, são guardados, enquanto que, o restante dos pesos são anulados.

c - Aglomerados (*Clustering*) de unidades intermediárias;

- Iniciar com nenhum aglomerado;
- Escolher uma unidade intermediária para um aglomerado novo;
- Para cada uma das unidades intermediárias restantes achar o aglomerado mais similar e, daí, se a similaridade entre o vetor de peso da unidade e o vetor de peso médio do aglomerado é maior do que um valor limite, por exemplo, 0,85, então, atualiza-se o último vetor com o primeiro vetor;
- Caso contrário, fixa-se a unidade intermediária para um novo aglomerado e, inicializa-se o vetor de peso médio do aglomerado com o vetor de peso da unidade em questão.

d - Retropropagação

- Nesta etapa, uma nova unidade intermediária, com o vetor de peso de entrada médio e o vetor de peso de saída médio, substituirá as unidades intermediárias, no mesmo aglomerado dentro de uma camada;
- A operação de aglomerados é seguida por uma segunda aplicação do algoritmo de retropropagação, o qual, completa o processo de treinamento.

4. Ajuste de pesos no algoritmo de retropropagação:

$$W_{ji}(n+1) = W_{ji}(n) + \Delta W_{ji}$$

$$\Delta W_{ji} = \eta D_j \left(\frac{\partial O_i}{\partial W_{ji}} \right) - \zeta W_{ji}(n)$$

Onde:

W_{ji} = peso da unidade j para a unidade i
η = taxa de aprendizado
n = número de iterações
ζ = parâmetro de decréscimo do peso
D_j = erro na saída da rede

5. O erro para a camada de saída é:

$$D_j = T_j - O_j$$

Onde:

T_j = ativação da saída desejada na unidade de saída j
O_j = ativação da saída real na unidade de saída j

6. O erro para as camadas intermediárias são:

$$D_j = \sum_k \left(\frac{dO_k}{dO_j}\right) D_k$$

7. O critério de parada é dado pela média do erro médio quadrático sobre as unidades da saída ($< 0,02$) ou, converge para um valor assintótico com flutuações pequenas ($< 0,001$).

1.3.2 HES com Abordagem de R. Sun

A abordagem dada por R. Sun[3] [176][174][173][175], usa um novo formalismo de RNA, o modelo *Discrete Neural Model/ Probabilistic Discrete Neural Model* (DN/PDN). Ele desenvolveu um modelo paralelo compacto de raciocínio aproximado, baseado em regra, chamado de *Connectionist Rule-Based Reasoner* (CRBR).

O modelo DN, é descrito pela Teoria de Automata, da seguinte forma:

$$W = \{N, M\}$$

Onde:

[3]De Azevedo e Barreto desenvolveram um formalismo para RNA baseado na Teoria de Sistema, estudo este começado em 1989 [46][48][44][45][47][42][43]. Além disso, esta utilização de Teoria de Sistema para formalização de RNA tem sido muito comum no Grupo de Pesquisas em Engenharia Biomédica (GPEB) [165][158].

N = {S,O,A,I,B,T,C}, conjunto de neurônios da rede[4]
S = conjunto de todos os estados possíveis de um neurônio
O = conjunto de todas as saídas
A = conjunto de todas as ações (símbolos de saídas)
I = conjunto de entradas
B = conjunto de todos os símbolos de entradas
T = função de transição de estado: $S \times I \rightarrow S$
C = função de ação (função de saída): $S \times I \rightarrow O$
M = conectividade entre neurônios no conjunto N: M = {(i,j,k)}, onde i e j são índices de neurônios e, k é um *label* único para uma sinapse particular.

Desta forma, os neurônios discretos podem ser implementados com uma RNA típica, bem como, utilizando um algoritmo de aprendizado para a rede, similar ao algoritmo de retropropagação clássico. Isto quer dizer que uma rede multicamada pode fazer este tipo de trabalho, isto é: a camada intermediária representa o estado corrente e, o valor entrada/saída é representado explicitamente por um nó individual. A saída da camada intermediária é direcionada para trás no sentido que a camada de entrada possa ajudar a decidir, em conjunto com as entradas correntes, qual estado será a próxima entrada (Figura 1.5).

O modelo PDN é a extensão simples do modelo DN, onde ele supõe que cada entrada é uma distribuição probabilística, ao invés de ser determinística. Formalmente, ele é definido como:

$$W = \{N, M\}$$

Onde:

N = {S,A,B,I,O,T,C,P1,P2}, conjunto de neurônios
S = conjunto de todos os estados possíveis de um neurônio
A = conjunto de todas as ações (símbolos de saídas)
B = conjunto de todos os símbolos de entradas
I = entradas
O = saídas
T = função de transição de estado: $S \times I \rightarrow S$; P1

[4]Nota-se neste conjunto que foram usados chaves e não a representação matemática como sendo $< \cdots >$ pois em automata se usa as chaves.

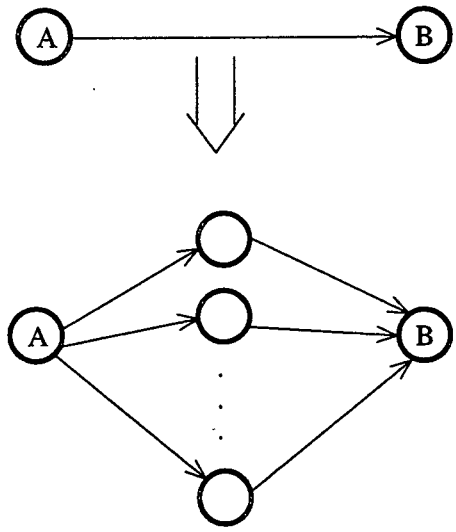

Figura 1.5: Substituindo valores de saídas múltiplas com nós intermediários múltiplos

C = função de ação (função de saída): $S \times I \rightarrow O$; P2

M = conectividade entre neurônios em N

P1, P2 = conjuntos de probabilidades associadas com cada transição de estado ou cada ação.

Assim, o modelo proposto é mapeado da seguinte forma: um conceito (predicado) é representado por uma montagem de nós DN e, é codificado em uma rede por conexões conectadas a outros nós. Estas conexões unidirecionais auxiliam a formar o conceito, bem como, a guiar o raciocínio. A função de ativação é função de valores, que representam o nível de ativação, juntamente com outra variável que indica as saídas de outros nós DN em outras montagens, pertencentes ao conjunto de símbolos de saída, no nó em questão. Além disso, esta variável também é função da dimensão temporal de padrões de ativação. Este modelo consiste de três camadas de nós DN: a camada de entrada, a de processamento e a de saída, as quais são conectadas entre si, sendo que cada uma pode manter o aprendizado e o pré e pós-processamento correspondentemente. A primeira camada, é onde os padrões de entrada são armazenados. A segunda, representa os fatos, na parte condicional de uma regra, que é conectada à montagem representando as premissas da mesma e são conectados com nós correspondentes em outras montagens. A terceira, é a saída do

modelo, a qual nos dá a conclusão do problema a ser abordado, ou seja, a parte do conseqüente de uma regra. O modo no qual este sistema desempenha o raciocínio é o chamado *data-driven*[5].

Este sistema incorpora um esquema para o raciocínio aproximado FEL (Lógica Evidencial *Fuzzy*), que é baseado na teoria da lógica *fuzzy*. Assim, o sistema pode lidar com informação parcial, incerta e evidência acumulativa. A Figura 1.6 representa um exemplo para o modelo DN/PDN proposto por R. Sun.

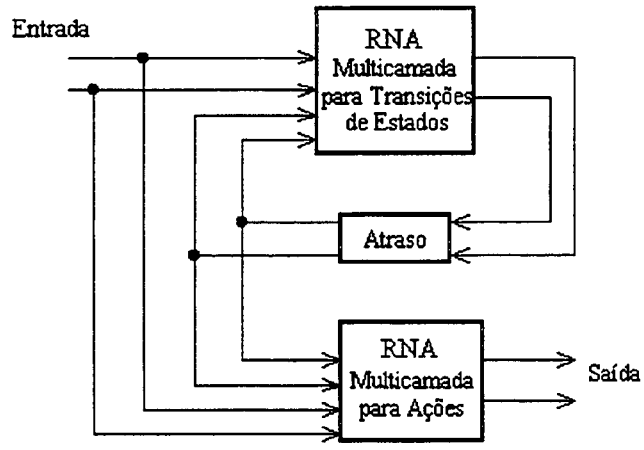

Figura 1.6: Implementando um automata de estado finito em RNA

1.3.3 HES com Abordagem de R. Machado

A abordagem dada por R. Machado [121][123][122][120], é baseada em um modelo onde o processo de AC é representado por grafos de conhecimento - que é um método, onde os especialistas expressam o seu conhecimento sobre cada hipótese do domínio do problema - selecionando um conjunto de evidências apropriados e construindo um grafo acíclico E/OU, dessas evidências, para a hipótese específica. Estes grafos são transformados em RNA na Base de Conhecimento Conexionista (BCC), automaticamente, por um algoritmo de conversão. A Figura 1.7 mostra o modelo proposto por R. Machado.

[5]É um dos métodos de busca de solução, também conhecido como encadeamento para frente ou *Forward Chaining* [68][154].

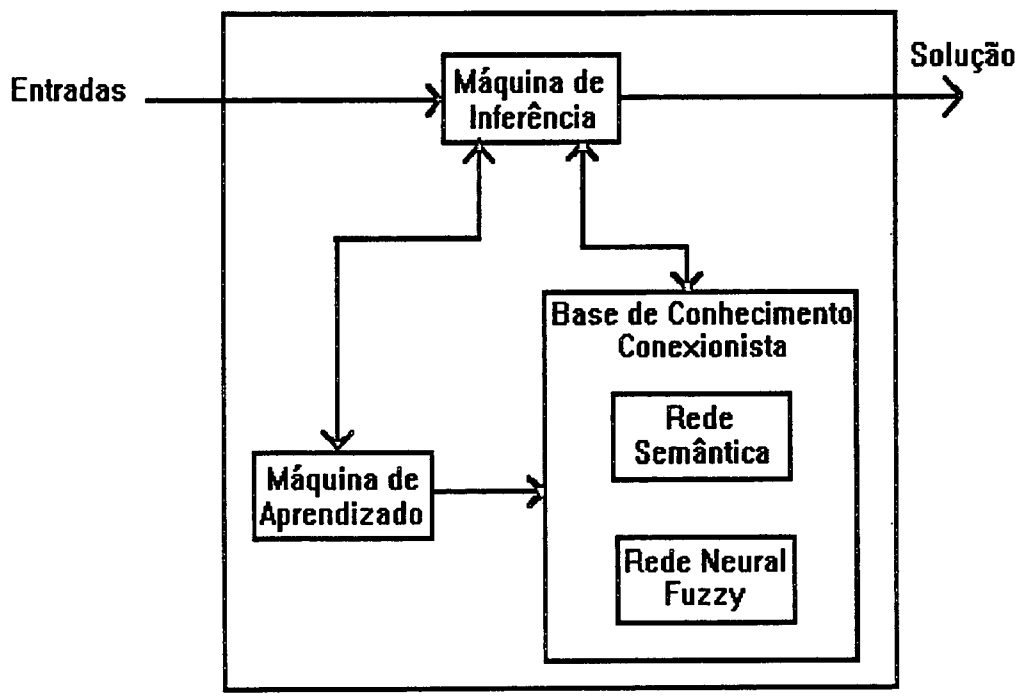

Figura 1.7: SE conexionista *fuzzy*

O SE conexionista *fuzzy* é composto de diversos módulos com funções específicas, resultando em um sistema bastante flexível. Os principais módulos são: BCC, máquina de inferência e máquina de aprendizado.

A BCC é composta de duas partes: rede semântica e RNA. A rede semântica é usada para representar os conceitos do domínio do problema (objetos) e suas relações, enquanto que, para a RNA, é adotado o Modelo Neural Combinacional (MNC), que é um modelo neural de ordem superior, o qual usa a lógica *fuzzy* para tarefas classificatórias [122][120]. Este modelo é composto de três ou mais camadas: a camada de entrada para evidências, as camadas intermediárias para abstrações intermediárias e a camada de saída para hipóteses. Após o treinamento da RNA é possível abstrair-se facilmente regras desta rede.

A máquina de inferência gerencia a consulta feita pelo usuário, trabalhando em dois níveis: rede semântica e RNA. Ao nível da rede semântica, a máquina de inferência analisa os objetivos da consulta e seleciona a melhor seqüência de raciocínio.

Ao nível da RNA, a máquina de inferência realiza a tarefa de classificação, podendo também, por um processo de encadeamento para trás (*backward*), decidir se efetua perguntas ao usuário, com o objetivo de obter mais informações que levem a uma solução, sendo ainda, capaz de explicar ao usuário o raciocínio utilizado.

A máquina de aprendizado tem por objetivo refinar o conhecimento do sistema, atuando iterativamente sobre a intensidade das conexões ou alterando a plasticidade das redes, usando para tanto, um conjunto de exemplos. A máquina de aprendizado seleciona e constrói, indutivamente, unidades intermediárias nas camadas intermediárias das RNA, para representar padrões importantes do domínio do problema. A BCC pode ser refinada incrementalmente através de dois processos:

- Ajuste dos pesos sinápticos, utilizando-se o algoritimo de punição-recompensa [121][123], o qual se baseia na lei de Hebb. Este algoritmo modifica os pesos das sinapses, forçando a rede a convergir para o comportamento desejado, expresso através de um conjunto de exemplos;
- Mudando-se a topologia da RNA através de Algoritmos Genéticos (AG). Usa-se os Operadores Genéticos (OG), mutação e *crossover*, para criar novos elementos nas camadas intermediárias.

Assim, esta arquitetura híbrida permite a construção de sistemas especialistas conexionistas *fuzzy*, capazes de herdarem propriedades desejáveis de ambos os paradigmas, neural e simbólico, tais como: representação de conhecimento do especialista, integração de fontes de conhecimento de múltiplos especialistas, aprendizado heurístico de exemplos, aprendizado incremental, tratamento de imprecisão e dados de entrada parciais e explicação do raciocínio, entre outras.

1.3.4 HES com Abordagem de M. M. Gupta

M. M. Gupta [76][77], desenvolveu uma arquitetura neural *fuzzy*, baseada na noção dos conectivos T-normas e T-conormas. O princípio básico deste sistema consiste no desenvolvimento de um neurônio *fuzzy*, baseado nas morfologias neuronais biológicas, seguidas pelos mecanismos de aprendizado. A sua RNA possui a topologia do tipo direta estática, com múltiplas camadas e com neurônios *fuzzy* do tipo OU. Um exemplo deste tipo de rede é mostrado na Figura 1.8.

Os sinais das entradas das RNA são expressos em termos da função de pertinência, considerando o intervalo [0,1]. As operações matemáticas utilizadas para

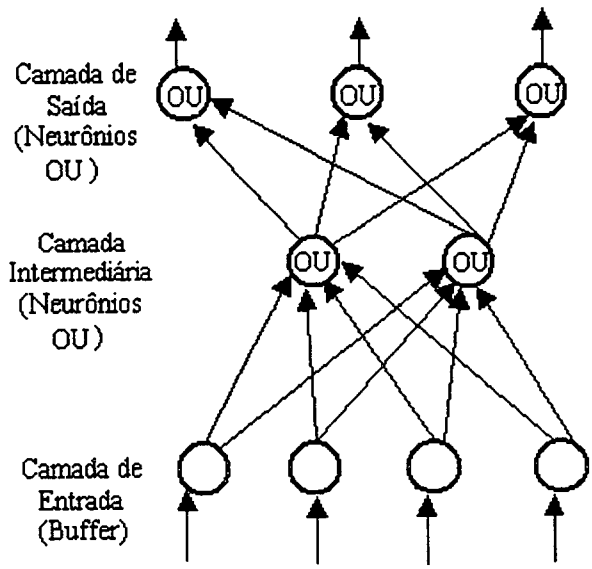

Figura 1.8: RNA com neurônios OU

modelar os sinais da rede são por meio da lógica *fuzzy*, obtidos através dos operadores Max/Min. Além disso, a operação do produto escalar, entre as entradas da rede pelo seus pesos, foi substituída pela operação T e a operação somatório pela operador S. Quanto a regra de aprendizado, adotada neste modelo, foi baseada no algoritmo de retropropagação para uma rede supervisionada, enquanto que, para uma rede não-supervisionada foi adotada a regra de aprendizado de Hebb. Ambas foram adaptadas para o seu modelo. A Figura 1.9 mostra dois modelos de sistemas neurais para uma rede supervisonada.

O formalismo matemático, usado por M.M. Gupta [75][76][77], para o cálculo do ajuste de erro, para a sua arquitetura neural, é dado por:

$$y(t) = \Psi[u(t)] \, \epsilon \, [0, 1]$$

$$\Psi[u(t)] = \tanh[u(t)]$$

Onde:

y(t) = saída da rede
u(t) = saída de um neurônio antes de passar pela função de ativação
Então, para neurônios OU:

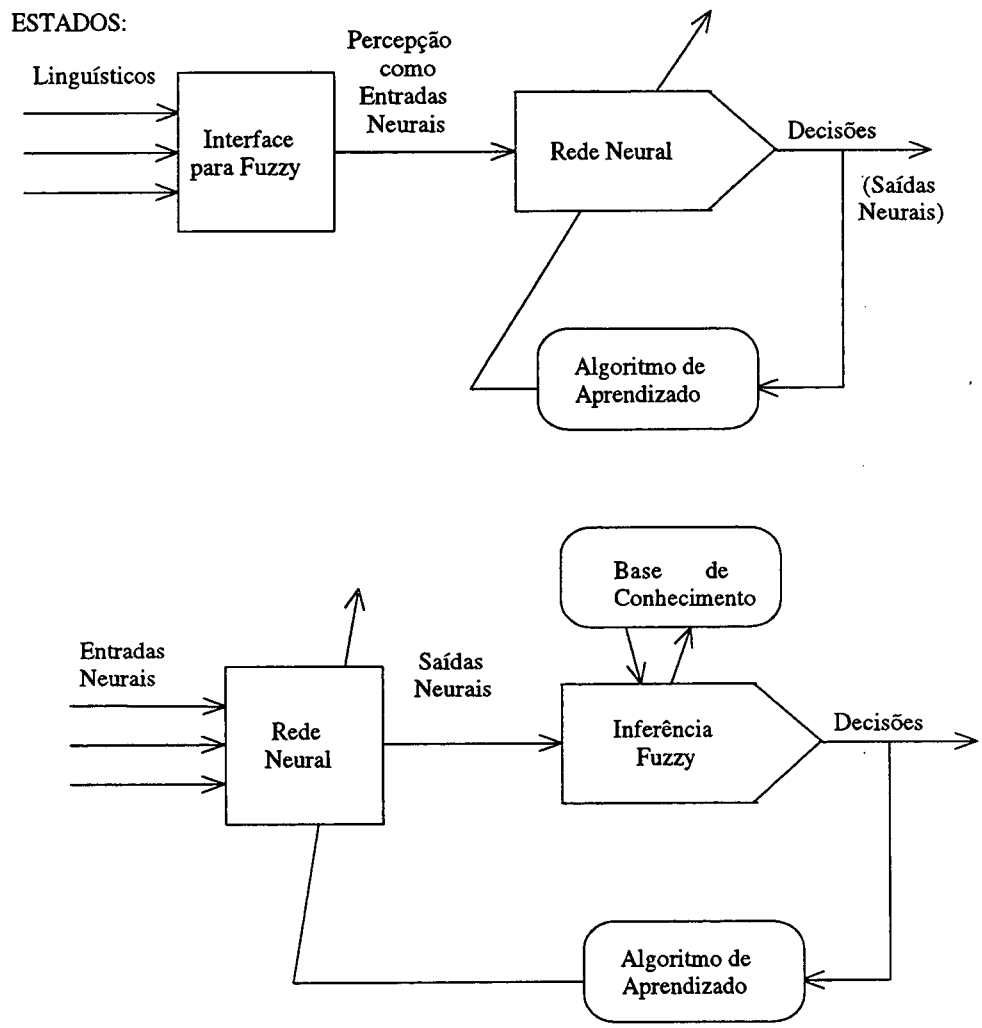

Figura 1.9: Dois modelos de sistemas neurais *fuzzy* usando a topologia direta

$$u(t) = S_{j=0}^{n}[w_i(t)\ T\ x_i(t)]\ \epsilon\ [0,1]$$

Onde:

$w_i(t)$ = forças das conexões sinápticas (pesos) em relação as entradas
$x_i(t)$ = entradas da rede

Assim, o erro na saída da rede, e(t), é definido como:

$$e(t) = [y_d(t) - y(t)]\ \epsilon\ [-1,1]$$

Onde:

$y_d(t)$ = saída desejada da rede
$y(t)$ = saída real da rede

Então, para um caso geral, usando-se o algoritmo de retropropagação modificado, tem-se que:

$$\Delta w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$\Delta w_i(t) = \eta x_i(t)[y_d(t) - y(t)]$$

Onde:

$w_i(t)$ = peso sináptico correspondente à entrada $x_i(t)$
$\Delta w_i(t)$ = mudança na conexão sináptica $w_i(t)$ através de um instante de tempo
η = taxa de aprendizado

Assim, para o caso dos neurônios OU, tem-se que:

$$\Delta w_i(t+1) = w_i(t)\ OU\ \Delta w_i(t)$$

$$\Delta w_i(t+1) = S[w_i(t), \Delta w_i(t)]$$

$$\Delta w_i(t+1) = max[w_i(t), \Delta w_i(t)]$$

Para o caso dos neurônios E, tem-se que:

$$\Delta w_i(t) = x_i(t)\ E\ e(t)$$

$$\Delta w_i(t) = T[x_i(t), e(t)]$$

$$\Delta w_i(t) = min[x_i(t), e(t)]$$

1.3.5 HES com Abordagem de S. Mitra e S.K. Pal

O modelo de S. Mitra e S.K. Pal [134][135][145], lida com uma RNA *fuzzy* para classificação e geração de regra, usando saídas binárias. O paradigma conexionista compreende: topologia direta, aprendizado supervisionado por algoritmo de retropropagação modificado e, imprecisões nas entradas e saídas da rede modelada, por meio de funções de pertinências. Além disso, este modelo desempenha duas tarefas: a construção da rede lógica *fuzzy* e da rede treinada, que são usadas para gerar regras.

A construção da rede lógica *fuzzy* tem como meta classificar os padrões multiclasses. O modelo proposto foi exemplificado para uma rede de três camadas, sendo que, a camada de entrada é referenciada apenas como um *buffer*, a camada intermediária como neurônios E e a camada de saída por neurônios OU (Figura 1.10). Os operadores lógicos, nomeados como norma T e T-conorma S, envolvem os neurônios E e OU, os quais, são empregados no lugar do somatório de pesos e funções sigmoidais. Vários operadores de implicação *fuzzy* são introduzidos para incorporar quantidades diferentes de interação mútua, durante o algoritmo de retropropagação de erros.

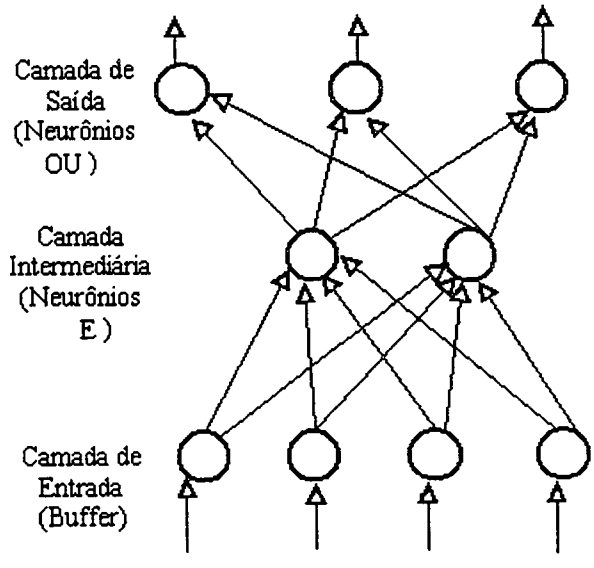

Figura 1.10: Uma RNA de três camadas com funções lógicas E e OU

Após a rede ser treinada, a RNA é usada para gerar regras. Contudo, para o objetivo de geração de regra e inferência, no caso de entradas parciais, este modelo é capaz de inquirir o usuário sobre características essenciais para uma dada informação, cuja justificativa é dada ao se atingir uma decisão, através da geração

na forma de regra. Estas regras são expressas como a disjunção de cláusulas conjuntivas (Figura 1.11). Por fim, a efetividade do modelo é testada em um problema de reconhecimento de voz e por conjuntos de padrões gerados artificialmente.

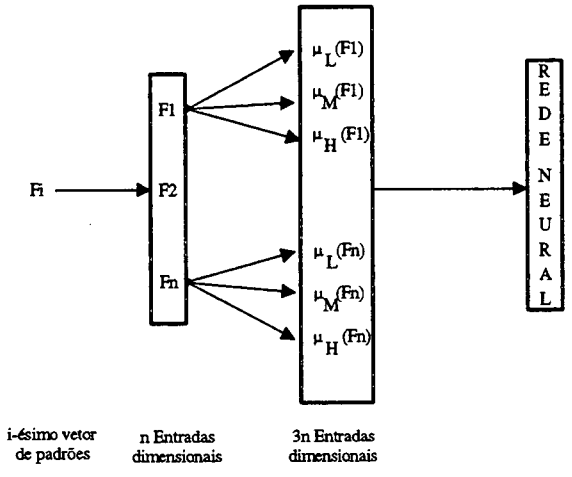

Figura 1.11: Diagrama de bloco da fase de entrada do modelo proposto

Onde:

$$\vec{F_i} = [F_{i1}, F_{i2}, \cdots, F_{in}]$$

$$\vec{F_i} = [\mu_{low_{(F_{i1})}}(\vec{F_i}), \mu_{medium_{(F_{i1})}}(\vec{F_i}), \mu_{high_{(F_{i1})}}(\vec{F_i}), \cdots, \mu_{high_{(F_{in})}}(\vec{F_i})] \tag{1.2}$$

1.3.6 HES com Abordagem de W. Pedrycz

W. Pedrycz [149][151][150], sugere dois tipos de arquiteturas híbridas para RNA *fuzzy*. Elas têm por objetivo realizar uma importante sinergia entre as RNA e os conjuntos *fuzzy*, para gerar modelos com esquemas articulados, explicitamente de representação de conhecimento. Estas redes são equipadas com capacidades de aprendizado substanciais e mecanismos eficientes de gerenciamento de imprecisões. Assim, uma das arquiteturas utiliza neurônios *fuzzy* E/OU, de modo que o neurônio do tipo E, é colocado na camada intermediária e o neurônio do tipo OU, é posto na camada de saída da RNA (Figura 1.12).

Onde:

Para os neurônios OU, tem-se que:

$$Y = OU(x; w)$$

$$Y = S_{j=1}^{n}[x_i \, T \, w_i]$$

Para os neurônios E, tem-se que:

$$Y = E(x; w)$$

$$Y = T_{j=1}^{n}[x_i \, T \, w_i]$$

Onde x_i, w_i, S e T têm os mesmos significados da abordagem de M.M. Gupta.

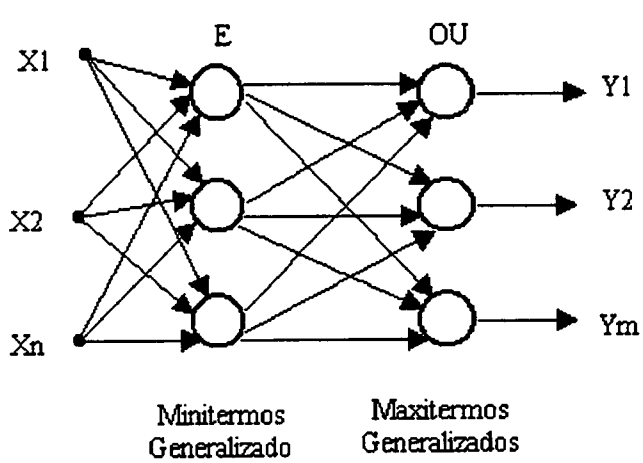

Figura 1.12: Arquitetura do neurônio E/OU

Outra arquitetura, diz respeito aos neurônios *fuzzy* OU/E, que são construídos de forma a colocar vários neurônios E e OU, em uma estrutura simples de duas camadas, de tal maneira que, para a primeira camada são misturados os neurônios

OU/E e, na outra camada são usados somente neurônios OU (Figura 1.13).

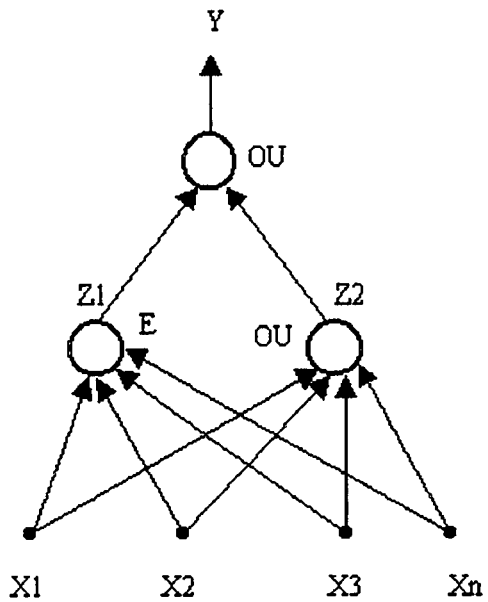

Figura 1.13: Arquitetura do neurônio OU/E

Onde:

$$Y = OU/E(x; w, \lambda, \mu)$$

Com:

μ = taxa de aprendizado
λ = valores das conexões

Então:

$$Y = OU([Z_1, Z_2]; \nu)$$

$$Z_1 = E(x; w_1)$$

$$Z_2 = OU(x; w_2)$$

$$\nu = [\lambda_1, \mu_2]$$

$$w_i = [w_{i1}, w_{i2}, \cdots, w_{in}]$$

Com:

$$i = 1, 2, \cdots, n$$

Para ambos os casos anteriormente mencionados, são usados os operadores lógicos Max/Min, da lógica *fuzzy* tradicional, para a tarefa de modificação dos pesos da referida rede. O algoritmo de aprendizado é sugerido como o de retropropagação modificado, com a melhoria da ferramenta conhecida como AG para otimização da RNA, no sentido do aprendizado da rede. O autor sugere que ele seja aplicado no desenvolvimento de *Flip-Flops JK fuzzy*, controladores *fuzzy* e redes de Petri *fuzzy*[149][151][150].

1.3.7 HES com Abordagem de H.C. Fu e J.J. Shann

H.C. Fu e J.J. Shann [57], apresentam uma RNA *fuzzy* para aprender o conhecimento de um sistema baseado em regras lógicas *fuzzy*. Esta rede contém cinco camadas, isto é, a camada de entrada, a de função de pertinência, a E, a OU e, por último, a de *defuzzification* (Figura 1.14).

O algoritmo de aprendizado, proposto por eles, se constitui numa mescla do de retropropagação e do de aprendizado tipo competitivo. Este algoritmo serve para treinar esta rede, de tal forma a adquirir as regras *fuzzy* e refinar o conhecimento por meio dos nós E/OU. Os autores creêm que a rede proposta adquire conhecimento mais preciso e muito mais rápido, durante o aprendizado de conhecimento. A justificativa assim se apresenta: a rede é parcialmente conectada (esparsa) e os ajustes dos pesos aprendidos na camada OU, isto é, os pesos das regras *fuzzy*, tem algumas características do aprendizado competitivo, implicando que o tempo de convergência para o algoritmo de aprendizado proposto é muito mais rápido do que para o algoritmo de retropropagação clássico - isto é, para uma RNA direta completamente conectada. Em resumo, os pesos atualizados são ajustados muito mais rapidamente para obter um conhecimento mais preciso. O sistema neural foi treinado para dois exemplos de sistemas de controladores adaptativos, onde em um deles foi aplicado o algoritmo de retropropagação clássico e no outro o algoritmo competitivo diferencial.

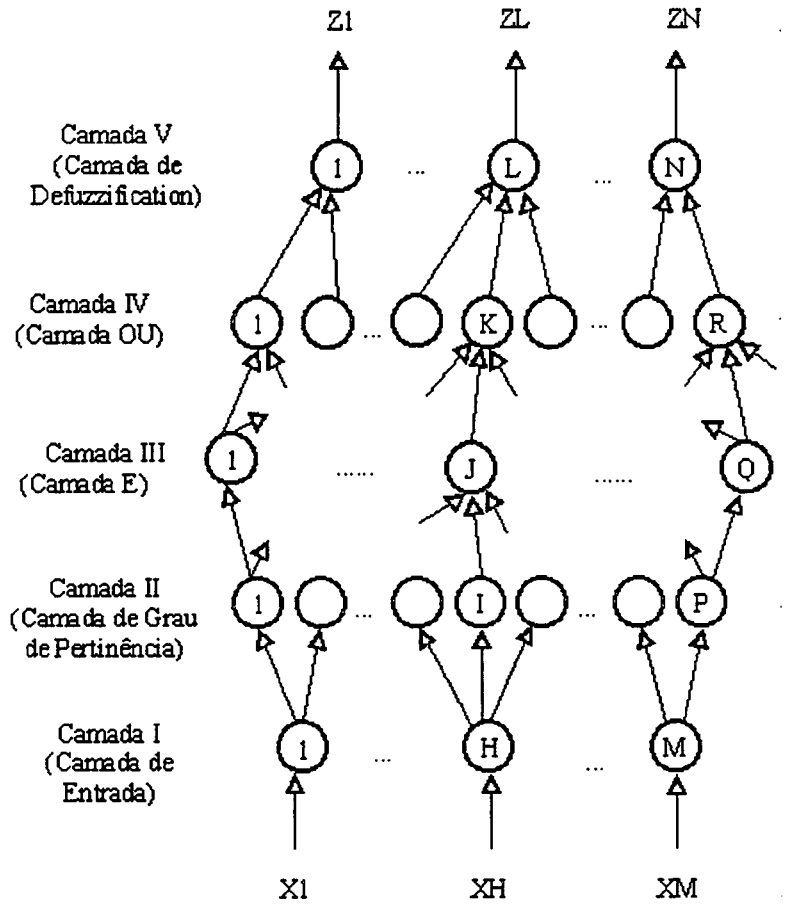

Figura 1.14: A estrutura da RNA *fuzzy*

1.3.8 HES com Abordagem de Y. Hayashi

Y. Hayashi [82][83][84][85][138], propos um sistema especialista neural *fuzzy* juntamente com um método para extrair regras *fuzzy Se/Então* automaticamente de uma RNA treinada. Esta rede é do tipo direta, cuja camada de entrada, consiste de grupos de células *fuzzy* e de células abruptas. Desta forma, a veracidade de informação *fuzzy* e *abrupta* de dados treinados é representada por grupos de células *fuzzy* e *abruptas*, respectivamente (Figura 1.15). O SE tem como funções principais: a generalização da informação derivada dos dados treinados, a representação do conhecimento na forma da RNA *fuzzy* e a extração de regras *fuzzy Se/Então*, com importância linguística de cada proposição no antecedente (parte *Se*) de uma RNA *fuzzy* treinada. A validade e efetividade da rede em questão foi realizada para um caso específico de diagnóstico médico.

1.3.9 HES com Abordagem de S.K. Halgamuge e M. Glesner

S.K. Halgamuge e M. Glesner [79][80][138] propuseram um modelo *neuro-fuzzy*, FuNe-I, que é baseado na arquitetura de uma RNA direta, conforme a Figura 1.16. Esta rede possui cinco camadas. A primeira camada contém uma unidade para cada variável de entrada e propaga os valores de entrada não modificados via as conexões (pesos) à segunda camada. Esta camada, consiste de unidades com funções de ativação sigmóide, que são usadas para criar funções de pertinências. A terceira camada contém unidades especializadas que são somente usadas para representar conjuntos *fuzzy*. As unidades da segunda e terceira camadas propagam suas ativações via as ligações não ajustadas para a quarta camada (Veja a Figura 1.16). As unidades da segunda camada têm conexões para a da terceira, que não são conectadas à da quarta.

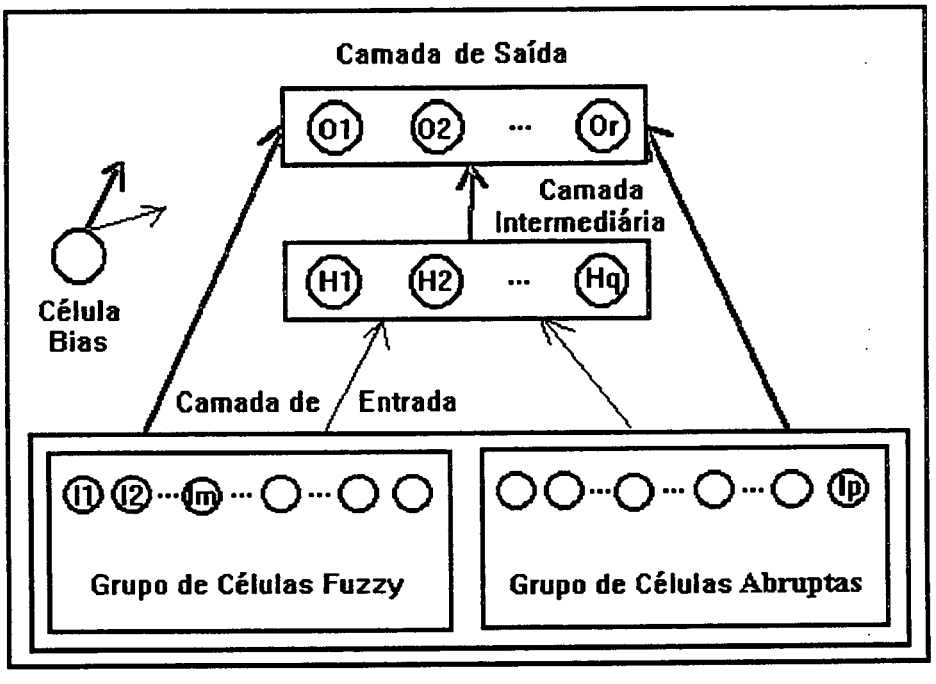

Figura 1.15: Uma RNA com grupos de células *abruptas* e *fuzzy*

A quarta camada consiste de unidades que representam regras *fuzzy*. Comparadas a outras abordagens *neuro-fuzzy*, o modelo FuNe-I é especial porque usa dois

tipos de regras: as antecedentes, que podem ser conjuntivas ou disjuntivas, e as regras com somente uma variável como antecedente (regras simples). Uma unidade computa as suas ativações, dependendo do tipo de regra que ela representa, por uma função *soft minimum* (conjunção ou *softmin* (1.3)), *soft maximum* (disjunção ou *softmax* (1.4)) ou uma função identidade.

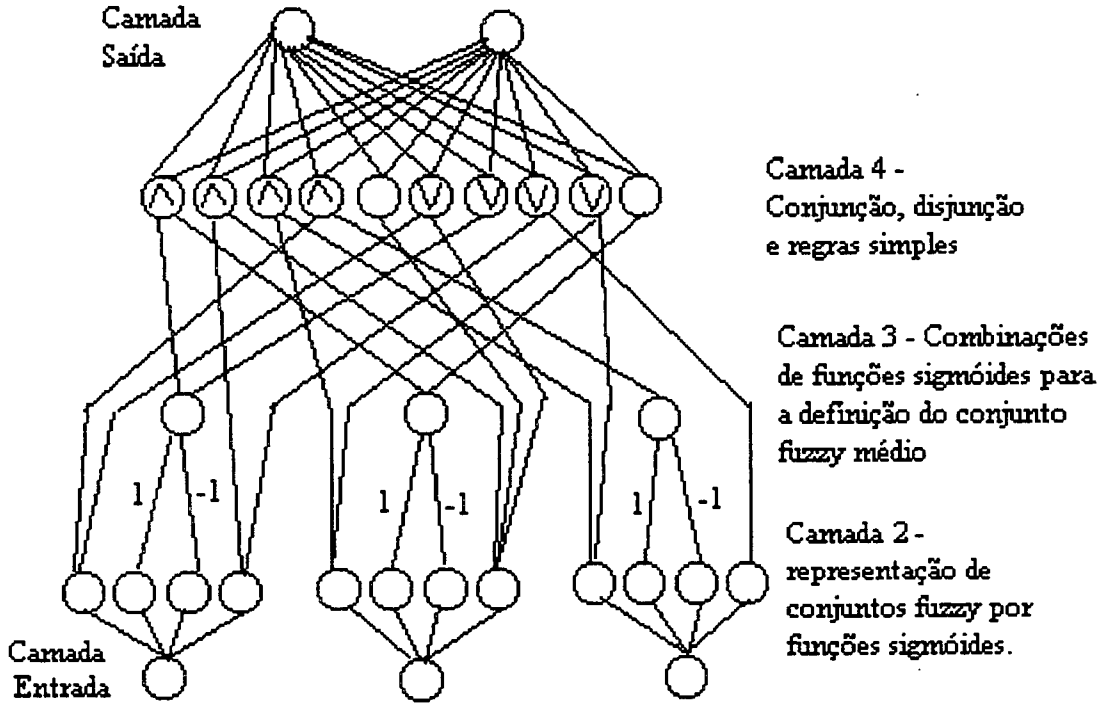

Figura 1.16: A arquitetura de um sistema FuNe-I

Então:

$$\widetilde{min}\{x_1, \cdots, x_n\} = \frac{\sum_{i=1}^{n} x_i e^{-\alpha x_i}}{\sum_{i=1}^{n} e^{-\alpha x_i}} \tag{1.3}$$

$$\widetilde{max}\{x_1, \cdots, x_n\} = \frac{\sum_{i=1}^{n} x_i e^{\alpha x_i}}{\sum_{i=1}^{n} e^{\alpha x_i}} \tag{1.4}$$

Onde α determina o comportamento da função *softmax* ou *softmin* e x_i corresponde as entradas da rede.

O parâmetro α da função *softmax* (ou *softmin*) é usado com um valor pré-fixado. Para $\alpha = 0$ se obtém a média aritmética e $\alpha \to \infty$ resulta na função máxima, propriadamente dita.

A quinta camada contém as unidades de saída da RNA, as quais computam suas entradas por uma soma de pesos e, suas ativações, por uma função sigmóide.

O sistema FuNe-I considera que cada variável é particionada por três conjuntos *fuzzy* que são nomeados de *pequeno*, *médio* e *grande*. As funções de pertinências são criadas por dois tipos de funções sigmóides:

$$s^{(direita)}_{\alpha_r,\beta_r}(x) = \frac{1}{1 + e^{-\alpha_r(x-\beta_r)}} \tag{1.5}$$

$$s^{(esquerda)}_{\alpha_l,\beta_l}(x) = \frac{1}{1 + e^{\alpha_l(x-\beta_l)}} \tag{1.6}$$

Estas são funções crescentes (1.5) ou decrescentes (1.6). Os parâmetros α e β determinam suas inclinações e posições. Então, os três conjuntos *fuzzy* mencionados, anteriormentes, podem ser representados como:

$$\mu^{(pequeno)}_{\alpha_k,\beta_k} = s^{(esquerda)}_{\alpha_k,\beta_k} \quad . \tag{1.7}$$

$$\mu^{(grande)}_{\alpha_g,\beta_g} = \mu^{(direita)}_{\alpha_k,\beta_k} \tag{1.8}$$

$$\mu^{(medio)}_{\alpha_l,\beta_l,\alpha_r,\beta_r} = s^{(direita)}_{\alpha_l,\beta_l} - s^{(direita)}_{\alpha_g,\beta_g} \tag{1.9}$$

A estrutura de conexão de um sistema FuNe-I é determinada pelo tipo de escolha da partição *fuzzy*. Para cada unidade de entrada há quatro unidades na segunda camada, onde uma unidade representa o conjunto *fuzzy pequeno* e, uma outra unidade, representa o *grande*. As outras duas unidades, junto com uma unidade linear na terceira camada, representa o *médio*. Suas saídas são multiplicadas por $+1$ ou -1

e propagadas para a unidade linear. O número dessas unidades lineares dependem das partições das variáveis de entrada. É possível usar mais do que três conjuntos *fuzzy* para cada variável. Funções de pertinência que não são localizadas nos limites do domínio (*médio*), são representadas por duas funções sigmóides sobrepostas. A tarefa das unidades na segunda e na terceira camada é a de computar os graus de pertinência dos valores de entrada. As unidades de regras na quarta camada, calculam os graus de desempenho das regras *fuzzy*.

Para FuNe-I há procedimentos de aprendizado para adaptar as funções de pertinência e para determinar uma base de regra. FuNe-I somente usa regras com uma ou duas variáveis no antecedente. Para construir uma base de regra, regras com duas variáveis são consideradas separadamente para antecedentes conjuntivos ou disjuntivos. O processo de aprendizado é baseado em um algoritmo de treinamento especial, que não é considerado aqui. Veja as referências [79][80][138] para maiores detalhes.

A rede FuNe-I é treinada de forma que os pesos das regras e os pesos da camada de entrada e da segunda são modificados. O algoritmo de retropropagação clássico pode ser usado como o processo de aprendizado porque todas as funções dentro da rede FuNe-I são deriváveis. A rede pode conter muitas regras, talvés mais do que as regras desejadas por um usuário. Então, é possível eliminar unidades de regras cujos pesos são muito pequenos. Este tipo de rede é usada para classificação de padrões baseado em regras *fuzzy*.

1.3.10 HES com Abordagem de J.-S.R. Jang

Um dos primeiros sistemas *neuro-fuzzy* híbridos para a aproximação de função foi o modelo ANFIS (*Adaptive-Network-based Fuzzy Inference System*) de J.-S.R. Jang [96][97][98][138]. Este modelo representa um sistema *fuzzy* do tipo Sugeno[6]. em

[6]O modelo *fuzzy* de Sugeno foi proposto por Takagi, Sugeno, e Kang em um esforço para desenvolver uma abordagem sistemática para gerar regras *fuzzy* de um dado conjunto de dados de entrada/saída. Uma típica regra *fuzzy* em um modelo *fuzzy* de Sugeno tem a forma de (Veja a referência [99] para maiores detalhes deste tipo de regra.):

$$Se\ x\ é\ A\ e\ y\ é\ B\ Então\ z = f(x, y),$$

onde A e B são conjuntos *fuzzy* no antecedente, enquanto $z = f(x, y)$ é uma função abrupta no conseqüente. Usualmente $f(x, y)$ é uma polinomial nas variáveis de entrada x e y.

uma arquitetura de RNA direta com cinco camadas[7]. ANFIS implementa regras da forma:

$$R_r\text{: } Se\ x_i \text{ is } A_{ji}^{(1)} \wedge \cdots \wedge x_n \text{ é } A_{jn}^{n}\ Então\ y = \alpha_0^{(r)} + \alpha_1^{(r)} x_1 + \cdots + \alpha_n^{(r)} x_n.$$

Onde $alpha_n^{(r)}$ à $\alpha_n^{(r)}$ são os parâmentros dos conseqüentes de todas as regras e x_1, $x_n \cdots x_n$ são as variáveis de entrada.

A base de regras deve ser conhecida a priori. ANFIS ajusta somente as funções de pertinência dos parâmetros antecedentes e conseqüentes das regras.

A regra R_r, definida anteriormente, usa somente uma variável para a saída. Entretanto é fácil usar mais do que uma variável. Para cada variável de saída, uma combinação linear adicional deve ser especificada através de um conjunto adicional de parâmentros do conseqüente para cada regra. Por motivo de simplicidade será somente considerado o sistema ANFIS com uma única variável de saída (Figura 1.17). As considerações seguintes também são válidas para um sistema ANFIS de múltiplas saídas.

A estrutura da rede ANFIS contém n unidades de entrada na camada U_0. As outras camadas, denotadas por $U_1, \cdots, U_5$, têm as seguintes funções:

- Camada 1: Cada unidade em U_1 armazena três parâmentros para definir uma função de pertinência na forma de um sino, que representa um termo lingüístico como sendo:

$$\mu_j^i(x_i) = \frac{1}{1 + ((\frac{x_i - c}{a})^2)^b}$$

  onde x_i é uma variável de entrada. Cada unidade é conectada a exatamente a uma unidade de entrada e computa o grau de pertinência do valor de entrada obtida. a é o limite inferior da função de pertinência na forma de um sino, b é o limite médio desta mesma função e c é o limite superior;

- Camada 2: Cada regra é representada por uma unidade em U_2. Cada unidade é conectada àquelas unidades na camada anterior, as quais são do antecedente

[7]As entradas não são contadas como uma camada por J.-S.R. Jang [96][97][98].

da regra. As entradas em uma unidade $R_r \in U_2$ são os graus de pertinências, os quais são multiplicados para determinar o grau de desempemho τ_r para a regra representada por R_r (Estas unidades são nomeadas com o símbolo do produtório ($\prod$). Veja na Figura 1.17);

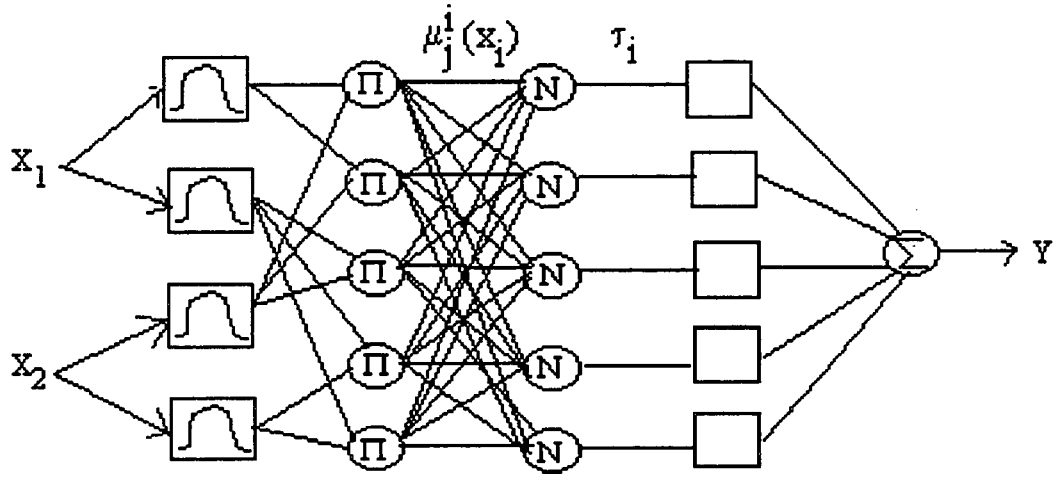

Figura 1.17: Estrutura do ANFIS

Onde:

$$\tau_r = \prod_{i=1}^{n} \mu_{j_r}^{(i)}(x_i)$$

- Camada 3: Nesta camada para cada regra R_r há uma unidade que computa o grau de desempenho relativo, dado por:

$$\bar{\tau}_r = \frac{\tau_r}{\sum_{R_i \in U_2} \tau_i}$$

Cada unidade é conectada a todas as unidades da regra em U_2. A letra N, que aparece na Figura 1.17, significa normalização;

- Camada 4: As unidades de U_4 são conectadas a todas as unidades de entrada (Estas ligações não estão mostradas na Figura 1.17) e para, exatamente, uma unidade em U_3. Cada unidade computa a saída de uma regra R_r por:

$$O_r = \bar{\tau}_r \cdot (\alpha_0^{(r)} + \alpha_1^{(r)} x_1 + \cdots + \alpha_n^{(r)} x_n)$$

- Camada 5: Uma unidade de saída computa a saída final y por fazer um somatório de todas as saídas de U_4.

Devido ANFIS usar somente funções deriváveis, torna-se fácil aplicar um algoritmo de aprendizado padrão da teoria de RNA. Para ANFIS uma mistura de retropropagação (gradiente descendente) e o *Least Square Estimation* (LSE) é usado. O algoritmo de retropropagação é utilizado para o aprendizado dos parâmetros antecedentes, isto é, as funções de pertinências, e o LSE é usado para determinar os coeficientes das combinações nos conseqüentes das regras. Uma das etapas no procedimento de aprendizado, subdivide-se em duas partes: na primeira parte, os padrões de entrada são propagados e os parâmetros do consequüente são estimados por um procedimento de LSE iterativo, enquanto os parâmetros do antecedente são determinados pelo conjunto de treinamento. Na segunda parte, os padrões são propagados novamente e, nesta etapa, o algoritmo de retropropagação é utilizado para modificar os parâmetros do antecedente, enquanto os parâmetros do conseqüente permanecem fixos. Este procedimento continua até um número de iterações pré-fixadas. Para maiores detalhes veja as referências [97][138].

1.3.11 HES com Abordagem de H.J. Zimmermann et al.

Em [138] é descrito um sistema *neuro-fuzzy* para predizer o *German DAX stock index* (Figura 1.18). Este sistema pode ser interpretado como uma rede *Radial Basis Functions* (RBF) especial. A estrutura da rede codifica os pesos da regra *fuzzy* cujos conseqüentes são números abruptos simples. Os conjuntos *fuzzy* nos antecedentes são modelados por funções gaussianas ou sigmóides, e o grau de desempenho de uma regra é determinado por multiplicar os graus de pertinências do antecedente dela. O valor da saída completa é computado por um somatório de pesos. Este sistema *fuzzy* é, portanto, um simples sistema tipo Sugeno de gerar regras *fuzzy*.

O algoritmo de aprendizado é baseado no de retropropagação e para um problema de aprendizado específico. O algoritmo modifica os parâmetros das funções de pertinência, os valores do conseqüente e os pesos das regras. A soma dos pesos permanecem constantes durante o aprendizado, ou seja, as regras computam com cada uma das regras com altos valores de pesos. Isto é feito para identificar regras supérfluas, as quais, idealmente, devem ter pesos próximos a zero após o aprendizado.

O algoritmo tenta preservar a semântica da base de regras. Um dado usuário pode especificamente restringir certas modificações que não são permitidas. Por

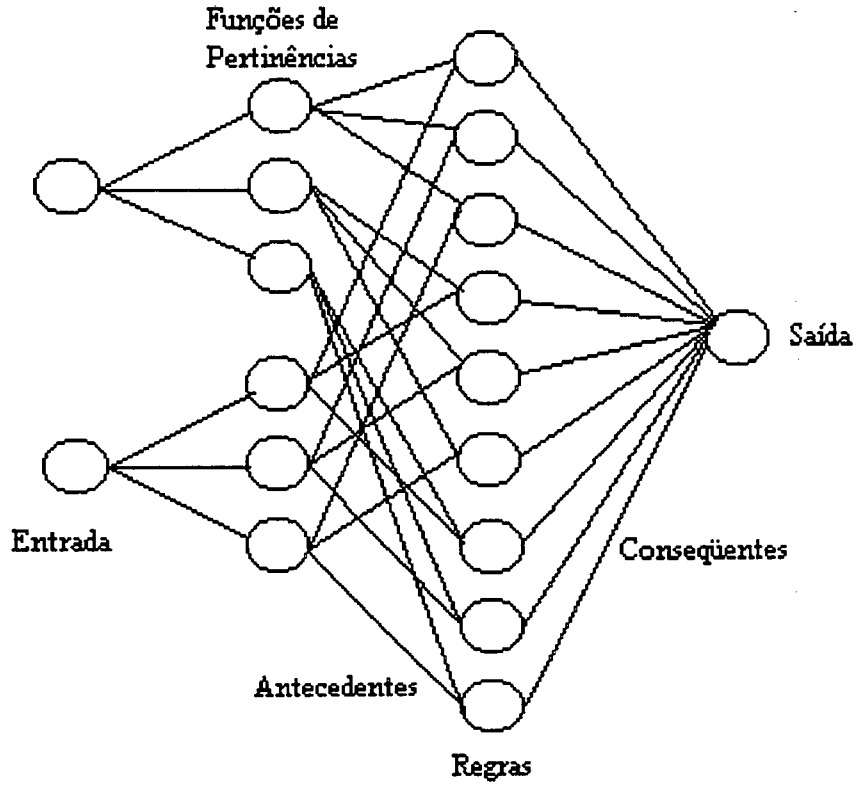

Figura 1.18: Modelo *neuro-fuzzy*: sistemas *fuzzy* tipo Sugeno simples com regras simples

exemplo, conjuntos *fuzzy* de mesma variável devem manter suas posições relativas ou alguns dos conjuntos *fuzzy* devem ser sempre idênticos.

Após o treinamento, algoritmos padrões que utilizam um tipo de processo de poda para RNA podem ser usados para deletar regras completas ou variáveis dos antecedentes das regras.

Finalizando, além dos HES abordados anteriormente, há outros trabalhos como [36][55][68][90][91][93][94][104][146][167][172][180][185][138], que são outros modelos híbridos desenvolvidos, porém, com algumas diferenças na mesma linha das arquiteturas citadas anteriormente, isto é, eles usam os princípios das redes diretas estáticas. Pode-se ainda citar os trabalhos de J.M. Barreto e F.M. de Azevedo [46][48][44][45][47][42][14] [13][12], os quais propuseram uma outra metodologia, baseadas em redes recursivas. Entretanto, esta abordagem, por estar fora do escopo deste trabalho, não será aqui tratada.

Sumariamente, uma grande maioria dos modelos desenvolvidos para sistemas híbridos têm em comum, servir como uma ponte entre os NNES e os RBES, permitindo assim, a construção de NNES a partir da definição de conceitos, relações e regras, como é feito para os RBES.

1.4 Técnicas de Extração de Regras para RNA

Explanação ao usuário é uma importante função em IA. Experiências com SE e RNA têm mostrado que a capacidade para gerar explicações é absolutamente crucial para a aceitação pelo usuário. RNA têm tradicionalmente tido dificuldades em gerar estruturas de explanações. Apesar de ser um problema em aberto muitos resultados significativos têm sido obtidos.

Explanação é a função chave em sistemas de IA. Ela é usada para atualizar estruturas de conhecimento em raciocínio baseado em casos, quando uma falha é reconhecida, isto é, aprendizado direcionado à falhas. Explanação é, também, utilizada para esclarecer os resultados de um processo de raciocínio para um usuário. Em alguns casos, este usuário pode nem sempre ser um especialista de um domínio, mas ele tem a responsabilidade de aceitar ou rejeitar uma solução produzida por um sistema de IA. Além disso, a explanação pode ser usada para aprendizado de conhecimento intensivo sempre que uma completa teoria de domínio é dada, bem como para esclarecer, por exemplo, resultados obtidos de uma RNA com respeito à classificação de uma dada doença (diagnóstico médico).

O termo explanação se refere a uma estrutura explícita, a qual pode, internamente, ser usada para raciocínio e aprendizado e, externamente, para a explicação de resultados para um usuário. Em sistemas baseados em regras, por exemplo, explanação inclue etapas intermediárias do processo de raciocínio, como é o caso de uma estrutura de provas, quais regras foram disparadas, etc. Esta estrutura pode ser utilizada para responder questionamentos do tipo *como*. Por exemplo, como a solução w foi produzida por um sistema de inferência? Devido à certas condições x e y que foram satisfeitas após os primeiros dados de entradas e levaram às conclusões w e z, as quais satisfizeram a condição k, e assim por diante. Este tipo de explanação, dentro de uma certa limitação, é absolutamente crucial para a aceitação pelo usuário de um sistema de inferência [53].

Experiências com SE têm mostrado que usuários exigem uma explicação de um resultado produzido por um SE e não aceitam uma solução sem explicação [41]. Conseqüentemente, esforços foram feitos na área de SE para permitir explanações, as quais, no mínimo, são de alguma coerência, significado e grau.

Embora a IA simbólica tenha introduzido várias formas bem sucedidas de explanação ao usuário, a transparência de uma explicação não é garantida. Uma base de regra organizada pobremente, por exemplo, com centenas de premissas por regras, destrói completamente a transparência de explanações baseadas em regras, isto é, a compreensibilidade das regras. Portanto, explanações baseadas em regras são agora amplamente reconhecidas como muito rígidas e inflexíveis [72][136].

Em [136], os autores queixam-se que o uso anterior de textos ou *templates* como parte das explanações ao usuário eram tão rígidos que o sistema sempre interpretava os questionamentos do mesmo modo e que havia uma perda de estratégia de respostas. Embora haja esforços de se obter vantagens de estratégias de diálogos de linguagem natural com iniciativas mistas, de modelos-usuários e de explanações planejadas explicitamente, existem poucas dúvidas que os sistemas correntes (regras, por exemplo) são ainda tão inflexíveis, não suscetíveis, incoerentes e rígidos. Conseqüentemente, se a explanação ao usuário é feita através da geração de conjuntos de regras (em IA simbólica e RNA), a qualidade e compreensibilidade da regra são tópicos importantes [136].

RNA são sistemas tipo *caixas-pretas*. Obter uma explanação sobre o raciocínio de uma RNA não é tão fácil. Isto se deve, principalmente, ao fato que o conhecimento aprendido é representado pela topologia da rede e pelos valores de pesos e *bias*. Estes, usualmente, não são compreendidos claramente por seres humanos. A perda da capacidade de explanação é uma das principais razões do porquê das RNA não despertarem o interesse das indústrias. Na maioria das aplicações do mundo real, usuários querem saber o raciocínio por trás da conclusão de um sistema de aprendizado ou de um SE. Assim, uma RNA deve ser capaz de dar uma explanação a uma dada resposta na saída desta. Isto pode ser feito de vários modos. Por exemplo, extrair regras *Se/Então*, convertendo RNA à árvores de decisões.

Extrair regras *Se/Então* é usualmente aceita como a melhor maneira de extrair o conhecimento representado na RNA. Não por causa de ser um trabalho fácil, mas devido as regras criadas no final, pois são mais compreensíveis para humanos do

que por alguma outra representação (por exemplo, árvores de decisão), e as regras extraídas de RNA treinadas podem ser usadas, ainda, em outros sistemas, como é o caso dos SE.

Os méritos de incluir técnicas de extração de regras, como um acessório para técnicas de RNA convencionais, incluem [31]:

1. Provisão de uma capacidade de explicação ao usuário;
2. AC para sistemas de IA simbólica para superar o *gargalo* da engenharia de conhecimento;
3. Potencial para contribuir para o entendimento de como as abordagens simbólicas e conexionistas para IA podem ser integradas beneficamente.

Então, técnicas de extração de regras procuram esclarecer ao usuário como a rede chegou a uma decisão, decodificando o(s) estado(s) interno(s) da RNA. A maioria dos esforços tem sido direcionado no sentido de apresentar as explanações como um conjunto de regras expressas como lógica simbólica convencional, isto é, valores booleanos, na forma de *Se ... Então ... Senão ....* Um esforço substancial tem sido também direcionado com respeito ao conhecimento codificado na RNA, usando-se conceitos extraídos da lógica *fuzzy*. Isto permite que regras sejam expressas em uma forma na qual trata com o que são verdades parciais. Por exemplo, *Se ... Então ... pode-ser-verdade* ou *Se ... Então ... poderia-possivelmente-ser-verdadeiro.*

Um outro assunto, o qual está intimamente relacionado à extração de regras de RNA, diz respeito ao uso de RNA para refinar regras simbólicas existentes. O ponto inicial neste processo é uma base de conhecimento inicial (regra), a qual pode não necessariamente ser completa, ou mesma, correta. Neste caso, a RNA é usada para produzir uma representação melhorada do domínio do problema, do qual um conjunto refinado de regras simbólicas pode ser extraído. Como este assunto não é pertinente a este trabalho, veja a referência [53] para maiores esclarecimentos.

1.4.1 Refinamento do Conhecimento

Sistemas de refinamento de conhecimento usam conhecimento de domínio impreciso. O conhecimento de domínio é integrado em um sistema de aprendizado de máquina (simbólico ou conexionista), o qual é revisado treinando o sistema com os exemplos disponíveis e o conhecimento do sistema é extraído como conhecimento revisado. O conhecimento extraído deve ser superior ao conhecimento de domínio

inicial.

O conhecimento extraído deve estar em uma forma compreensível para humanos (provavelmente na mesma forma como o conhecimento do domínio). Usualmente o conhecimento do domínio e o conhecimento extraído do sistema treinado são representados como regras *Se/Então*. Em sistemas de aprendizado simbólico, cada conhecimento aprendido é representado por regras *Se/Então* ou eles podem ser facilmente convertidos para regras *Se/Então*. Se o sistema de aprendizado é uma RNA, a extração de conhecimento se torna complicada.

A vantagem principal de sistemas de refinamento do conhecimento é que o conhecimento do domínio pode ser incompleto e não necessariamente correto como é exigido em um sistema baseado em explanações. Desde que o conhecimento do domínio seja equivalente a um número de exemplos, não há a necessidade de se ter tantos exemplos quantos requerem os sistemas de aprendizado simbólico. Algumas das técnicas de refinamento de conhecimento existentes usando RNA podem ser encontrados em [181][183][143][64][65].

1.4.2 Classificação

Conforme em [4][5], os tipos de técnicas de extração de regras são subdivididas em: decomposicional, pedagógica e eclética. Cada uma destas será abordada a seguir.

Decomposicional

A abordagem decomposicional é aquela em que o foco está nas regras extraídas ao nível de unidades individuais (intermediárias e saídas) no interior da RNA treinada. Então, a *visão* fundamental da RNA treinada é a de *transparência*. Uma condição básica para técnicas de extração de regras, nesta categoria, é que a saída computada (valor) de cada unidade intermediária e de saída na RNA treinada, deve ser mapeada como um resultado booleano (sim/não), o qual corresponde à noção do conseqüente de uma regra. Assim, cada unidade intermediária e de saída pode ser interpretada como uma função degrau ou uma regra Booleana, a qual reduz o problema de extração de regra para um problema de determinar as situações no qual a regra é verdadeira, ou seja, um conjunto de conexões resultantes cuja soma de pesos garante que as *bias* das unidades é excedida independentemente do valor da ativação presente em outras conexões resultantes. As regras extraídas ao nível

da unidade individual são, então, agregadas para formar a base de regras composta para a RNA como um todo.

Pedagógica

Esta abordagem é dada para aquelas técnicas de extração de regras as quais tratam a RNA treinada como uma *caixa-preta*, ou seja, a visão fundamental da RNA treinada é *opaca*. A idéia principal na abordagem pedagógica é considerar a extração de regra como uma tarefa de aprendizado, onde o conceito alvo é a função computada pela rede e as características das entradas são, simplesmente, as características das entradas da RNA. Assim, as técnicas pedagógicas auxiliam a extrair regras que mapeiam entradas diretamente em saídas. Tais técnicas tipicamente são usadas em conjunto com algoritmo de aprendizado simbólico e a motivação básica é usar a RNA treinada para gerar exemplos para o algoritmo de aprendizado.

Eclética

Nesta classificação, são compostos elementos de ambas as técnicas de extração de regras decomposicional e pedagógica. Nesta categoria são determinadas as técnicas que utilizam o conhecimento sobre a arquitetura interna e/ou os vetores de pesos na RNA treinada para complementar um algoritmo de aprendizado simbólico.

1.4.3 Extração de Regras Booleanas Usando Abordagens Decomposicionais

RNA utilizando o algoritmo de retropropagação padrão para treinamento, têm sido aplicadas com sucesso para domínios de problemas envolvendo aprendizado e generalização. Assim houve uma motivação forte para se desenvolver e se aplicar técnicas de extração de regras para tais RNA. Alguns dos trabalhos precursores nesta área, adotaram o termo mencionado na subseção anterior, como abordagem decomposicional, no sentido de focalizar uma procura e fazer a extração de regras Booleanas convencionais ao nível das unidades individuais (intermediárias e saídas) da RNA treinada [65][182]. De interesse particular, são duas abordagens nas quais a motivação básica foi considerada: procurar incialmente os conjuntos de pesos contendo uma ligação/conexão de um valor suficiente (positivo) para garantir que a *bias* na unidade sendo analisada é excedida, independente dos valores nas outras ligações/conexões. Se uma ligação/conexão é encontrada, de modo que satisfaça o critério, ela é escrita como uma regra. A procura, então, procede para subconjuntos de dois elementos e, assim, sucessivamente. As regras extraídas ao nível de unidade

individual são, então, agregadas para formar uma base de regra composta para a RNA como um todo.

Uma das primeiras técnicas de extração de regras de uma RNA foi desenvolvida por L.M. Fu [59][57]. Ele desenvolveu um algoritmo denominado de *KT*. Neste tipo de algoritmo, o problema de mapeamento da saída de cada unidade (intermediária e saída) em uma função booleana foi alcançada pelo simples artifício de: *Se* $0 \leq$ *saída* $\leq$ *limite*$_1$, *Então não; Se limite*$_2 \leq$ *saída* ≤ 1, *Então sim*; onde *limite*$_1 <$ *limite*$_2$.

Um outro exemplo para esta linha de abordagem é o algoritmo *SUBSET*. Ele foi desenvolvido por G. Towell e J. Shavlik [182][183]. Neste algoritmo, o foco está em extrair regras a partir de cada neurônio das camadas intermediária e de saída. Deve-se seu nome posto que a idéia básica é procurar por subconjuntos de pesos para cada neurônio cuja soma supera o valor da *bias* do mesmo. Considera-se que os pesos podem assumir valores positivos e negativos e, também, que os neurônios estejam na máxima ativação (próximos de um) ou, então, inativos (próximos de zero) [50].

O algoritmo SUBSET pode ser descrito da seguinte forma:

- Para cada neurônio das camadas intermediárias e de saída fazer:
  - Formar S_p subconjuntos, combinando somente pesos positivos do neurônio cujo somatório supera o valor de *bias*;
  - Para cada elemento P dos subconjuntos S_p fazer:
    - Formar S_n subconjuntos, de N elementos, considerando as combinações mínimas de pesos negativos, de forma que a soma absoluta destes pesos seja maior do que a soma de P menos o *bias* do neurônio;
    - Formar a regra: *Se P e not N Então <nome do neurônio>*

Um exemplo de aplicação deste algoritmo, que pode ser comprovado intuitivamente, é mostrado na Figura 1.19 [50].

A maior limitação deste tipo de técnica de extração de regras está no processamento excessivo necessário para alcançar as soluções possíveis, principalmente em RNA com elevado número de neurônios. Outro problema que surge é que algumas regras geradas podem ter um elevado número de antecedentes, tornando, algumas vezes, proibitivo o seu uso prático.

A técnica RuleNet de C. McMillan et al. [131] é um dos primeiros exemplos que foi empregada para uma RNA especializada incorporando a abordagem decomposicional, a qual é usada como a base para extrair regras booleanas. RuleNet é uma RNA que aprende a mapear *strings* → *strings* e da qual regras *condição/ação* podem ser extraídas. O *modus operandi* no RuleNet é iterar através das seguintes etapas:

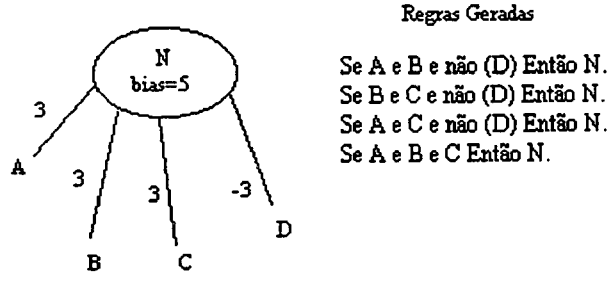

Figura 1.19: Exemplo do algoritmo SUBSET

1. Treinar uma RNA para um conjunto de padrões entrada/saída;
2. Extrair as regras simbólicas usando os pesos das conexões na rede;
3. Injetar as regras extraídas no sentido inverso na rede e continuar treinando.

O processo termina quando a base resultante das regras extraídas caracteriza adequadamente o domínio do problema. O treinamento da RNA especializada é baseado no trabalho de Jacob et al. [95] e incorpora uma camada de entrada, uma de saída e uma intermediária, onde esta última, corresponde a parte do antecedente de uma regra (ou condição), enquanto que a camada de saída, ao conseqüente de uma regra (ou ação). A maior crítica desta abordagem é que a arquitetura/treinamento da RNA especializada é direcionada para um domínio de problema específico e, portanto, a abordagem perde a generalidade.

Um desenvolvimento importante na utilização de arquiteturas de RNA especializadas foi a publicação do algoritmo MofN proposto por G.G. Towell e J.W. Shavlik [182]. Originariamente, foi desenvolvido para extrair conhecimento de uma estrutura particular de RNA, definida a partir de uma base de dados gerada por um conjunto de regras de produção sobre um determinado conhecimento especializado, ou seja,

de uma rede baseada em conhecimento chamada KBANN (*Knowledge Based Artificial Neural Network*) [183]. Este algoritmo foi desenvolvido como uma forma de reduzir as limitações da técnica SUBSET, diferindo desta, basicamente, na maneira como apresenta as regras geradas. O formato geral das regras é:

Se (M dos seguintes N antecedentes são verdadeiros) Então ...

A idéia básica desta técnica é que antecedentes individuais não possuem todos o mesmo grau de importância. Deve-se supor que grupos de antecedentes formam classes equivalentes, nas quais cada antecedente tem a mesma importância e podem ser trocados entre si (membros de uma mesma classe).

A seguir é dada, resumidamente, a descrição básica da técnica MofN, isto é:

1. Para cada neurônio das camadas intermediárias e de saída, formar grupos de pesos similares numericamente;
2. Atribuir aos pesos de cada grupo o valor médio calculado para o grupo;
3. Eliminar qualquer grupo que não tenha efeito significante para a ativação do neurônio (valor médio pequeno);
4. Mantendo os pesos constantes, otimizar os valores das *bias* de todos os neurônios através do algoritmo de retropropagação;
5. Extrair uma única regra a partir dos grupos, considerando a soma dos pesos dos grupos e o valor das *bias* otimizadas;
6. A partir da regra geral, simplificar pesos e combinações possíveis.

Um exemplo de aplicação deste algoritmo é mostrado na Figura 1.20 [50].

Um outro desenvolvimento significante na evolução de técnicas para extrair regras de uma RNA treinada, é a técnica RULEX de R. Andrews e S.Geva [6]. Esta técnica foi elaborada para explorar o comportamento de um algoritmo particular de treinamento de RNA - a *Constrained Error Backpropagation* (CEBP), que é um tipo de algoritmo que executa as tarefas de classificação e aproximação de maneira similar às redes RBF (*Radial Basis Function*). As unidades intermediárias da rede CEBP são unidades suscetíveis localmente (*Locally Responsive Units - LRU*) baseada na sigmóide, que tem o efeito de particionar os dados treinados em um conjunto de regiões disjuntas, sendo que cada região é representada por uma unidade da camada

intermediária única. Cada LRU é composta de um conjunto de *ridges* (arestas), isto é, uma aresta para cada dimensão da entrada.

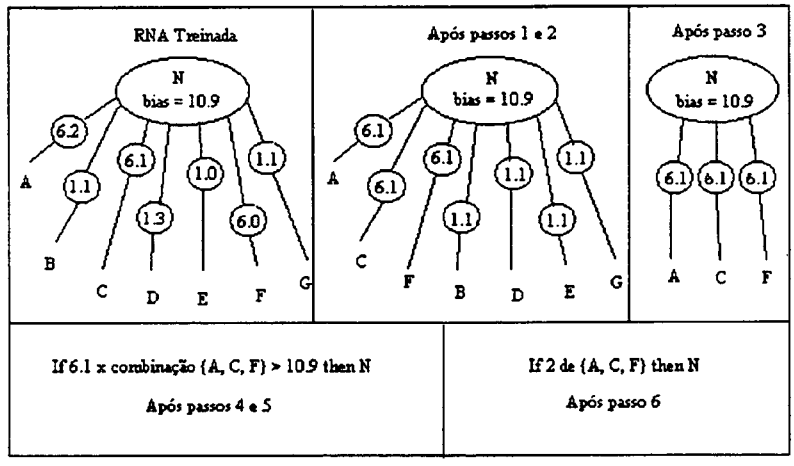

Figura 1.20: Exemplo do algoritmo MofN

Uma aresta produzirá uma saída plausível somente se o valor apresentado como entrada estiver dentro da faixa ativa da aresta. A saída da LRU é a soma do limite das ativações das arestas. Assim, para que um vetor seja classificado por uma LRU, cada componente do vetor de entrada deve estar dentro da faixa ativa da sua correspondente aresta. Desta forma, é possível se obter regras proposicionais, que são extraídas das LRU. Por exemplo:

> *Se* $Aresta_1$ *é ativa e* $Aresta_2$ *é ativa e ...* $Aresta_n$ *é ativa Então o padrão pertence a Classe Alvo*

A técnica RULEX também contém procedimentos para manter antecedentes negativos, bem como para remover antecedentes e regras redundantes. Diferente de outros métodos decomposicionais, tais como KT e SUBSET, os quais empregam variações de técnicas de *procura e teste*, RULEX desempenha extração de regra por interpretar diretamente os vetores pesos como regras. Conseqüentemente, RULEX livra-se dos problemas computacionais das outras técnicas decomposicionais e utiliza o recurso empregado pelas heurísticas para controlar a procura do espaço de solução. Esta técnica tem sido adptada para acomodar tanto os dados de entrada discretos, como os contínuos e os mistos.

1.4.4 Extração de Regras Booleanas Usando Abordagens Pedagógicas

Na abordagem pedagógica para extração de regras, um dos primeiros trabalhos publicados foi o de K. Saito e R. Nakano [160]. Nesta implementação, a RNA é tratada como uma *caixa-preta* com regras definidas para um domínio de problema na área de diagnóstico médico, extraídas em virtude das mudanças nos níveis dos neurônios de entrada e saída. Eles, também, trataram o problema referente ao tamanho do espaço de solução para evitar combinações insignificantes de entradas (sintomas médicos neste domínio) e restringir o número máximo de sintomas coincidentes a serem considerados. Mesmo com estas heurísticas, o número de regras extraídas para um domínio de problema considerado relativamente simples, foi extremamente grande. Este resultado enfatiza uma das maiores preocupações com as técnicas de extração de regras, ou seja, visam a explanação e não o *obscurecimento.*

A técnica Análise-VI (VIA), desenvolvida por S.B. Thrun [178], é também a síntese de uma abordagem pedagógica que extrai regras que mapeiam entradas diretamente em saídas. O algoritmo usa um procedimento de *gerar-testar* para extrair regras simbólicas de RNA, que utiliza o algoritmo de retropropagação padrão como treinamento para tais redes. As etapas básicas deste algoritmo são:

1. determinar intervalos arbitrários para todas as unidades, ou um subconjunto destas, na RNA. Estes intervalos constituem restrições nos valores para as entradas e as ativações da saída;

2. refinar os intervalos por detecção e excluir iterativamente valores de ativação que são provados inconsistentes com os pesos e *bias* da RNA;

3. o resultado da etapa 2 é um conjunto de intervalos[8], os quais são consistentes ou inconsistentes com os pesos e *bias* da RNA.

S.B. Thrun compara a abordagem dele à análise de sensibilidade, em que caracteriza a saída da RNA treinada por variações sistemáticas nos padrões de entrada e por examinar as mudanças na classificação da RNA. A técnica é diferente, fundamentalmente, das outras porque esta analisa as ativações das unidades individuas na RNA treinada, em que o foco está no que é chamado de *intervalos de validade.* Um intervalo de validade de uma unidade especifica uma faixa máxima para seu valor

[8]Neste contexto, um intervalo é definido como sendo inconsistente se não há padrão de ativação, o qual, absolutamente, pode satisfazer as restrições impostas pelos intervalos de validade iniciais.

de ativação. A técnica resultante prove uma ferramenta geral para checar a consistência das regras em uma RNA treinada. O algoritmo VIA é designado como uma procedimento de objetivo geral para extração de regras. Enquanto a técnica VIA parece não ser limitada a uma classe específica de um dado domínio de problema, S.B. Thrun relata que este algoritmo falhou para gerar um conjunto completo de regras em um problema relativamente complexo envolvendo a tarefa de treinar uma rede para ler em voz alta (NETtalk).

A técnica *Extração de Regra como Aprendizado* foi desenvolvida por M.W. Craven e J.W. Shavlik [39] e é uma técnica de extração de regra que utiliza a abordagem pedagógica. Uma característica proeminente desta técnica é que, dependendo da implementação particular usada, a abordagem *Extração de Regra como Aprendizado* pode ser classificada tanto como uma pedagógica quanto decomposicional. A diferença está no procedimento usado para estabelecer se uma dada regra concorda com a rede. Este procedimento aceita uma classe c e uma regra r e retorna *verdadeiro* se todos os exemplos convertidos por r são classificados como membros da classe c. Se, por exemplo, o algoritmo VIA é usado por este processo, então a abordagem é pedagógica, enquanto que se uma implementação tal como aquela de L.M. Fu é usada, a classificação da técnica é decomposicional. Como a técnica VIA discutida anteriormente, a técnica *Extração de Regra como Aprendizado* também não requer um algoritmo especial de treinamento para a rede. Os autores sugerem dois critérios de parada para controlar o algoritmo de extração de regra, a saber: (1) estimar se o conjunto de regras extraídas é um modelo suficientemente preciso da RNA ou (2) terminar após um certo número de iterações em que os resultados não produzam regras novas, ou seja, um critério de *paciência*. Os autores relatam tanto a complexidade do algoritmo desta abordagem, quanto a qualidade das regras extraídas, com particular ênfase na *fidelidade*[9] de regra, a qual é medida por comparar o desempenho de classificação de um conjunto de regra à RNA treinada, da qual as regras foram extraídas.

A idéia principal desta técnica é tratar a extração de regra como uma tarefa de aprendizado, onde o conceito alvo é a função computada pela rede e as características de entrada são simplesmente as características de entrada da rede. A descrição resumida de um algoritmo que usa esta técnica é dada a seguir.

/* Inicialize as regras R_c para cada classe */

[9]A fidelidade de um conjunto de regras é a fração de exemplos no qual o conjunto de regra concorda com a RNA treinada.

Para cada classe c

$R_c := 0$

repetir

e := Exemplos()
c := Classificado(e)
Se e não convertido por R_c, *Então*

/* aprende uma nova regra */
r := regra conjuntiva formada de e
Para cada antecedente r_i de r

ŕ := r mas com r_i reduzido
Se subconjunto(c, ŕ) = verdadeiro
Então r := ŕ

$R_c := R_c \vee r$

Até encontrar o critério de parada

A tarefa da função *Exemplos* é prover exemplos de treinamento para o algoritmo de aprendizado de regra. As opções são: (1) selecionar membros do conjunto usado para treinar a RNA, (2) amostrar aleatoriamente ou (3) criação de exemplos de uma classe específica. A função subconjunto(c, ŕ) determina se a regra modificada ainda concorda com a rede, ou seja, se todos os exemplos que são convertidos pela regra são membros da classe dada.

Um outro exemplo recente de uma abordagem pedagógica é a RULENEG. Ela foi desenvolvida por E. Pop et al.[152] e focaliza apenas a tarefa de extração de regras conjuntivas. A gênese da técnica RULENEG é a observação que cada regra simbólica, no cálculo proposicional, pode ser expressa como uma disjunção de conjunções. Além disso, uma regra conjuntiva é mantida somente quando todos os antecedentes na regra são verdadeiro. Então, por mudar o valor verdade de um dos antecedentes, o conseqüente da regra muda. Dada uma RNA treinada e os padrões usados na fase de treinamento da RNA, as regras aprendidas pela RNA são extraídas de acordo com o algoritmo descrito a seguir.

Inicialize o *Rule-Holder* para vazio
Para cada padrão s do conjunto treinado

Ache a classe C para s pelo uso da RNA /*C = RNA(s)*/
Se s não é classificado pelas regras existentes

Inicialize a nova regra r para a classe C
Para cada entrada i na rede
Faça uma cópia $\acute{s}$ de s
Negue a i-éssima entrada em $\acute{s}$
Ache a classe $\acute{C}$ para $\acute{s}$ pelo uso da
RNA /* $\acute{C}$ = RNA($\acute{s}$) */
Se C não é igual a $\acute{C}$
adicione a i-éssima entrada e o seu valor verdade para r
Fim para cada entrada
Adicione r para o Rule-Holder
Fim para cada padrão

RULENEG é designado para selecionar somente regras conjuntivas por padrão de entrada, mas esta técnica é ainda capaz de extrair todas as regras aprendidas dos padrões.

O sistema BRAINNE de S. Sestito e T. Dillon [162] foi projetado para extrair regras de uma RNA treinada usando o algoritmo de retropropagação clássico. Neste contexto, é classificado como pedagógico, desde que basicamente seja usada uma medida de proximidade entre as entradas e saídas da rede como o ponto principal para gerar o conjunto de regra. A classificação suplementar desta abordagem, como uma condição de um regime de treinamento para uma RNA especializada, é baseada em uma nova idéia de tomar uma rede inicial treinada com m entradas e n saídas e transformá-la em uma rede com $m + n$ entradas e n saídas. Então, esta rede transformada é treinada novamente. A próxima fase, no processo, é desempenhar uma comparação na direção do par dos pesos para as conexões entre cada uma das unidades de entrada adicionais n e as unidades intermediárias correspondentes. A menor diferença entre os dois valores é a maior contribuição da unidade de entrada original, ou seja, um atributo do domínio de um problema, para a saída. Uma inovação maior na técnica BRAINNE é a capacidade de lidar com dados contínuos como entrada, sem primeiro ter que empregar uma fase de *discretização*. A técnica BRAINNE segmenta automaticamente os dados contínuos em faixas discretas e extrai regras diretamente correspondendo a *Se ... Então ... Senão ....*

Uma abordagem adicional é a técnica DEDEC de A.B. Tickle et al. [179][4]. Ela tem como objetivo central extrair o conhecimento codificado em uma RNA

treinada como um conjunto de regras simbólicas. Esta técnica é dividida em três fases. No ponto inicial, a Fase 1, usa uma RNA treinada em um dado domínio de problema. Ela é aplicável à RNA diretas com multicamadas e treinadas pelo método de retropropagação. As Fases 2 e 3 representam os dois processos principais desta técnica. Em essência, o objetivo primário da Fase 2 é evitar a necessidade de gerar e testar todas as combinaçêos possíveis de entradas da RNA, com o propósito de identificar as dependências funcionais entre as entradas e saídas da RNA. Isto é atingido de forma a analisar os vetores de pesos e a arquitetura da RNA treinada, produzindo:

1. uma classificação das entradas da RNA em ordem de suas importâncias relativas de modo a predizer a saída da RNA;

2. um aglomerado (*clustering*) das entradas da RNA classificadas.

O objetivo da Fase 3 é utilizar a informação obtida na Fase 2 para focalizar a procura de dependências funcionais e, então extrair as regras correspondentes. O ponto inicial é o conjunto inicial de atributos de dados usados para treinar a RNA e as decisões correspondentes computadas pela RNA[10] treinada. A seguir são descritas as etapas da Fase 3.

1. Selecione os aglomerados próximos dos atributos identificados no processo de classificação/aglomerado (Fase 2);

2. Escolha cada caso do conjunto resultante treinado, isto é, o conjunto de casos do conjunto original treinado, para o qual uma regra para a classificação correta ainda não tenha sido encontrada em um caso contendo somente aqueles atributos extraídos dos aglomerados dos atributos selecionados para este ponto, encadeados com o atributo de decisão;

3. Testar este conjunto de casos selecionados para a dependência funcional;

4. Se a dependência funcional existe, então termine o processo. Caso contrário, gere o conjunto mínimo de regras para aqueles valores dos atributos os quais não ambiguamente determinam um valor de decisão;

5. Remova os casos correspondentes do conjunto de treinamento resultante corrente.

[10]Se a RNA foi treinada para um ponto de erro zero, então, este conjunto e o conjunto de treinamento original seriam idênticos.

Enquanto a existência de dependência funcional constitui um ponto definitivo limitador, é possível introduzir um número alternativo de critério de parada, isto é, quando um número específico de iterações seja alcançado ou quando o tamanho do conjunto de treinamento resultante cai abaixo de algum valor limite.

1.4.5 Extração de Regras Fuzzy

Paralelo ao desenvolvimento de técnicas para extração de regras booleanas de RNA treinadas, os sistemas *neuro-fuzzy* têm sido a síntese de técnicas correspondentes para extração de regras *fuzzy*. Analogamente ás técnicas discutidas previamente para os sistema convencionais de lógica booleana, sistemas *neuro-fuzzy* compreendem três elementos distintos: sendo o primeiro elemento um conjunto de mecanismos/procedimentos que introduz o conhecimento de um especialista na forma de regras *fuzzy* em uma estrutura de RNA (fase de inicialização do conhecimento). A diferença essencial aqui, é que, esta etapa envolve a geração da representação das funções de pertinência. O segundo elemento é o processo de treinamento da RNA, o qual, neste caso, focaliza as funções de pertinência de acordo com os padrões nos dados de treinamento. E finalmente, a análise e extração do conhecimento refinado inserido na forma de um conjunto de funções de pertinência modificado constitui-se ao último passo.

Um dos primeiros trabalhos desenvolvido nesta área foi o de R. Masuoka et al. [125]. Eles usaram uma abordagem decomposicional para definir um conjunto inicial de regras *fuzzy* extraídas de especialistas em um dado domínio de problema. A técnica é formada por uma arquitetura de RNA especializada com três fases: a fase de entrada, onde uma RNA de três camadas é composta de uma unidade de entrada, uma ou duas unidades intermediárias e uma unidade de saída, é usada para representar a função de pertinência de cada antecedente da regra, ou seja, as variáveis de entrada. As operações *fuzzy* nas variáveis de entrada {E, OU, NÃO} são representadas pela segunda fase chamada como a fase de *Rule Net* (RN) e, por fim, as funções de pertinências, as quais constituem os conseqüentes das regras, são representadas em uma terceira fase (saída). Nesta técnica, o problema de elicitar um conjunto compacto de regras como a saída é fornecido pelo corte na fase RN, no sentido daquelas conexões na rede que são menores do que um valor limite.

Em um modo similar, H.R. Berenji [15] demonstrou o uso de uma RNA especializada para refinar uma base de conhecimento de regras *fuzzy*, usada como parte de um controlador. A característica marcante desta técnica é que o conjunto de regras

governadas pela operação do controlador é conhecida. Além disso, a RNA é usada para modificar as funções de pertinência tanto para os antecedentes quanto para os conseqüentes das regras.

S. Horikawa et al. [93] desenvolveram três tipos de RNA *fuzzy*, as quais podem identificar, automaticamente, as regras *fuzzy* fundamentais, bem como refinar as funções de pertinência correspondentes modificando os pesos das conexões das RNA usando o algoritmo de retropropagação. Nesta abordagem, a base da regra inicial é criada usando o conhecimento de um especialista ou iteragindo seletivamente através de possíveis combinações das variáveis de entrada e o número de funções de pertinência. O tipo de RNA *fuzzy* utilizada por esta técnica corresponde ao modelo FuNe-I, desenvolvido por S.K. Halgamuge e M. Glesner [80], o qual é uma generalização deste trabalho no sentido de usar um processo baseado em regra para identificar inicialmente os nós relevantes da regra para regras conjuntivas e disjuntivas em relação a cada saída.

Ambos, FNES (*Fuzzy Neural Expert System*) de Y. Hayashi [82][83][84][85] e o modelo *fuzzy* para RNA multicamadas de S. Mitra et al. [134][133] (cujos sistemas híbridos já foram descritos na Seção 1.3), são direcionados especificamente para o problema de prover ao usário uma explanação (justificação) de como uma conclusão particular foi atingida. Em ambas as técnicas, o conjunto dos antecedentes das regras é determinado pela análise e classificação dos vetores de pesos na RNA treinada para determinar a influência relativa em uma dada saída (classe). Entretanto, enquanto FNES lida com o envolvimento de um especialista na fase de entrada para converter os dados de entrada no formato desejado, no procedimento do modelo *fuzzy* para RNA multicamadas, este processo foi automatizado. A seguir é descrito um algoritmo de extração de regras proposto por Y. Hayashi. Contudo, faz-se necessário fazer algumas considerações. Um dos problemas desta abordagem corresponde a determinar o valor dos grupos das células de entrada da rede em questão. Cada célula de saída O_k assume os valores +1 ou -1. As proposições (itens de entrada) correspondem aos grupos de células de entrada que estarão em um antecedente *se-parte* de cada regra. Além disso, as ativações das células de entrada $I_i(i = 1, 2, \cdots, p)$, a célula intermediária $H_i(j = 1, 2, \cdots, q)$ e a célula de saída $O_k(k = 1, 2, \cdots, r)$ podem ser calculadas usando as seguintes fórmulas:

$$SH_j = \sum_{i=0}^{p} w_{ji} I_i$$

$$SO_k = \sum_{i=0}^{p} u_{ki} I_i + \sum_{j=1}^{q} v_{kj} H_j$$

$$H_j = \begin{cases} +1 \; ou \; Verdadeiro & (SH_j > 0) \\ 0 \; ou \; Desconhecido & (SH_j = 0) \\ -1 \; ou \; Falso & (SH_j < 0) \end{cases}$$

$$O_k = \begin{cases} +1 \; ou \; Verdadeiro & (SO_k > 0) \\ 0 \; ou \; Desconhecido & (SO_k = 0) \\ -1 \; ou \; Falso & (SO_k < 0) \end{cases}$$

O valor da célula I_0 é sempre +1 e é conectada a cada uma das outras células, exceto para as células de entrada.

Então:

Passo I: Selecione uma célula de saída O_k;

Passo II: Selecione um grupo de célula. Se o grupo de célula selecionado é *fuzzy*, direcione os valores deste grupo para (+1,-1,-1), (+1,+1,-1) ou (+1,+1,+1); enquanto que se o grupo de célula selecionado for abrupto, direcione os valores deste grupo em (+1,+1,+1) ou (-1,-1,-1). Além disso, direcione o valor dos demais grupos, os quais não foram selecionados, para (0,0,0);

Passo III (Pesquisa Para Frente): Determine todos os valores das células intermediárias H_j usando os valores do grupo de célula dados no Passo II e:

$$H_j = \begin{cases} +1 \; ou \; Verdadeiro & (|SH_j| > USH_j \text{ e } SH_j > 0) \\ 0 \; ou \; Desconhecido & (|SH_j| \leq USH_j) \\ -1 \; ou \; Falso & (|SH_j| > USH_j \text{ e } SH_j < 0) \end{cases}$$

onde:

$$USH_j = \sum_{j:I_i \text{ e desconhecido}} |w_{ji}|$$

Além disso, determine o valor da célula O_k usando:

$$O_k = \begin{cases} +1 \textit{ ou } Verdadeiro & (|SO_k| > USO_k \text{ e } SO_k > 0) \\ 0 \textit{ ou } Desconhecido & (|SO_k| \leq USO_k) \\ -1 \textit{ ou } Falso & (|SO_k| > USO_k \text{ e } SO_k < 0) \end{cases}$$

onde:

$$USO_k = \sum_{i:I_i \text{ e desconhecido}} |u_{ki}| + \sum_{i:H_i \text{ e desconhecido}} |v_{kj}|$$

Se o valor de O_k é +1 ou -1, ir para o Passo V. Caso contrário (o valor de O_k é 0), ir para o Passo IV. Embora todos os grupos de células estejam em um antecedente (*se-parte*), se o valor de O_k é 0, não haverá estrutura de regras *fuzzy Se/Então* para a célula de saída O_k. Então, ir para o Passo VI;

Passo IV (Pesquisa Para Trás): Seja v^* o valor máximo de $|v_{kj}|$ o qual é um valor absoluto do peso das conexões entre a célula de saída O_k e a célula intermediária H_j, cujo valor de ativação é 0. Além disso, seja u^* o valor máximo de $|u_{ki}|$, o qual é um valor absoluto do peso das conexões entre a célula de saída O_k e a célula de entrada I_i, cujo valor é 0. Se $u^* \geq v^*$ ou valores de todas as células intermediárias são determinadas, ir para o Passo IV-1. Caso contrário, ir para o Passo IV-2;

Passo IV-1: Para a célula de entrada I_i, a qual é dependente a $u_{ki}(|u_{ki}| = U^*)$, se a célula de entrada I_i é incluída no grupo de célula *fuzzy*, ir para o Passo IV-1-F. Enquanto o grupo de célula abrupta, ir para o Passo IV-1-C;

Passo IV-1-F: Se $SO_k \geq 0$, selecione um padrão do grupo de célula *fuzzy*, o qual tem o valor máximo de SO_k entre (+1,-1,-1), (+1,+1,-1) e (+1,+1,+1). Inversamente, se $SO_k < 0$, selecione um padrão, o qual tem o valor mínimo de SO_k. Ir para o Passo V;

Passo IV-1-C: Se $SO_k \geq 0$, selecione um padrão do grupo de célula abrupta, o qual tem o valor máximo de SO_k em (-1,-1,-1) (+1,+1,+1). Inversamente, se $SO_k < 0$, selecione um padrão, o qual tem o valor mínimo de SO_k. Ir para o Passo V;

Passo IV-2: Seja w^* o valor máximo de $|w_{ji}|$, o qual é um valor absoluto do peso das conexões entre a célula intermediária H_j, a qual é dependente a $v_{kj}(|v_{kj}| = v^*)$ e a célula de entrada I_i, cujo valor de ativação é 0. Selecione a célula de entrada I_i, a qual é dependente a conexão $w_{ji}(|w_{ji}| = w^*)$. Se a célula de entrada I_i é incluída no grupo de célula *fuzzy*, ir para o Passo IV-2-F. Considerando que no grupo de célula abrupta, ir para o Passo IV-2-C;

Passo IV-2-F: Se $SH_j \geq 0$, selecione um padrão do grupo de célula *fuzzy*, o qual tem o valor máximo de SH_j entre (+1,-1,-1), (+1,+1,-1) e (+1,+1,+1). Inversamente, se $SH_j < 0$, selecione um padrão, o qual tem o valor mínimo de SH_j. Ir para o Passo V;

Passo IV-2-C: Se $SH_j \geq 0$, selecione um padrão do grupo de célula abrupta, o qual tem o valor máximo de SH_j em (-1,-1,-1) (+1,+1,+1). Inversamente, se $SH_j < 0$, selecione um padrão, o qual tem o valor mínimo de SH_j. Ir para o Passo V;

Passo V (Extração da estrutura de regras *Se/Então*): Se o valor de O_k é determinado, extrair os itens de entrada correspondendo a um grupo de uma determinada célula com as proposições em um antecedente (*Se-parte*). Aqui, se o valor de O_k é +1, o conseqüente é direcionado para O_k *é Verdadeiro.* Inversamente, se o valor de O_k é -1, o consequënte é direcionado para O_k, *que é Falso.* Se estruturas multiplas de regras *Se/Então* com o mesmo antecedente e conseqüente são extraídos, adote um deles;

Passo VI (Condição de finalização do algoritmo de extração para cada célula de saída): Para a célula de saída O_k, se há alguns grupos de células os quais ainda não são selecionados, ou para selecionar grupos de células, há alguns padrões, os quais ainda não são selecionados, ir para o Passo II. Caso contrário, ir para o Passo VII;

Passo VII (Condição de finalização de todo o algoritmo de extração) : Repetir os Passos II até VI exposto acima, até a condição de finalização do algoritmo de extração para cada célula de saída é satisfeita. Se há alguma célula de sída O_k a qual não foi ainda selecionada no Passo I, ir para o Passo I. Caso contrário, pare todo o algoritmo de extração.

O próximo e último algoritmo de extração de regras apresentado aqui, refere-se ao utilizado por S. Mitra et al. [134][133]. Eles consideram que as entradas, quando forem do tipo lingüísticas, são quantificadas na forma de conjuntos *fuzzy*-π, da seguinte forma:

$$low \equiv \left\{\frac{0.95}{l}, \frac{0.6}{m}, \frac{0.02}{h}\right\}$$

$$medium \equiv \left\{\frac{0.7}{l}, \frac{0.95}{m}, \frac{0.7}{h}\right\}$$

$$high \equiv \left\{\frac{0.02}{l}, \frac{0.6}{m}, \frac{0.95}{h}\right\}$$

Este algoritmo é aplicado à RNA com multiplas camadas e treinadas por retro-propagação. A geração das cláusulas é desenvolvida por *Backtracking*. O sistema responde com uma regra *Se/Então* aplicável para o caso em questão. Estas regras não se encontram explícitas na base de conhecimento. Elas são geradas pelo sistema de inferência dos pesos das conexões como necessidade para explanações. Uma conclusão particular, olhando uma dada saída j, é inferida dependendo de uma medida de certeza chamada bel_j. Ela é definida como:

$$bel_j = y_j - \sum_{i \neq j} y_i$$

Assim, os passos deste algoritmo pode ser descritos como sendo:

1. Geração das cláusulas por *backtracting*

   (a) Escolha do padrão Entrada-Saída;

   (b) Escolha do neurônio i com impacto positivo na conclusão de uma saída j;

   (c) Determinação do conjunto m_l em relação às entradas do neurônio escolhido;

$$m_l = \{a_1, a_2, \cdots, a_{m_l}\}$$

   (d) Determinação do conjunto dos pesos acumulados para o neurônio i;

$$y_i > 0.5$$

$$wet_i = max[wet_{a_k} + w_{a_k i}] \tag{1.10}$$

   (e) Determinação dos pesos wet_{a_k};

$$wet_{a_k} = \{w_{ja_1}, w_{ja_2}, \cdots, w_{ja m_i}\}$$

   (f) Seleção do conjunto dos neurônios de entrada m_o;

$$m_o = \{a_1, a_2, \cdots, a_{m_o}\}$$

   (g) Determinação dos pesos do caminho referente ao neurônio j;

(h) Ordenação em ordem decrescente de Impacto de Rede (IR) dos elementos do conjunto de pesos obtidos no item anterior como:

$$IR_i = y_i * wet_i$$

$$\sum_{i_s} wet_{i_s} > 2 \sum_{i_n} wet_{i_n} \tag{1.11}$$

(i) Seleção dos i_s neurônios de entrada e os i_n neurônios restantes para as cláusulas, tal que:

$$m_o = |i_s| + |i_n|$$

2. Geração das cláusulas

Dedução dos antecedentes:

(a) Determinação de u_{s_1} para uma entrada lingüística, é obtida como[11]:

$$u_{s_1} = (l_{s_1} - 1) \, mod \, 3 + 1$$

(b) Repetem-se os itens anteriores até que todos os $|l_s|$ sejam selecionados pela Equação 1.11.

Dedução dos conseqüentes:

(a) Determinação da medida de certeza bel_j, dada como:

$$bel_j = y_j - \sum_{i \neq j} y_i$$

(b) Determinação dos conseqüentes de uma regra através das seguintes propriedades:

$$prop = \begin{cases} baixo & \text{Se } l_{s_1} - 3(u_{s_1} - 1) = 1 \\ \text{médio} & \text{Se } l_{s_1} - 3(u_{s_1} - 1) = 2 \\ alto & \text{Caso Contrário} \end{cases}$$

onde $1 \leq l_{s_1} \leq 3n$, $1 \leq u_{s_1} \leq n$ e n é a dimensão do vetor do padrão de entrada.

3. Volta-se ao item 1 caso se deseje determinar as regras *Se/Então* para justificar todas as saídas apresentadas pelos padrões de Entrada-Saída do NNES vencedor.

[11] Entenda-se aqui a notação mod 3 como sendo o resto da divisão da expressão matemática $(l_{s_1} - 1)$ por 3.

Outras técnicas para a abordagem dada aqui podem ser encontradas em [138], como é o caso do NEFCON (*Neuro-Fuzzy Controller*) e o do NEFCLASS (*Neuro-Fuzzy Classification*). Outras referências ainda, são [4][139][142].

Capítulo 2

Fundamentação Teórica

Neste capítulo, apresentam-se os fundamentos teóricos, ou seja, as ferramentas básicas necessárias para o desenvolvimento deste trabalho. Portanto, para uma melhor clareza do assunto, este capítulo foi dividido nas seguintes seções: os sistemas simbólicos, os sistemas sub-simbólicos, os sistemas *fuzzy*, os AG e o problema do raciocínio médico.

2.1 Sistemas Simbólicos

SE são programas sofisticados, que manipulam conhecimento para resolução de problemas, em áreas restritas de um dado domínio. Da mesma forma, que os especialistas humanos, tais sistemas, se utilizam de lógica simbólica e regras oriundas da prática para encontrar as soluções. Entretanto, estes sistemas, para serem desenvolvidos necessitam de um conjunto de atividades, entre as quais, podem-se citar [161]:

- Aquisição de conhecimento (AC);
- Desenvolvimento de um modelo conceitual do conhecimento adquirido;
- Análise do modelo conceitual, para determinar requisitos para a implementação de um SE;
- Seleção ou desenvolvimento de uma linguagem ou um *Shell* para SE que satisfaça estes requisitos;
- Projeto e desenvolvimento de uma base, que implemente o modelo conceitual.

Entretanto, apesar de todas as dificuldades provenientes da implementação e manuseio do conhecimento de um especialista de domínio, os sistemas simbólicos mais difundidos, referentes à área de desenvolvimento de SE, ainda são os que se utilizam de regras de produção. Desta forma, a seguir, são descritas, com maiores detalhes,

algumas das atividades anteriormente citadas. O objetivo principal é mostrar a necessidade destes fatores para se alcançar um bom desempenho, no desenvolvimento dos chamados Sistemas Especialistas Baseados em Regras (RBES).

2.1.1 Fontes de Conhecimento

Antes de se entrar diretamente nos tipos de fontes de conhecimento, faz-se necessário entender o que seja os termos *informação, dados e conhecimento.* Conforme B.I. Blum [16][132], eles são definidos como:

Dados: São itens não interpretados fornecidos a um analista ou a um *solucionador de problemas*, como os sinais adquiridos por um eletrocardiógrafo ou um equipamento de imagem.

Informação: É uma coleção de elementos de dados organizados (ou interpretados) para conferir significado ao usuário, como os registros médicos automatizados ou um fluxograma.

Conhecimento: É a formalização das relações, experiências e regras, através da qual a informação é formada a partir dos dados, como dos sistemas baseados em conhecimentos.

Os tipos de fontes de conhecimento são vastas. Entretanto, a utilidade e valor de cada tipo varia de acordo com o escopo e foco do engenheiro do conhecimento e do projeto do SE ou Sistema Baseado em Conhecimento (SBC).

Em geral, o conhecimento pode ser considerado como público ou privado. Conhecimento público, ou o que se encontra publicado na literatura, é adquirido de livros, artigos, obras empíricas, registros de vídeo e audio, e apresentações. Estas fontes de conhecimento contêm conceitos gerais e teorias de um domínio. Conhecimento privado é o que é refinado e sintetizado para influenciar eventos em um domínio. O conhecimento heurístico é um tipo de conhecimento privado, isto é, são métodos de problemas em que se usa a tentativa e o erro, de modo que se consiga chegar a um objetivo final [187].

No entanto, SE e SBC diferem levemente um do outro. O desenvolvimento de um típico SE se inicia com um engenheiro de conhecimento que extrai o conhecimento de um especialista de um dado domínio usando várias técnicas de EC e codifica este conhecimento em regras e fatos. Depois de representado simbolicamente, o conhecimento elicitado é transportado para um computador que eletronicamente repete

as análises e as estratégias de solução de problemas. Um procedimento similar caracteriza o desenvolvimento de sistemas com base de conhecimento. Contudo, estes sistemas derivam seu conhecimento particular de outras fontes e incorporam tópicos de assuntos que não requerem aptidão ou educação especiais, como aqueles encontrados em livros ou manuais [71]. Isto implica que os SE podem ser considerados como casos particulares de SBC, enquanto que os SBC, o caso mais geral.

2.1.2 Base de Conhecimento

Conforme E.L. Passos [148], a base de conhecimento de um SE é o local onde se armazenam conhecimento, fatos, regras e informação a cerca do problema presente.

A base de conhecimento é o elemento que, além de ser o suporte, é também o elemento chave para o SE. A eficácia e o valor de um SE é largamente determinada por meio dela. A base de conhecimento inclui não só a identificação da solução do problema como as estratégias da solução. Estas estratégias de solução do problema são representadas por regras heurísticas. O especialista de domínio e o engenheiro de conhecimento têm condições de traduzir o conhecimento de domínio do especialista em regras, ou seja, por meio de fatos e de ações.

Os fatos intrinsecos a um especialista são coletivamente chamados de conhecimento declarativo. As ações constituem o conhecimento procedural. Juntos, os estados declarativos e procedurais, são a representação codificada das habilidades, experiências, intuições e treinamentos do especialista.

O valor de uma base de conhecimento reside na qualidade dos conteúdos e não na quantidade. A qualidade da base de conhecimento tem uma correlação direta com o conhecimento e regras heurísticas do especialista do domínio, além da habilidade do engenheiro do conhecimento em extrair e representar este conhecimento. Engenharia do conhecimento é uma arte que usa a ciência.

A base de conhecimento é um dos maiores interesses do engenheiro desta área. Além disso, ela pode evoluir com o tempo, o que requer atenção para as modificações e valorizações, bem como, trocas de condições no domínio do problema [187].

2.1.3 Aquisição de Conhecimento

É o processo de identificar, extrair, analisar e documentar o conhecimento obtido de um especialista de domínio, com o propósito de construir um SE ou SBC [18][38].

Elicitação de Conhecimento

A Elicitação de Conhecimento (EC) é uma das etapas da AC. Ela é a etapa mais importante, onde o engenheiro de conhecimento extrai o conhecimento do domínio do especialista [38].

O elicitante ou o engenheiro de conhecimento é quem obtém material de alguma fonte relevante, e analisa, interpreta e coloca em uma forma pré-codificada, que será útil para codificar o conhecimento em uma linguagem apropriada para SBC, e que permita uma investigação, por todas as partes interessadas, no desenvolvimento do sistema [18].

O produto da EC será a representação do conhecimento relevante para a tarefa e o processo que opera sobre os conhecimentos, que podem ser usados como uma base para o projeto e implementação de um SBC.

O engenheiro de conhecimento deverá ter uma visão clara do produto da elicitação e da estrutura do mundo real antes de iniciar a elicitação. Caso contrário, a EC é considerada ineficiente. Além disso, é necessário que se leve em conta a análise do conhecimento e a provável estrutura do conhecimento, antes de efetivamente iniciar o processo de EC.

Uma visão compartilhada do SBC é crucial para a elicitação. Os participantes do domínio (um ou mais especialistas, por exemplo) necessitarão conhecer em que atividades do desenvolvimento eles podem contribuir, além de ter uma idéia clara dos objetivos do SBC e como o conhecimento que está sendo investigado, auxiliará o empreendimento. O grupo de desenvolvimento deverá estabelecer credibilidade em relação ao pessoal do domínio.

O elicitante do conhecimento deverá mostrar que tem competência técnica para elicitar e analisar o conhecimento, bem como, o profissionalismo para fazê-lo sem colocar em perigo os interesses das pessoas, ou grupos do domínio, que cooperam na atividade. No primeiro contato com o especialista, o elicitante deverá deixar claro: o que é um SBC; suas implicações no ambiente de trabalho; quanto é requerido do

seu tempo e esforço; o que se espera dele; que o SBC terá desempenho limitado; será desenvolvido considerando de modo a ser respeitada a forma confidencial; como a cooperação afetará suas relações com seus colegas e seus superiores; qual será sua remuneração; e, finalmente, que o especialista tem a opção de cooperar ou não.

Em [18] está referenciado, com mais detalhes, quanto tempo o engenheiro de conhecimento deve levar em cada sessão, bem como deve proceder para adquirir o conhecimento de um especialista de um domínio.

Técnicas de Elicitação de Conhecimento

A importância da abordagem utilizada para adquirir conhecimento costuma determinar a qualidade do conhecimento do especialista bem como o esforço necessário para sua aquisição. Há várias técnicas de EC que têm sido desenvolvidas, as quais são:

Observação - Esta técnica consiste em observar como um especialista soluciona um problema real [117].

Entrevista - É uma técnica conhecida também como *questionamento*, uma vez que o tipo principal de interação é a colocação de perguntas pelo entrevistante (elicitante) e a resposta pelo entrevistado (especialista) [137].

Discussão Focalizada - É semelhante a técnica de entrevista, apenas diferenciando na introdução de um terceiro elemento, o foco, que interage com o elicitante e a fonte humana de conhecimento [51].

Análise de Protocolo - Requer que o especialista *pense em voz alta* enquanto soluciona um problema [35].

Geração de Matriz - É uma técnica que faz uso de tabelas (matrizes) quando do processo de EC, em especialistas que costumam tabular seu conhecimento [137].

Ordenamento de Cartões - É outra técnica a qual visa gerar um mapa das estruturas conceituais usadas pelo especialista [130].

Análise de Discurso - É quando as sessões de entrevistas são gravadas, transcritas

e analisadas [51].

Análise Comportamental para Solução de Problema Clínico - Esta técnica simula uma sessão clínica para obtenção da tomada da história de uma doença real [107].

Em [18][11] estão referenciados, com maiores detalhes, várias outras técnicas para o processo de EC.

Representação de Conhecimento

Representação do conhecimento consiste nos métodos usados para modelar os conhecimentos de especialistas em algum campo, de forma eficiente, e deixando-os prontos para serem acessados pelo usuário de um sistema inteligente [157].

A representação de conhecimento tem por objetivo sustentar o processamento necessário para chegar-se a uma conclusão ou para fornecer opções para selecionar uma solução [18].

O engenheiro de conhecimento pode representar o conhecimento do domínio, usando várias técnicas de representação de conhecimento. Estas opções incluem, no caso de sistemas simbólicos, regras de produção, redes semânticas, frames, lógica fuzzy e outras, enquanto que, para os sistemas sub-simbólicos, também chamados de sistemas conexionistas, pode-se citar as RNA.

Além disso, estas técnicas podem ser combinadas com técnicas para o tratamento da incerteza como, por exemplo, FC, Teoria da Crença ou Evidência (Teoria de Dempster-Shaffer) e Métodos Probabilísticos (Métodos Bayesianos).

Regras de produção

As regras de produção são um dos tipos mais comuns e um dos mais usados para representação de conhecimento em SE. Um dos motivos se deve ao fato que este tipo de estrutura é muito parecida com o modo das pessoas falarem sobre como resolvem seus problemas, ou seja, o fato de parecer natural, ao humano, usar esta forma de representação de conhecimento para raciocinar e decidir. Um exemplo prático é o sistema MYCIN [164], o qual usa este tipo de técnica em sua implementação.

A seguir são descritos alguns pontos relevantes no desenvolvimento de um SE por regras de produção.

Um SE, que envolva como representação de conhecimento regras de produção, pode conter regras ortogonais e não-ortogonais. Num conjunto de regras que exprimem o conhecimento de um especialista de um determinado domínio, se uma regra pode ser deduzida de uma outra, ela é considerada não-ortogonal, caso contrário, é dita ortogonal[1]. Além disso, certas regras não são redundantes, caso este em que se um conjunto de regras pode ser expresso através de um conjunto menor de regras ou que uma regra possa ser provada a partir de outras, é possível concluir que aquele conjunto de regras é um conjunto mínimo para representar aquele conhecimento. É possível também, quando do desenvolvimento de um SE por regras de produção, fazer um programa para descobrir se as regras contidas em uma base de conhecimento representada por regras possuem regras mínimas ou não. Uma maneira de se verificar isto poderia ser da seguinte forma: tomar regra por regra e tentar descobrir se, por exemplo, uma delas pode ser deduzida de uma outra, ou melhor, se esta é verdadeira ou falsa, imaginando que todas as outras foram dadas. Caso se consiga demonstrar que alguma regra é *verdade*, conhecendo as outras, é porque esta não seria necessária. Então, este conjunto de regras não seria mais um conjunto mínimo. Caso contrário, seria considerado um conjunto mínimo de regras. Desta forma, poder-se-ia inferir que o conhecimento elicitado de um dado especialista de um domínio estaria contindo nesta base de conhecimento como sendo o mínimo e que esta base de conhecimento não poderia ser reduzida para menos do que ela foi construída inicialmente, porque estar-se-ia perdendo as informações iniciais cedidas pelo especialista.

O raciocínio, usando-se o modelo de representação de conhecimento por regras de produção, envolve o gerenciamento da base de regras, a manutenção do contexto em que o sistema está inserido, a interpretação adequada das regras e a aplicação de algum algoritmo de solução [154].

Quanto ao gerenciamento da base de regras, este envolve o controle do conjunto de fatos, o acréscimo e retirada de regras e a verificação de sua consistência. Já,

[1]Talvés aqui haja um sentindo duplo com respeito a palavra *ortogonal*. No contexto deste trabalho, se está referenciando à ortogonalidade de regras e não à ortogonalidade de vetores. É sabido que não existe o conceito de ortogonalidade de ângulo, mas que é possível se dizer que uma regra é dependente ou não de outras, ou seja, é um conceito de Linearmente Dependente (LD) ou Linearmente Independente (LI) generalizado. Então, o conceito de ortogonalidade de regras, para sermos mais puristas, pode ser trocado por: um conjunto mínimo de regras, ou seja, aquelas regras que são consideradas como LI generalizadas.

o interpretador de regras, age ciclicamente, sendo que, em cada ciclo, descobre as regras aplicáveis, escolhe a regra que será utilizada, resolve conflitos e dispara as ações resultantes da utilização da regra escolhida. Por fim, conectado ao algoritmo de solução, devem haver recursos para se optar entre as diversas formas de busca, como é o caso da busca para trás (*backward*), para frente (*forward*) ou ambas.

Contudo, no raciocínio que utiliza o modelo de representação de conhecimento através de regras de produção, dois aspectos devem ser ainda ressaltados: a estratégia de controle e a resolução de conflitos. A estratégia de controle é uma imposição relacionada com o tamanho do sistema, enquanto que a resolução de conflitos está relacionada mais com o tipo de árvore (ou grafo) de inferência produzida. Assim, como sugestão, para auxiliar os aspectos problemáticos acima abordados, ver a referência [154], pois este assunto, foge do escopo deste trabalho.

Por fim, neste tipo de representação de conhecimento, os conhecimentos são codificados através de pares de *Condição-Ação.* As regras (base de conhecimento), têm duas partes: uma antecedente ou premissa ou cauda (*Se*) e outra conseqüente ou cabeça da cláusula (*Então*).

A Parte *Se* é formada por condições as quais são conectadas por meio de palavras conhecidas como conectivos lógicos, como E, OU e NÃO. A parte *Então* é avaliada apenas se a parte *Se* for verdadeira. A combinação de nós de decisão conectados e um nó de conclusão representam uma regra *Se/Então.*

Os demais tipos de representação simbólica, como é o caso dos *frames*, redes semânticas e outras, não serão abordadas neste trabalho, pois foge dos interesses deste. Assim, veja as referências [148][154][157]. Entretanto, quanto a representação de conhecimento, por meio de lógica *fuzzy* e RNA, estas serão abordadas nas próximas seções.

2.1.4 Grafos E/OU

A teoria dos grafos, surgiu independentemente nas diversas áreas do conhecimento, mas pode ser inteiramente considerada como uma área da matemática. A mais antiga menção ao assunto ocorreu no trabalho de Euler, no ano de 1736, considerado como uma simples situação de adivinhação. O problema é análogo aos atuais quebra-cabeças dados às crianças. Baseia-se em um desenho, cujas linhas devem ser percorridas sem que se tire o lápis do papel e sem passar duas vezes sobre a mesma

linha. Além dele, pode-se citar, ainda no século XIX, Kirchhoff e Cayley, que desenvolveram a teoria das árvores; Hamilton, 1859, com o problema de percorrer as arestas de um dodecaedro regular, de tal modo que cada vértice fosse percorrido exatamente uma única vez; Appel e Haken, 1977, com o celébre problema das quatro cores, e ainda outros [153].

Nos dias de hoje, a teoria dos grafos proporciona uma ferramenta simples, acessível e poderosa para construção de modelos e resolução de problemas, relacionados com arranjos de objetos discretos. Além disso, ela serve como um modelo matemático para qualquer sistema envolvendo uma relação binária.

Por ser um dos mais simples e mais elegantes assuntos da matemática moderna, a teoria dos grafos possui uma variedade de aplicações. Baseada na simples idéia de pontos, interligados por linhas, ela combina estes ingredientes básicos, em um rico sortimento de formas, e dota estas formas, com propriedades flexíveis, fazendo com que esta teoria seja uma ferramenta útil para estudar vários tipos de sistemas.

Conforme C.F. Liaw [115], grafos E/OU podem ser conceituados, de acordo com a teoria clássica de grafos, representando-se uma decomposição de um problema complicado, em uma seqüência de sub-problemas mais simples. Assim, seja G um grafo E/OU, onde N(G) é o conjunto de nós em G e A(G) é o conjunto de arcos direcionados em G. Então, $A(G) \subseteq N(G) \times N(G)$. O problema inicial dado é representado por um único nó em G, chamado de nó de partida s. Os subproblemas, representando os problemas primitivos, correspondem ao conjunto finito de nós em G, chamado de conjunto de meta Γ. Os outros nós, em G, representam estados de problemas diferentes e podem ser classificados em um dos dois tipos: OU ou E. Um nó OU representa um subproblema, o qual, pode ser solucionado, se um dos subproblemas antecessores adjacentes for solucionado, isto é, um nó OU n é solucionado, se algum nó, no conjunto antecessor adjacente, for solucionado. Um nó E corresponde a um subproblema, o qual é resolvido somente se cada um dos subproblemas antecessores adjacentes for solucionado[2] [115].

Desta forma, para o problema específico deste trabalho, são feitas algumas considerações correspondentes as proposições mencionadas acima, a saber:

a) O problema de espaço-estado, representado como um grafo E/OU, direcionado

[2]O desenvolvimento matemático de grafos E/OU, pode ser visto em [115][153].

e finito, onde os estados são os nós no grafo;

b) Mais de um nó de partida no grafo;

c) Um conjunto finito de nós de meta no grafo;

d) Um vetor de valor de custo, associado com cada arco no grafo (positivos e negativos);

e) Para cada nó é usado um conjunto de vetores de funções heurísticas, o qual, provê informações nos custos a serem acumulados, ao longo de uma das partes de um grafo de solução.

A Figura 2.1, mostra um exemplo de Grafo E/OU, onde:

A, B, C, D, E, F, G, H - Nós de partida
P, Q, R - Nós E
X, Y - Nós OU
V1 a V14 - Vetores de valor de custo

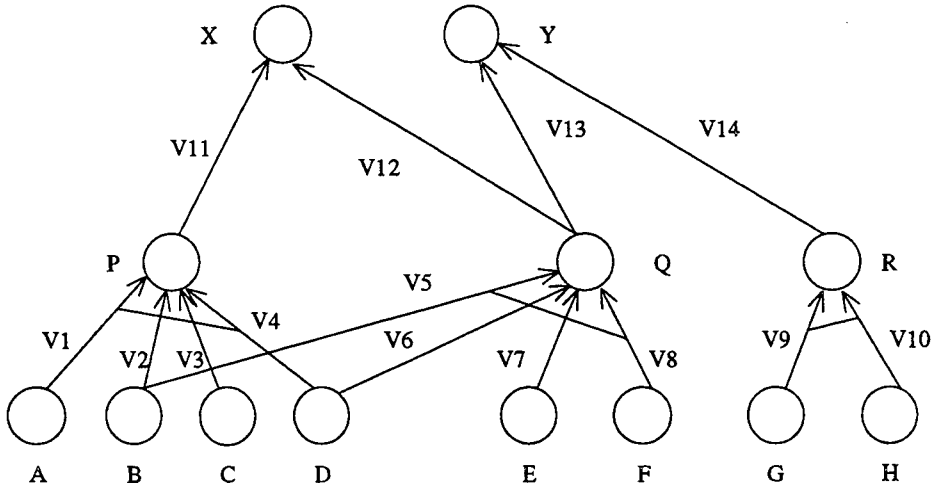

Figura 2.1: Grafo E/OU

Além do que já foi exposto, sobre a utilidade de grafos E/OU, estes também, são utilizados na implementação de algoritmos de busca, como uma técnica de busca heurística [141][157]. Contudo, neste trabalho, este tipo de ferramenta será utilizado apenas como um passo intermediário, para auxiliar na solução de um dos problemas de AC. O desenvolvimento matemático de algoritmos de solução de busca não será

detalhado neste projeto, mas podem ser encontrados em [115][141][153].

2.1.5 Cláusulas de Horn

Cláusula é uma disjunção de literais, enquanto as cláusulas de Horn são definidas como as cláusulas que contêm, no máximo, uma conclusão [110].

Contudo, antes de estudar especificamente as cláusulas de Horn, para melhor compreensão do trabalho, cabe esclarecer que, uma cláusula pode ser obtida desde que as fórmulas estejam na Forma Normal *Prenex*[3]. Esta forma foi introduzida por Davis e Ptunam em 1960 [71].

A forma normal *prenex* é definida como sendo uma fórmula F na lógica de primeira ordem, onde são retirados todos os quantificadores existentes, que prefixem a fórmula, isto é, se e somente se estiver na forma $Q1x1 \cdots Qnxn(M)$.

Onde:

Qixi = $\forall$ xi ou $\exists$ xi.
(M) = uma fórmula que não contenha quantificadores.

Os procedimentos normais, para o uso da forma normal *prenex*, são dados como a seguir:

Passo 1: Eliminar os conectivos lógicos $\rightarrow$ e $\leftrightarrow$ usando as seguintes leis:

$F \leftrightarrow G = (F \rightarrow G) \wedge (G \rightarrow F)$
$F \rightarrow G = \sim F \vee G$

Passo 2: Repetir o uso das seguintes leis como um meio de facilitar a obtenção da forma normal *prenex*:

Lei de De Morgan:

$\sim (F \vee G) = \sim F \wedge \sim G$

[3]É usada a forma *prenex* em qualquer fórmula para facilitar os procedimentos de prova.

$\sim (F \wedge G) = \sim F \vee \sim G$

Leis da Negação:

$\sim (\sim F) = F$
$\sim (\forall x Fx) = \exists x \sim Fx$
$\sim (\exists x Fx) = \forall x \sim Fx$

Passo 3: Renomear as variáveis limitantes, se necessário.

Passo 4: Usar as leis abaixo, para mover os quantificadores à esquerda da fórmula total, obtendo-se a forma normal *prenex*.

$(Qx)Fx \vee Q = Qx(Fx \vee Q)$
$(Qx)Fx \wedge Q = Qx(Fx \wedge Q)$
$(\forall x)Fx \wedge (\forall x)Qx = (\forall x)(Fx \wedge Qx)$
$(\exists x)Fx \vee (\exists x)Qx = (\exists x)(Fx \vee Qx)$
$(Q1x)Fx \vee (Q2x)Qx = (Q1x)(Q2z)(Fx \vee Qz)$
$(Q3x)Fx \wedge (Q4x)Qx = (Q3x)(Q4z)(Fx \wedge Qz)$

Além disso, quando uma fórmula está na forma normal *prenex*, deve-se eliminar os quantificadores existenciais por uma função, se as variáveis estiverem no escopo do quantificador universal, caso estejam fora, substitui-se por uma constante. Assim, as constantes e funções, usadas para substituir as variáveis existenciais, são chamadas de Funções de Skolem[4].

Por exemplo, dado que: $\forall x \exists y P(x, y)$

Skolemizada: $\forall x P(x, f(x))$

A seguir, ão mostrados os passos necessários para se chegar a uma cláusula. Dado que:

$\forall x \forall y(\exists z(P(x, z) \wedge P(y, z)) \rightarrow \exists u Q(x, y, u))$

[4]As funções de Skolem são formadas utilizando-se símbolos funcionais, ainda não empregados.

Passo 1: Passar para a forma normal *prenex*

Eliminação do Condicional
$\forall x \forall y(\sim \exists z(P(x,z) \wedge P(y,z)) \vee \exists u Q(x,y,u))$
Lei de De Morgan
$\forall x \forall y(\sim \exists z(P(x,z) \vee \sim P(y,z)) \vee \exists u Q(x,y,u))$
Lei da Negação
$\forall x \forall y(\forall z \sim (P(x,z) \vee \sim P(y,z)) \vee \exists u Q(x,y,u))$
Todos os Quantificadores Prefixados
$\forall x \forall y \forall z \exists u(\sim P(x,z) \vee \sim P(y,z) \vee Q(x,y,u))$

Passo 2: A variável quantificada existencialmente, deve ser substituída pela função de Skolem.

$$\forall x \forall y \forall z(\sim P(x,z) \vee \sim P(y,z) \vee Q(x,y,f(x,y,z))$$

Passo 3: Abandona-se os quantificadores prefixados, obtendo-se a cláusula.

$$\sim P(x,z) \vee \sim P(y,z) \vee Q(x,y,f(x,y,z))$$

Uma vez, conceituados e especificados os passos necessários para se chegar a uma cláusula, retomamos as cláusulas de Horn, denominadas primeiramente, pelo matemático Alfredo Horn em 1951, que demonstrou que um problema tanto pode ser expresso em lógica na forma clausular como em cláusulas de Horn[5].

Uma expressão é dita ser uma cláusula de Horn quando está na forma clausular e, o conseqüente é constituído por, no máximo, uma fórmula positiva (conclusão) [148].

A maioria dos formalismos, para programação computacional, conduzem a uma analogia às cláusulas de Horn. Além disso, a maioria dos modelos de resolução de problemas, os quais têm sido desenvolvidos em IA, podem ser vistos, como modelos para problemas expressos por meio de cláusulas de Horn.

As cláusulas de Horn são um subconjunto importante da forma clausal. Além disso, os métodos de inferência, para este tipo de cláusulas, têm uma resolução de

[5]As cláusulas de Horn são consideradas um subconjunto do cálculo de predicados, e serviu como base para formação da linguagem de programação PROLOG [7].

problemas simples e interpretação de programação computacional. É importante salientar, que embora as cláusulas não-Horn possam ser dispensáveis na teoria, elas são indispensáveis na prática. Além disso, a extensão dos métodos de resolução de problemas de cláusulas de Horn para a forma clausal, em geral, é uma extensão significante dos modelos mais simples de resolução de problemas, os quais, hoje, são mais populares.

Assim, exemplificando-se a transformação para uma cláusula de Horn, tem-se que:

$$P \rightarrow Q \, e \, R$$

Então, para passar a uma cláusula de Horn, é substituída por:

$$P \rightarrow Q$$

$$P \rightarrow R$$

Assim, estas fórmulas são equivalentes à fórmula anterior.

2.2 Redes Neurais

Nas últimas décadas, muita atenção tem sido devotada ao estudo do chamado paradigma conexionista, o qual, objeta que é virtualmente impossível transformar em algoritmos os processos de raciocínio. Isto é, reduzir a uma seqüência de passos lógicos ou aritméticos, diversas tarefas que a mente humana executa com facilidade e rapidez, como reconhecer rostos, compreender e traduzir línguas ou evocar uma memória pertinente a determinada situação. O desafio estaria, portanto, em desenvolver e implementar num computador modelos cognitivos que reproduzissem as capacidades naturais do cérebro humano. Só assim seria possível simular tarefas, como as de associação, categorização e percepção de traços marcantes, naturalmente realizadas pelo homem. É a capacidade que nosso cérebro tem de se auto-organizar, portanto, que deve ser reproduzida pelo processo computacional.

Para que isso seja possível, a informação inteligente deve ser processada artificialmente, similar aquela processada no cérebro, isto é, por um conjunto de elementos computacionais simples - chamados neurônios artificiais - em analogia com as células do sistema nervoso, em várias vias paralelas. Então, os neurônios artificiais, distribuídos no espaço e ligados por conexões - chamadas sinapses - trocariam sinais

inibitórios ou excitatórios, competindo ou cooperando entre si. O comportamento inteligente emergiria da ação simultânea dessa coletividade, sem a necessidade de elementos centralizadores.

O estudo dos processos computacionais, que podem ser realizados por sistemas dinâmicos, que obedecem a esse paradigma, é chamado de neurocomputação. Os modelos cognitivos propostos são chamados de Redes Neurais Artificiais (RNA), ou também, modelos conexionistas.

2.2.1 Morfologia do Neurônio Biológico

O sistema nervoso é constituído de cerca de duzentos bilhões de células de dois tipos: as chamadas células gliais - supõe-se que atuam principalmente como elemento de suporte e nutrição - e os neurônios, células especializadas em receber a informação proveniente, seja do próprio corpo, seja do ambiente externo, integrá-la e retransmiti-la a outras células. São eles que servem de modelo para as RNA [105].

Os neurônios apresentam, além de um corpo celular - chamado soma -, revestido por uma membrana, uma complexa rede de prolongamento: são os dendritos, que podem ser muitos e ramificados, e o axônio, geralmente único (Figura 2.2). O corpo do neurônio mede apenas alguns milésimos de milímetro e os dendritos têm poucos milímetros, mas o axônio, pode ser bem mais longo (em torno de 10cm) e, em geral, tem calibre uniforme [11][105].

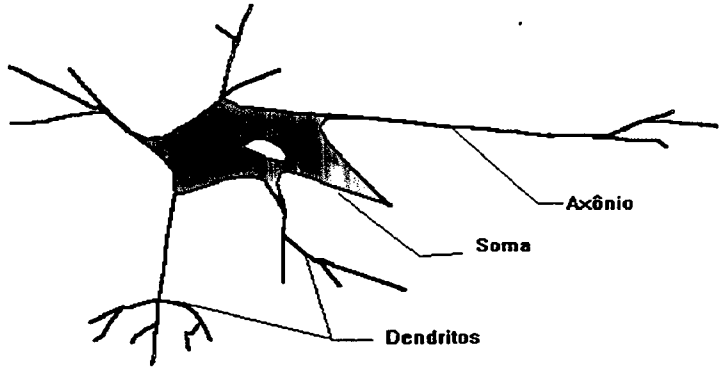

Figura 2.2: Esquema simplificado de um neurônio

Os dendritos têm por função receber as informações ou impulsos nervosos de

outros neurônios e conduzi-los ao corpo celular. Ali, a informação é processada e geram-se novos impulsos, que o axônio retransmitirá a outro neurônio, com o qual mantém contato, geralmente, através de um de seus dendritos [11][78].

A sinapse aparece como um ponto de contato entre a terminação axônica de um neurônio e o dendrito de outro neurônio. Além disso, é pelas sinapses que os neurônios se unem funcionalmente formando a RNA. Convém lembrar, que um único neurônio pode ter de mil a dez mil sinapses, e pode ser conectado com outros mil a dez mil neurônios, formando circuitos complexos. Além disso, as sinapses funcionam como válvulas, sendo capazes de controlar a transmissão de impulsos, isto é, o fluxo da informação entre os neurônios na RNA. Essa capacidade de regulação é chamada de eficiência sináptica.

Assim, o soma recebe a informação sináptica e desempenha o processamento de informação. Além disso, para que possam desempenhar suas funções, corpo celular, dendritos, axônios e sinapses, devem possuir determinadas características quanto ao modo de gerar, conservar e transmitir impulsos nervosos. Deve-se lembrar que os neurônios são preenchidos internamente e ao redor por substâncias químicas, principalmente íons de sódio (Na^+), cálcio (Ca^{++}), potássio (K^+) e cloro (Cl^-). Entre o interior (K^+) e o exterior (Na^+) do neurônio, separados pela membrana, existe uma diferença de potencial elétrico, o qual é conhecido como potencial de membrana [11].

Quando o neurônio está em repouso, o potencial de membrana se mantém constante, a cerca de $-60mV$, o que significa que a voltagem no interior do neurônio é negativa em relação à do meio extracelular. Esse valor da voltagem da célula, em repouso, é chamado de potencial de repouso. Se a voltagem do citoplasma, se torna mais positiva que o potencial de repouso, diz-se, que a célula está despolarizada; quando a mudança ocorre na direção oposta, fala-se de hiperpolarização [76][77][78].

O concomitante elétrico, de um impulso nervoso, é o potencial de ação. O processo de gerá-lo, em um neurônio ou nos axônios, é devido a troca de íons (K^+ e Na^+), causado por uma mudança na permeabilidade da membrana celular. Se a estimulação for suficientemente intensa e prolongada, gera-se no soma e se transmite ao axônio, o chamado trem de impulsos - a emissão de uma série de potenciais de ação, numa freqüência determinada pelas características da célula [11].

Assim, como os axônios codificam informação em freqüência, os dendritos são

especializados em decodificá-la. A chegada de um potencial de ação à sinapse, provoca uma leve alteração de sua voltagem, chamada potencial pós-sináptico. Como a voltagem retorna, lentamente ao estado de repouso, impulsos que chegam dos mais diversos neurônios podem ser integrados, pois, os dendritos somam os efeitos de potenciais de ação, deflagrados em tempos diferentes.

Existe um espaço entre duas células chamado *espaço (gap) sináptico* (cerca de 50 a 200 Angstroms). Um neurônio comunica-se com outro pela transferência de energia química do axônio para o dendrito de uma célula a outra, através de produtos químicos específicos, chamados *neurotransmissores* no espaço sináptico. Estes neurotransmissores despolarizam a membrana dendrítrica e aumentam a possibilidade que o neurônio alvo continuará a transmissão [11][158].

Normalmente, quando um impulso elétrico chega a um terminal do neurônio, ele não passa à membrana pós-sináptica. Em vez disso, ocorre a liberação das moléculas neurotransmissoras, que são armazenadas nas vesículas dos neurotransmissores químicos. Os neurotransmissores atravessam a membrana pós-sináptica para interagir com os receptores na célula pós-sináptica. É a partir da interação de neurotransmissores com receptores, que é produzido uma atividade elétrica no neurônio pós-sináptico [105].

De especial importância para a elaboração de RNA, é a existência de dois tipos de sinapses: as excitatórias - cujo efeito é despolarizar o potencial pós-sináptico - e as inibitórias, que hiperpolarizam esse potencial. As primeiras, permitem a passagem de informação entre neurônios; as segundas, envolvem o bloqueio da atividade de uma célula por outra, impedindo ou dificultando a passagem da informação.

2.2.2 Modelo Básico de um Neurônio

A primeira tentativa de reproduzir o funcionamento de neurônios, em um modelo cognitivo, foi empreendida por Waren McCulloch e Walter Pitts, em 1943 [129]. Eram os chamados neurônios lógicos, que podiam ser conectados formando-se diferentes tipos de RNA [196].

Generalizando o modelo do neurônio de McCulloch - Pitts, tem-se o modelo geral de neurônio. Veja a Figura 2.3.

O neurônio geral tem um conjunto de n entradas X_j, onde o subscrito j abrange

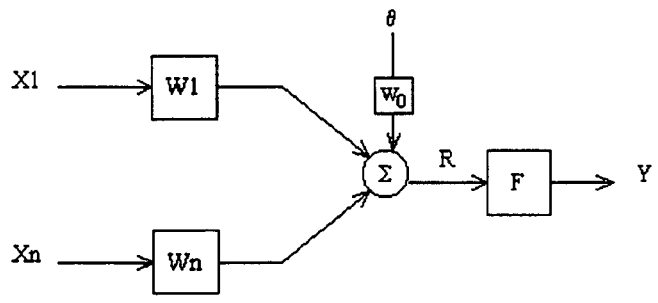

Figura 2.3: Neurônio artificial

valores de 1 a n e indica o tipo de sinal de entrada. Cada entrada X_j é multiplicada por um fator de peso W_j. Então, X_jW_j são combinadas usando-se uma função, que pode ser uma soma ponderada, um produto, ou outro tipo de função, se bem que é mais raro de acontecer para produzir um estado de ativação, que através da função F_j vai produzir a saída do neurônio Y_j. Além disso, há um termo *bias* θ, que tem que ser alcançado ou excedido para o neurônio produzir um sinal, uma função não-linear F_j, que age no sinal produzido, isto é, a ativação R_j, e uma saída Y_j, após a função não-linear. Y_j constitui, também, entrada para outros neurônios. Quando o neurônio é parte de uma rede de muitos neurônios, ele é referido como um nó. Então, para m nós, em uma rede, um subscrito adicional i é necessário para distinguir um neurônio simples. Entradas, pesos, sinais de ativação, saída, *bias* e função não-linear, são escritos como X_{ij}, W_{ij}, R_i, Y_i, θ_i, F_i, respectivamente.

A função de transferência deste modelo é descrita pela relação:

$$Y_i = F_i\left(\sum_{j=1}^{n} W_{ij}X_{ij} - \theta_i\right)$$

Enquanto que a condição de exitação do neurônio é:

$$\sum_{j=1}^{n}(W_{ij}X_{ij}) \geq \theta_i$$

Além disso, a função não-linear[6] tem por objetivo assegurar que a resposta do neurônio seja limitada, isto é, a resposta real do neurônio é condicionada como uma resposta de estimulo de ativação, grande ou pequeno, e daí é controlada. Porém, a função não-linear, usada em muitos paradigmas, não é necessariamente uma réplica daquela de um neurônio biológico. Freqüentemente, é meramente usada por conveniência matemática. Assim, funções não-lineares diferentes são utilizadas, dependendo do paradigma e do algoritmo a ser tratado. Algumas não-linearidades, que podem ser citadas, são: função sigmóide, função rampa, função degrau [196].

Contudo, o modelo apresentado até o presente momento, considera o neurônio como estático. Porém, há casos em que se deseja que o neurônio seja dinâmico. Para tanto, a Figura 2.4 apresenta uma possibilidade de aplicação de dinâmica. Entretanto, o neurônio dinâmico não será abordado neste trabalho, porque está fora do escopo dele. Para maiores detalhes, tem-se a referência [42].

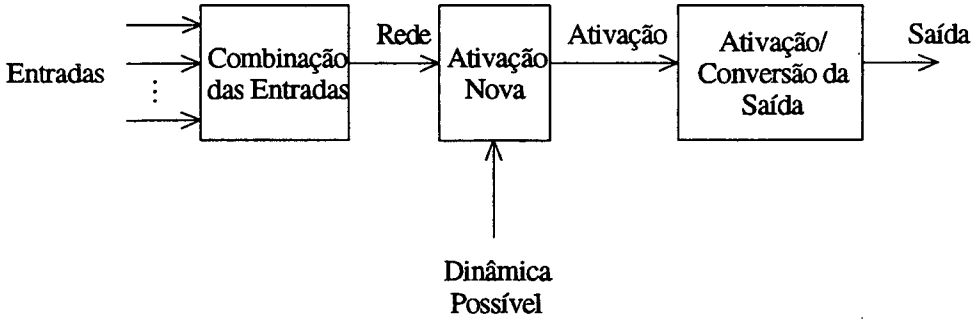

Figura 2.4: Diagrama de blocos de um neurônio dinâmico

2.2.3 Topologias das RNA

Similarmente aos sistemas biológicos, os quais, são mais ou menos dispostos em um emaranhado de arquiteturas específicas, assim também, as RNA usam arquiteturas diferentes para tipos diferentes de tarefas.

A arquitetura da rede consiste de três componentes: o número de elementos de processamento, o modo como as conexões são feitas e a neurodinâmica (função so-

[6]Também conhecida por função de ativação não-linear, conforme [76] e [134].

matório, função de transferência e regras de aprendizado) dos neurônios.

As RNA são compostas de neurônios artificiais, os quais são chamados de elementos de processamento ou, apenas, de neurônios. Cada um dos neurônios recebe entrada(s), processa a entrada(s) e endereça a uma única saída. A entrada pode vir de vários tipos de dados naturais (por exemplo, valores de pixeis de caracteres, digitalização de imagem e padrões de vozes) ou, até mesmo, da saída de outros neurônios. Quanto à saída da RNA, ela pode ser o produto final, isto é, a solução para um problema, ou uma entrada para outro neurônio [186]. Além disso, os neurônios de uma RNA podem ser agrupados em camadas. A Figura 2.5, mostra um exemplo de RNA com três camadas. Tipicamente, a camada onde os padrões de entrada são aplicados é a de entrada, a camada onde a saída é obtida é a de saída, e, por fim, a camada entre as de entrada e saída é a intermediária, a qual pode ser também mais de uma, dependendo do objetivo da aplicação da RNA.

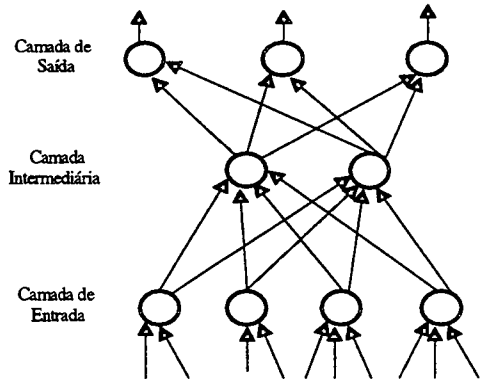

Figura 2.5: RNA com uma camada intermediária

Uma conexão é uma linha única de comunicação, que vai de um neurônio - que recebe uma informação -, a outro, que envia esta mesma informação. Há dois tipos de conexões indo a um neurônio: excitatória e inibitória. As conexões inibitórias tendem a prevenir a ativação do neurônio, enquanto as conexões excitatórias tendem a causar a ativação do neurônio. Assim, em uma RNA, estas conexões são representadas por uma matriz de pesos (W), as quais, são considerados como um dos elementos chaves em uma RNA. Os pesos expressam a força relativa, isto é, o valor matemático dos dados de entrada iniciais. Em outras palavras, os pesos exprimem

a importância relativa de cada entrada a um elemento de processamento.

Já, a função somatório utiliza a soma algébrica ponderada de todos os elementos de entrada a cada elemento de processamento, ou seja, a função somatório multiplica cada valor de entrada por seu padrão de conexão (peso), e totaliza-os para obter um somatório de $W_i X_i$, cujo objetivo é computar a estimulação interna ou nível de ativação (função ativação) do neurônio.

Baseado no nível de ativação, o neurônio pode ou não exibir uma saída. A relação entre o nível de ativação interna e a saída pode ser linear ou não. Tais relações são expressas por uma função de transferência, as quais podem ser de vários tipos diferentes. Além disso, a seleção da função específica determina a operação da rede, juntamente com o objetivo de modificar os níveis de saída para um valor razoável. Esta transformação é feita antes que a saída alcance o próximo nível. Entretanto, sem tal transformação o valor da saída pode ser muito grande, especialmente quando várias camadas são envolvidas. Normalmente, esta transformação ocorre na saída de cada neurônio ou na saída final da rede [186].

Todavia, o número de neurônios e como eles são agrupados nas camadas determina o quanto de detalhes a rede pode manipular. Enquanto isso, a neurodinâmica afeta os modos nos quais a rede processará os dados, bem como é relacionada, freqüentemente, à estratégia de conexão usada. Como mencionado anteriormente, as conexões podem transferir informação, primeiramente, em uma única direção (direta ou *feedforward*) ou em ambas as direções (com ciclos ou *feedback*). O método utilizado tem um enorme efeito na dinâmica de como a rede opera. Assim uma RNA, devido a maneira como as informações são processadas nas suas conexões, pode ser representada por um grafo orientado rotulado [42].

As redes diretas são aquelas cujo grafo não tem ciclos - os sinais somente vão por um caminho e as saídas são somente dependentes do somatório dos sinais provenientes dos outros neurônios - o que faz com que não haja *loops* no sistema. Uma vez treinadas, estas redes sempre produzem a mesma saída em resposta a mesma entrada. Além disso, é freqüentemente comum representá-las em camadas e, neste caso são chamadas de redes em camadas.

A Figura 2.6, mostra uma rede direta com três (3) camadas de neurônios, onde as conexões sinápticas são representadas apenas por setas. Os primeiros modelos

desenvolvidos foram do tipo direta linear. Em 1972, dois trabalhos simultâneos, desenvolvidos independentemente, por J.A. Anderson [3], um neurologista, e Teuvo Kohonen [108], um engenheiro eletricista, propuseram o mesmo modelo para uma memória associativa, o associador linear. Hoje, as RNA mais comumente difundidas são as redes diretas não-lineares, principalmente, por existirem métodos de aprendizado fáceis de usar para estas redes. Um método bastante usado é o de retropropagação. Além disso, estas redes são capazes de aproximar, com maior ou menor precisão, dependendo do número de neurônios da rede, qualquer função não-linear [113][10].

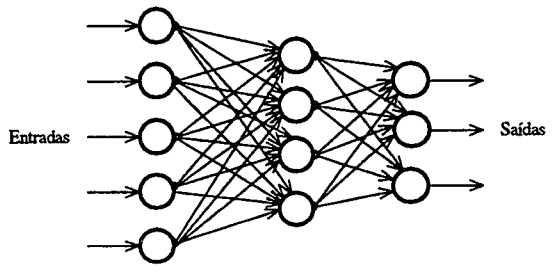

Figura 2.6: Rede direta

Quanto as redes com ciclos, são aquelas cujo grafo contém ao menos um ciclo (Figura 2.7). Neste tipo de rede, a matriz de peso (matriz sináptica) é criada por adicionar ao produto interno de cada vetor de padrão de entrada com ele mesmo ou com uma entrada associada. Após a construção da rede, um padrão de entrada inexato ou parcial, pode ser apresentado à rede, e, após um tempo, a rede pode eventualmente convergir para um dos padrões originais da entrada. Como exemplo, podem-se citar as redes de Hopfield [101], a BAM (*Memórias Associativas Bidirecionais*) [109] e as IAC (*Interactive Activation and Competition*) [128]. Na referência [118], tem-se maiores detalhes, pois este assunto foge do escopo deste trabalho.

2.2.4 Aprendizado em RNA

Aprendizado em RNA é um processo altamente importante e, ainda nos dias de hoje, continua a ser submetido a intensas pesquisas em ambas abordagens relacionadas às redes biológicas e neurais. Sabe-se, portanto, que o aprendizado não é um processo único. Há diferentes processos de aprendizado, isto é, cada um adequado

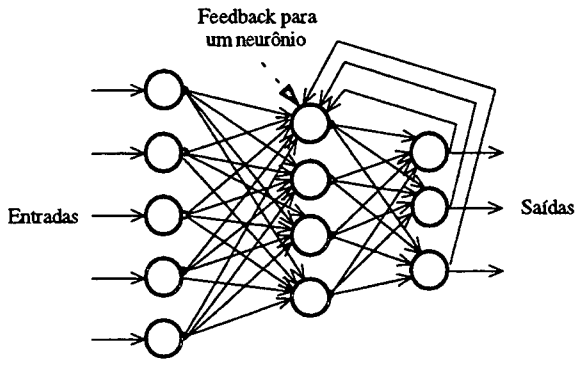

Figura 2.7: Rede com ciclos

a diferentes tipos de redes.

De uma maneira geral, o aprendizado pode ser classificado em dois tipos: associativo e não-associativo. O aprendizado associativo implica em aprender sobre o relacionamento que há entre pares de estímulos. Este tipo de aprendizado é um modelo para RNA supervisionadas. Quanto ao aprendizado não-associativo, não há estímulos secundários para associar com os estímulos primários. Neste tipo de aprendizado, a repetição de um estímulo fornece a oportunidade para aprender sobre suas propriedades. Assim, este tipo de aprendizado é um modelo para RNA não-supervisionadas [113].

Aprendizado Supervisionado

Durante a sessão de treinamento de uma RNA, pares de entradas e saídas são apresentados à ela. A rede toma cada entrada e produz uma resposta na saída. Esta resposta é comparada com o sinal de saída desejado. Se a resposta real difere da resposta desejada, a RNA gera um sinal de erro, o qual é, então, usado para calcular o ajuste que deve ser feito para os pesos sinápticos da rede. Assim, a saída real casa com a saída desejada. Em outras palavras, o erro é minimizado. O processo de minimização de erro requer um circuito especial conhecido como um professor ou supervisor, daí o nome Aprendizado Supervisionado.

Para este tipo de aprendizado, no uso de RNA, leva-se em consideração a importância de cálculos requeridos ao minimizar o erro, pois depende do algoritmo usado. Claramente, o algoritmo é puramente uma ferramenta matemática derivada

de técnicas de otimização. Tais técnicas são muitíssimas usadas para os paradigmas conexionistas, porém, alguns parâmetros devem ser levados em conta, tais como: o tempo requerido por iteração, o número de iterações por padrão de entrada para o erro alcançar um valor mínimo durante a sessão de treinamento, se a RNA atingiu um mínimo local ou global, caso tenha alcançado um mínimo local se a rede pode escapar dele, etc. Um dos algoritmos, que se utiliza dos parâmetros acima mencionados, é o de retropropagação, o qual será detalhado na Seção 2.2.5.

Aprendizado Não-Supervisionado

Em contraste ao aprendizado supervisionado, o aprendizado não-supervisionado[7] não requer um professor, isto é, não há saída desejada. Durante a sessão de treinamento, a RNA recebe em sua entrada excitações muito diferentes ou padrões de entrada e organiza, arbitrariamente, os padrões em categorias. Quando uma entrada é aplicada à rede, a RNA fornece uma resposta de saída indicando a classe a qual a entrada pertence. Se uma classe não pode ser encontrada para o padrão de entrada, uma nova classe é gerada. Este tipo de aprendizado é utilizado em sistemas classificadores.

2.2.5 Regras de Aprendizado

Para que se possa solucionar um problema, usando-se o modelo das RNA, o primeiro passo é estabelecer um conjunto de pesos para suas conexões, ativar um conjunto de unidades que correspondam a um padrão de entrada e observar o padrão para o qual a rede converge e em que se estabiliza. Se o padrão final não corresponder ao que se deseja associar, como resposta ao de entrada, é preciso fazer ajustes nos pesos e ativar novamente o padrão de entrada. Por causa de sua semelhança com o aprendizado humano, esse processo de ajustes sucessivos das RNA é chamado de aprendizagem.

Os paradigmas desenvolvidos, neste sentido, utilizam regras de aprendizado que são descritas por expressões matemáticas, chamadas de equações de aprendizado [105]. Estas equações descrevem o processo de aprendizado para o paradigma, o qual, na realidade, é o processo para auto-ajustar seus pesos sinápticos.

Uma das regras de aprendizado para RNA, bem conhecida, é a regra de Hebb [87][88]. Outra, é a regra delta [189], que é inspirada na regra de Hebb. Esta regra

[7]Para maiores detalhes sobre outros tipos de aprendizado, veja [113][105].

foi desenvolvida por Bernard Widrow e Ted Hoff, por isso, é conhecida também como regra de aprendizado de *Widrow-Hofff* ou como *Least Mean Square* (LMS), tendo em vista, que ela minimiza o erro médio quadrático [113]. Além destas, há a regra delta generalizada, que será descrita a seguir.

Regra Delta Generalizada

Muitas redes usam alguma variação da fórmula para treinamento da regra delta. Uma delas, é a regra delta generalizada ou, também, conhecida por algoritmo de retropropagação.

O algoritmo de retropropagação foi desenvolvido por P. Werbos, em 1974 [188], e redescoberto, independentemente, por D. Parker (1982) [147] e D.Rumelhart et al. (1986) [159][105]. Desde sua redescoberta, o algoritmo de retropropagação tem sido amplamente usado como um algoritmo de aprendizado para RNA, com topologia direta com múltiplas camadas.

Uma RNA que utiliza um algoritmo de retropropagação aprende de forma supervisionada (através de exemplos), em tempo discreto e utiliza um método de gradiente descendente para correção de erro, ou seja, o algoritmo de codificação executa um mapeamento entrada-saída, através da minimização de uma função de custo. A função de custo é minimizada, realizando-se iterativamente ajustes nos pesos sinápticos, de acordo com o erro quadrático acumulado[8] para todos os padrões do conjunto de treinamento[9]. Outras funções de custo podem ser utilizadas, mas independentemente disto, o procedimento de ajuste de pesos é realizado através do cálculo da mudança do valor da função de custo, com respeito à mudança em cada peso (método do delta). O processo de evolução da redução gradativa de erro, que acompanha a minimização, pode levar a convergência. A medida que a rede aprende o valor do erro converge para um valor estável, normalmente irredutível. O processo de aprendizagem prossegue, até que algum critério seja estabelecido, como por exemplo, um valor mínimo de erro global, ou uma diferença sucessiva mínima entre erros calculados para cada iteração.

Além disso, as funções de ativação não-linear, usadas pelo algoritmo de retropropagação (*backpropagation*), devem respeitar certos critérios, como serem não-lineares

[8]Nem sempre o erro quadrático é acumulado. Este é acumulado quando o treinamento é por épocas.

[9]Eventualmente, o treinamento pode ser feito por *padrão* e não por *época*.

e deriváveis. A presença de não-linearidade é importante porque, caso contrário, a relação entrada/saída da rede seria reduzida a um *perceptron* de uma única camada [86][159].

Aprendizado

A Figura 2.8 representa um exemplo típico de uma rede multicamada. Nessa figura, X_i, H_i e Y_i, representam níveis de ativação dos neurônios da camada de entrada, da intermediária e da de saída. Os pesos sobre as conexões, entre as camadas de entrada e intermediária, são denotadas por $W1_{ij}$, enquanto que, os pesos sobre as conexões entre as camadas intermediária e de saída são denotadas por $W2_{ij}$. Essa rede tem três camadas, embora seja possível e, às vezes útil, ter mais. Cada neurônio de uma camada é conectado para a frente a cada neurônio da camada seguinte. As ativações fluem da camada de entrada para a camada intermediária e, daí, para a camada de saída. Como as ativações fluem em apenas uma direção, não há necessidade do processo de relaxamento iterativo.

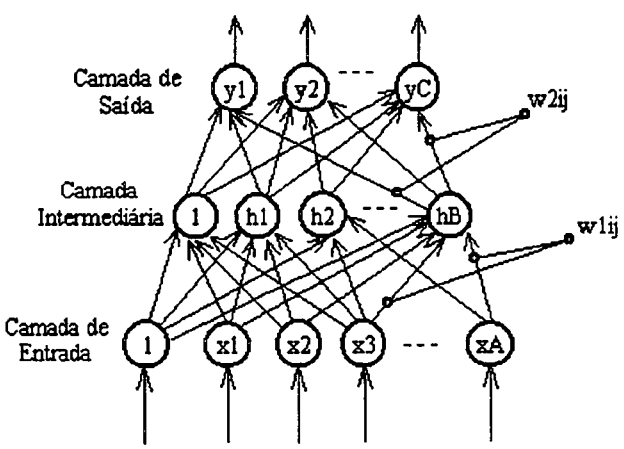

Figura 2.8: Uma rede multicamada

Quanto aos padrões, ou melhor, ao conjunto de exemplos a ser aplicado na entrada da RNA, este deve ser escolhido de forma a representar todos os padrões[10] existentes no possível conjunto de padrões que se quer classificar. De preferência, coloca-se vários exemplares de cada classificação de saída (por exemplo, vários conjuntos de sintomas associados a um determinado diagnóstico). Os exemplos podem

[10]Isto nem sempre é possível, apesar de ser o desejável.

ser sintéticos ou reais, sendo preferível utilizar-se os segundos, caso seja possível. Por exemplo, um conjunto de exemplos pode ser derivado, aleatoriamente, de um conjunto maior de pares de padrões entrada-saída, e usado para treinar a rede até a convergência. Em seguida, os pares de padrões restantes podem ser utilizados para testar se a rede é capaz de classificar corretamente casos conhecidos.

Durante a sessão de treinamento da rede, um par de padrões é apresentado (X_k, T_k), onde X_k é o padrão de entrada e T_k é o padrão de saída desejado. O padrão X_k causa a resposta na saída de cada neurônio em cada camada e, então, uma saída real Y_k na camada de saída. Na camada de saída, a diferença entre as saídas reais e desejadas fornecem um valor de erro. Este erro depende dos valores dos pesos dos neurônios em cada camada. Entretanto, este erro é minimizado, e durante este processo, valores novos para os pesos são obtidos. A rapidez e a precisão do processo de aprendizado, isto é, o processo de atualização de pesos, também depende de um fator conhecido como taxa de aprendizado.

Antes de se iniciar o processo de aprendizado por retropropagação, é necessário que se tenha:

- O conjunto de padrões de treinamento, entrada e saída desejada;
- Um valor para a taxa de aprendizado;
- Um critério que finalize o algoritmo;
- Uma metodologia para atualizar os pesos;
- A função de ativação não-linear derivável (usualmente, usa-se a sigmóide), bem como a operação matemática do ajuste dos pesos, no passo para trás, também derivável;
- Valores de pesos iniciais.

O processo, então, inicia-se por aplicar, além de um conjunto de padrões de entrada-saída (X_k, T_k), um conjunto de pesos aleatórios. A rede ajusta seus pesos toda vez que executa um ciclo, ou seja, uma época. Assim, a cada época é apresentado um conjunto de padrões em que cada um deles requer dois estágios: um passo para a frente e um para trás. O passo para frente envolve apresentar um exemplo da entrada para a rede e deixar as ativações fluírem, até chegarem à camada de saída. Durante o passo para trás, o produto da rede, isto é, obtido no passo para frente (saída da rede), é comparado com a saída desejada, e valores de erro são computados para os neurônios da camada de saída. Os pesos conectados aos neurônios da

camada de saída podem ser ajustados para reduzir esses erros. Depois, pode-se usar os valores de erro dos neurônios da camada de saída, a fim de derivar as estimativas de erro, para os neurônios das camadas intermediárias. Finalmente, os erros são propagados para trás (por isso o nome de retropropagação) até a conexão, cuja raiz, está nos neurônios da camada de entrada.

Além disso, o algoritmo de retropropagação atualiza seus pesos incrementalmente, depois de analisar cada par entrada-saída. Assim, após o algoritmo ter visto todos os pares entrada-saída, e ajustado seus pesos, diz-se que uma época foi concluída. O treinamento de uma rede que utilize este tipo de algoritmo de aprendizado, em geral, requer muitas épocas. Porém, além de se fazer a correção dos pesos por época, uma outra maneira de se atualizar os pesos, é por padrões, isto é, ela se dá a cada padrão dentro de uma época.

Desenvolvimento Matemático

Considerando uma rede direta de três camadas, com uma função de ativação tipo sigmóide unipolar, tem-se:

$$f(x) = \frac{1}{1 + e^{(-gx)}} \tag{2.1}$$

Cuja derivada é:

$$\frac{df(x)}{dx} = g\, f(x)(1 - f(x))$$

Onde g é o ganho somático, isto é, responsável pela inclinação da curva de uma dada função de ativação.

O erro total J, para a rede e para todos os padrões, é definido como a soma dos quadrados das diferenças entre a saída real da rede e seu valor desejado para a camada de saída, ou seja:

$$J = \frac{1}{2} \sum_{k=1}^{n} (T_k - Y_k)^2$$

Para maiores detalhes, sobre o desenvolvimento matemático, da avaliação de um conjunto de pesos, em todas as camadas da rede que minimize J, veja a referência [105].

Assim, a computação dos erros dos neurônios da camada de saída, é dado por:

$$\delta 2_k = (1 - Y_k)(T_k - Y_k)$$

Onde :

$\delta 2_k$ = erro da camada de saída
T_k = saída desejada
Y_k = saída real da rede

Enquanto a computação dos erros dos neurônios da camada intermediária, é dado por:

$$\delta 1_k = H_k(1 - H_k)\sum_{r=1}^{n} \delta 2_r W 2_{rk}$$

Onde:

$\delta 1_k$ = erro da camada intermediária
H_k = saída da camada intermediária
$W2_k$ = peso entre a camada de saída e a camada intermediária

Conseqüentemente, o ajuste dos pesos da camada de saída, é:

$$\Delta W 2_{lk} = \eta \delta 2_l H_k$$

Onde o coeficiente de aprendizagem é denotado por η.

Enquanto que, o ajuste dos pesos de uma dada camada intermediária, é:

$$\Delta W 1_{lk} = \eta \Big[\frac{dY_l}{dx}\Big(\sum_{k=1}^{n}(T_k - Y_k)W 2_{lk}\frac{dY_k}{dx}\Big)\Big]Y_{lk}$$

O processo de computar o gradiente e ajustar os pesos é repetido até atingir um erro mínimo.

Algoritmo de Treinamento de Retropropagação

Baseado na referência [157], descreve-se a seguir, sucintamente, os passos de um algoritmo de retropropagação típico:

1. Seja A, o número de neurônios da camada de entrada, conforme determinado pelo comprimento dos vetores de entrada de treinamento, C, o número de neurônios da camada de saída. Escolha B, o número de neurônios da camada intermediária. Conforme a Figura 2.8, as camadas de entrada e intermediária têm, cada uma, um neurônio extra, usado como *bias* (limiar), portanto, usa-se os intervalos $(0, \cdots, A)$ e $(0, \cdots, B)$ para estas camadas, especificamente.

2. Inicialize os pesos da rede. Cada peso deve ser ajustado aleatoriamente.

3. Inicialize as ativações dos neurônios com *bias*, ou seja, $x_0 = 1$ e $h_0 = 1$.

4. Escolha um par entrada-saída. Suponha que o vetor de entrada seja X_l, e que o vetor de saída real seja Y_l. Atribua níveis de ativação, aos neurônios da camada de entrada.

5. Propague a ativação dos neurônios da camada de entrada para os da camada intermediária, usando-se como sugestão, a função sigmóide unipolar[11].

$$H_k = \frac{1}{1 + e^{-a}}, \forall\, k = 1, \cdots, B$$

Onde:

$$a = \sum_{l=0}^{A} (W1_{lk} X_l)$$

6. Propague a ativação dos neurônios da camada intermediária para os da camada de saída.

$$Y_j = \frac{1}{1 + e^{-a}}, \forall\, k = 1, \cdots, C$$

[11] Da Equação 2.1, nós supomos aqui que:

$$a = gx$$

Onde:

$$g = 1$$

$$x = \sum_{l=0}^{A} (W1_{lk} X_l)$$

As suposições matemáticas referenciadas acima também são válidas para os itens seguintes.

Onde:

$$a = \sum_{l=0}^{B}(W2_{lk}H_l)$$

7. Compute os erros dos neurônios da camada de saída, denotada por $\delta 2_k$.

$$\delta 2_k = Y_k(1 - Y_k)(T_k - Y_k),\ \forall\, k = 1, \cdots, C$$

8. Compute os erros dos neurônios da camada intermediária, denotada por $\delta 1_k$.

$$\delta 1_k = H_k(1 - H_k)\sum_{l=1}^{C}\delta 2_l W2_{lk},\ \forall\, k = 1, \cdots, B$$

9. Ajuste os pesos, entre a camada intermediária e a da saída.

$$\Delta W2_{lk} = \eta\delta 2_{kj}H_l,\ \forall\, l = 0, \cdots, B;\ \forall\, k = 0, \cdots, C$$

10. Ajuste os pesos entre a camada de entrada e a intermediária.

$$\Delta W1_{lk} = \eta\delta 1_k X_l,\ \forall\, l = 0, \cdots, A;\ \forall\, k = 1, \cdots, B$$

11. Vá para a etapa 4 e repita. Quando todos os pares entrada-saída, tiverem sido apresentados à rede, uma época terá sido completada. Repita as etapas de 4 a 10, para tantas épocas quantas forem desejadas.

D. Rumelhart et al. [159], descreveram um método para melhorar o tempo de treinamento do algoritmo de retropropagação, ao mesmo tempo, que realça a estabilidade do processo. Chamado de *momento*, o método adiciona um termo para o ajuste de peso, que é proporcional ao valor da mudança de peso anterior. Uma vez que o ajuste é feito, ele é guardado e serve para modificar o ajuste de pesos subseqüente. Desta forma, as expressões referentes aos ajustes de pesos, com a incidência do termo *momento*, denotam uma combinação convexa entre a contribuição do gradiente com a variação anterior nos pesos das conexões. Isto pode ser explicado ao se aplicar a pesquisa de gradiente descendente para reduzir o erro total, na saída da rede neural, através do ajuste de pesos. Conforme J.C. Zurada [196], isto é usualmente feito de acordo com a fórmula:

$$\Delta W(t) = -\eta\nabla E(t) + \beta\Delta W(t-1)$$

Onde:

t e t - *1* - indicam a etapa de treinamento atual e a anterior
β - termo de momento
$\nabla E(t)$ - gradiente descendente do erro

Desta forma, antes de se definir as fórmulas finais para as modificações dos pesos entre as camadas da RNA, primeiramente abordar-se-á o comportamento do gradiente descendente em um determinado ponto, por exemplo, *A'* em relação a uma dada trajetória (superfície $\nabla E(t)$) para um caso de duas dimensões. Levar-se-á em conta para este estudo o uso dos coeficientes de aprendizagem e de momento. Suponha que: as derivadas consecutivas $\partial E/\partial W_1$ e $\partial E/\partial W_2$, nos pontos *A'* e *A"*, são do mesmo sinal. Obviamente, combinando-se as componentes do gradiente de várias etapas adjacentes resultaria na convergência mais rápida. Se a superfície do erro apresenta uma variação uniforme em torno de um certo eixo, a taxa de aprendizado, em torno desta componente particular, aumentaria. Por adicionar o termo momento, o ajuste de peso em *A"* é valorizado pela fração do ajuste de pesos em *A'*. Então, após iniciar o procedimento do gradiente descendente, por exemplo, em *B'*, as duas derivadas $\partial E/\partial W_1$ e $\partial E/\partial W_2$, inicialmente negativas em *B'*, ambas alteram seus sinais em *B"*. Neste ponto, o gradiente descendente não possui uma direção eficiente de ajuste de peso. Isto se deve ao fato que o deslocamento desejado de *B"* estaria mais direcionado a um ponto mínimo *M*, ou preferivelmente movendo o vetor peso ao longo de um *vale*. Assim, o deslocamento $\eta\nabla E(t+1)$ em *B"* moveria preferivelmente os pesos por meio de uma grande magnitude do que através do vale e próximo do ponto inicial de *B'*. Movendo-se os pesos por $-\eta\nabla E(t+1)+\beta\Delta W(t)$, entretanto, reduz-se a magnitude do ajuste de peso de tal forma que o vetor resultante $\Delta W(t+1)$ surge mais ao longo do vale. Daí, pode-se inferir que se as componentes do gradiente mudam os sinais em duas iterações consecutivas, a taxa de aprendizado em torno deste eixo diminuiria. Esta discussão indica que o termo momento auxilia a rapidez da convergência e alcança um perfil de aprendizado eficiente e mais confiável.

Portanto, as etapas 9 e 10 de modificação de pesos são alteradas de forma que elas incluam o termo de momento. Então, as fórmulas ficam:

$$\Delta W2_{lk}(t+1) = \eta\delta 2_k H_i + \beta\Delta W2_{lk}(t)$$

$$\Delta W1_{lk}(t+1) = \eta\delta 1_k X_i + \beta\Delta W1_{lk}(t)$$

2.3 Lógica fuzzy

A dificuldade ou a impossibilidade de obter-se todas as informações e de equacionar a realidade imprecisa do mundo, levou alguns cientistas, a propor lógicas alternativas, segundo seus modos de pensar, mais propícias à representação daquele mundo particular. A lógica *fuzzy* de Zadeh[12] é um exemplo [190][191]. Outros, acrescentaram fatores para exprimir o quanto a informação é confiável. O SE Mycin da autoria de E.H. Shortliffe [163], pode neste caso ser citado como exemplo. Este sistema emprega regras de produção com Fatores de Certeza (FC), no auxílio do raciocínio probabilístico nas doenças infecciosas sangüíneas.

Portanto, boa parte dos problemas relacionados com a representação de conhecimento, resultam das dificuldades que se tem em expressar com a precisão desejada, as idéias sobre pensamentos, sensações ou percepções do mundo físico que nos rodeia. Então, uma teoria que permite dar forma matemática às expressões próprias, por exemplo, da linguagem natural, sem diminuir a potência expressiva das mesmas, é a Matemática *Fuzzy*. Através dela, pode-se realizar *operações com palavras*, onde os conjuntos *fuzzy* são os *valores* das palavras [17].

Exemplificando o que foi dito anteriormente, a motivação para os conjuntos *fuzzy* vem da necessidade de representar proposições do tipo:

Pedro é muito alto.
Joana está ligeiramente mais gorda.
Lígia é baixa.
João está com febre alta.

Assim, a imprecisão, a respeito de uma afirmação, é expressa através de um número que em vez de probabilidade exprime possibilidade da afirmação ser correta. No caso do exemplo, *João está com febre alta*, pode-se exprimir a possibilidade da febre pertencer ao conjunto de coisas altas.

O tratamento da imprecisão, pode ser requerido em diversas etapas do manejo do conhecimento [157]:

[12] O termo *fuzzy* foi introduzido por L.A. Zadeh, em 1965, bem como a formalização dos conjuntos *fuzzy*. Isto se deve ao fato que, até aquele momento, a imprecisão estava sendo tratada em termos de incerteza, através de modelos probabilísticos. Este tratamento não satisfatório levou a publicação do artigo *Fuzzy Sets* em 1965 [190].

a) Coleta da informação;
b) Definição dos elementos do conhecimento;
c) Combinação de elementos entre si, ou seja, imprecisão nas premissas;
d) Forma de obter conclusões, isto é, aplicação de uma regra de raciocínio;
e) Avaliação de uma seqüência de regras ou estruturas, como é o caso da aplicação sucessiva de regras de raciocínio.

Na aquisição de informação, o engenheiro de conhecimento e o próprio especialista, deparam-se com informações não facilmente classificáveis como por exemplo, a caracterização de um sinal como fraco, médio ou forte. Pessoas diferentes certamente discordarão na classificação de alguns sinais.

Na definição dos elementos de conhecimento, quando a preocupação se concentra em fazer sínteses, é freqüentemente necessário, recorrer a métodos estatísticos e probabilísticos para depurar índices e freqüências.

Na combinação de elementos de conhecimento entre si, permitindo certa conclusão, é onde a presença da imprecisão mais se acentua. Entra aí, a consideração de os eventos serem ou não independentes entre si e o quanto cada informação pesa sobre a conclusão.

A avaliação de uma seqüência de eventos é, certamente, o caso mais difícil de ser equacionado, uma vez que envolve o estabelecimento dos raciocínios que o especialista deve efetuar. Para cobrir todas as decisões possíveis do especialista, é necessário que todas as árvores de decisão sejam construídas e avaliadas. Isto é possível de ser feito, apenas para problemas pequenos. Em problemas maiores, deve-se recorrer, forçosamente, a métodos de raciocínio aproximado (impreciso), que constituem a forma, até hoje, conhecida de abordar tais problemas.

Assim, a teoria tradicional dos conjuntos define a pertinência as proposições *Pedro é muito alto, Joana está ligeiramente mais gorda, Lígia é baixa e João está com febre alta*, como um predicado booleano (sim ou não). Por outro lado, a teoria *fuzzy* permite representar a pertinência a um conjunto como uma distribuição de possibilidades[13], como é mostrada na Figura 2.9, lado esquerdo, onde aparece o

[13]Zadeh define distribuição de possibilidades como sendo: Seja em Y uma variável com valores em X; então, a distribuição de possibilidade, Π_Y, associado com Y, pode ser visto como uma restrição *fuzzy*, nos valores que podem ser assumidos por Y. Assim, a distribuição é caracterizada

conjunto de pessoas altas e o conjunto de pessoas muito altas. Já, na mesma Figura 2.9, lado direito, mostra-se a definição booleana, padrão de pessoas altas. Nesta última, ou uma pessoa é alta ou não, e deve haver uma altura específica que defina o limite. O mesmo se aplica para o caso das pessoas muito altas. Na primeira figura, o tamanho de alguém aumenta com sua altura até o valor 1 ser alcançado.

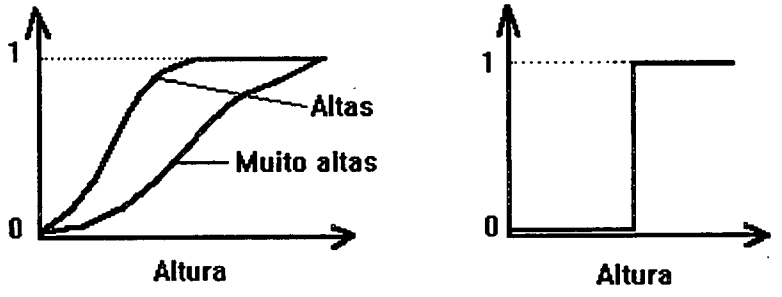

Figura 2.9: Pertinência *fuzzy* versus pertinência abrupta

Então, este tipo de imprecisão leva em consideração a teoria de conjuntos, porém, restringida para o intervalo [0,1]. Ela implica na representação de imprecisão por um grau expresso pelo valor de uma função de pertinência (grau de pertinência, grau de compatibilidade ou grau de verdade). Esta função de pertinência pode expressar o grau de um elemento pertencer a um conjunto.

Definindo:

Se X, é uma coleção de objetos, denotados genericamente por x, então, um conjunto *fuzzy* A em X, é um conjunto de pares ordenados, ou seja:

$$A = (x, \mu_A) | x \epsilon X$$

Onde:

A : conjunto *fuzzy*

x : elemento pertencente ao universo de discurso X

por uma função de distribuição de possibilidade $\Pi_Y : X \rightarrow [0,1]$, o qual, associa com cada $x \epsilon X$, o grau de possibilidade que Y pode assumir de x como um valor [103].

X : universo de discurso
μ_A: grau de pertinência

Contudo, como são manipulados estes fatores de imprecisão (*fuzzy*) em uma regra *fuzzy*? Na lógica booleana, a função de operadores booleanos E, OU e NÃO, é bem conhecida, já que é bastante empregada em problemas onde se necessite de respostas, como: falso ou verdadeiro, 0 ou 1, sim ou não. Na lógica *fuzzy*, os valores não são *crisp* e suas *fuzziness* exibem uma distribuição usando-se a função de pertinência [105].

Há várias maneiras destas funções de pertinência serem combinadas e trabalhadas. Uma delas, é pelos operadores Max/Min. Em termos simples, se considerarmos a operação união[14] (equivalente a OU), o resultado é igual à variável de entrada, com o maior valor de todos elas, ou seja, $max(x_1, x_2, x_3, \cdots, x_n)$. Isto é, se $\mu_A = 0,5$, $\mu_B = 0,7$ e $\mu_{A \cup B} = \mu_A \; OU \; \mu_B$, então $\mu_{A \cup B} = max(0,5; 0,7) = 0,7$. Agora, se considerarmos a operação intersecção[15] (equivalente a E), a resposta é igual ao valor mínimo das variáveis de entrada, ou seja, $min(x_1, x_2, x_3, \cdots, x_n)$. Neste caso, se $\mu_{A \cap B} = \mu_A \; E \; \mu_B$, então $\mu_{A \cap B} = min(0,5; 0,7) = 0.5$. Se considerarmos o operador complemento (equivalente a NÃ0), então a resposta é o complemento de 1, ou $\overline{x} = 1 - x$. Se $\mu_C = \overline{\mu}_B$, então $\mu_C = 1 - 0,7 = 0,3$. Outra maneira, de solucionar este problema, é utilizando-se outros operadores de agregação, como é o caso do produto algébrico e soma algébrica [195].

Porém, há ainda o problema de como são geradas estas regras *fuzzy*. Na maioria dos problemas, as regras *fuzzy* são geradas baseadas na experiência passada, como é o caso do diagnóstico médico ou controle *fuzzy*, onde deve-se conhecer, a maioria das possíveis relações entrada-saída em termos *fuzzy*. Assim, as relações entrada-saída ou regras são, então, expressas facilmente com os estados *Se/Então*, tais como:

$$Se\ A\ E/OU\ B\ Entao\ C$$

Onde:

E/OU : união ou intersecção lógica
A e B : entradas acrescidas de um grau de pertinência

[14]A classe correspondente dos operadores de união para conjunto *fuzzy* é referenciada também como normas triangulares ou t-normas.

[15]A classe correspondente dos operadores de intersecção para conjunto *fuzzy* é referenciada também como co-normas triangulares ou t-co-normas (s-normas).

C : ação (conclusão) da regra.

Outro questionamento, que pode ser efetuado, refere-se em que parte(s) da regra *fuzzy*, deve-se colocar os graus de pertinência. Essa condição depende da necessidade e aplicabilidade de cada problema. Esta etapa envolve a perspicácia do engenheiro de conhecimento de lidar com as variáveis envolvidas em tal problema, juntamente com o auxílio do especialista de domínio. Portanto, esta condição é essencial para que todo o desenvolvimento do projeto atinja o objetivo esperado, com um certo grau de confiabilidade, principalmente, no caso de desenvolvimento de um SE.

Os graus de pertinência, para um determinado conjunto de regras, podem ser colocados somente na parte *Se* da regra, bem como, tanto no antecedente como no conseqüente (*Então*) de uma regra. Exemplificando, supondo que um certo especialista médico tenha articulado a seguinte regra:

R_1 : *Se* febre alta (0,8) *E* estável (0,7) *E* erupção na pele com nevus (0,6) *E* seguidas de bolhas (0,8) *E* acnes (0,9) *Então* A.

Os valores que se apresentam a cada variável acima, são as importâncias relativas de todos os sintomas de uma dada doença. Assim, cada sintoma é acompanhado por seus respectivos graus de pertinência (confiança), os quais são obtidos do médico em relação ao diagnóstico de uma doença. Isto quer dizer que por exemplo, ao se supor que o paciente apresenta febre alta, implica em dizer, que ele está com febre alta, com confiança de 0,8. Assim, aplica-se um dos operados da lógica *fuzzy*, o operador de Mínimos, para a possibilidade das variáveis envolvidas na respectiva regra. Desta forma, obtém-se o grau de pertinência resultante, ou seja:

$$R_1 = min(0,8; 0,7; 0,6; 0,8; 0,9) = 0,6$$

Contudo, para o caso em que aparecem os graus de pertinência, tanto na parte *Se* quanto na parte *Então* da regra, tem-se que:

R_2 : *Se* febre moderada (0,5) *E* erupção na pele com nevus (0,6) *E* seguidas de bolhas (0,8) *E* acnes (0,9) *Então* B com 0,6.

Então, o grau de pertinência resultante para a regra R_2 é:

$$R_2 = min((min(0,5; 0,6; 0,8; 0,9)); (0,6)) = 0,5$$

Uma vez redefinida, desta maneira, a pertinência de um conjunto, é possível definir um sistema de raciocínio baseado em técnicas de combinação de distribuições. Assim, a lógica *fuzzy* é mais uma das maneiras de se representar o conhecimento de um especialista de domínio, porém, acrescida de um dos modelos de imprecisão utilizados pela IA.

2.4 Computação Evolucionária

Computação evolucionária é o nome genérico, dado a métodos computacionais, inspirados na teoria da evolução. Os algoritmos usados, em computação evolucionária, são conhecidos como Algoritmos Evolucionários (AE) [10].

Atualmente, os AE mais conhecidos são: Algoritmos Genéticos (AG), Programação Evolucionária e Estratégias Evolucionárias. Todos compartilham de uma base conceitual comum, que consiste na simulação da evolução de estruturas individuais, via processos de seleção e os operadores de busca, referidos como Operadores Genéticos (OG), tais como, mutação[16] e crossover (cruzamento ou recombinação). O processo depende do *fitness* (aptidão), atingido pelas estruturas individuais, frente a um ambiente. A seleção é focalizada nos indivíduos com um alto grau de aptidão, explorando então, a informação da aptidão disponível. O *crossover* e a mutação pertubam estes indivíduos, fornecendo heurística geral para a exploração.

Contudo, para que se tenha uma melhor compreensão dos AE, é necessário que se façam algumas avaliações dos processos biológicos, nos quais eles são baseados. A próxima seção, tratará deste assunto.

2.4.1 Base Biológica

Charles Darwin e Alfred Russel Wallace foram os precursores, através de suas evidências para a teoria de evolução, em 1858, na revolução tanto do pensamento biológico quanto da filosofia humana. Desde então, esta teoria é uma das mais aceitas pelo mundo científico, originando o chamado paradigma Neo-Darwiniano, proveniente da teoria evolucionária Darwiniana clássica, combinada com o selecionismo de Weismann e a genética de Mendel [56].

[16]Nós cremos que seja melhor falar de *variabilidade*. A mutação sendo um operador para obter esta variabilidade.

O Neo-Darwinismo afirma que a história de existência de vida, em nosso mundo, é atribuída completamente por somente uns poucos processos estatísticos, que agem sobre populações e espécies. Estes processos são: reprodução, mutação, competição e seleção. Reprodução é uma propriedade óbvia de toda vida. Mas, similarmente, mutação é garantida para algum sistema, no qual, ela própria se reproduz continuamente, em um universo positivamente entrópico. Competição e seleção, tornam-se as conseqüências inevitáveis de alguma população expandida restringida para uma área finita. Evolução é, então, o resultado desses processos estatísticos, interagindo, fundamentalmente, nas populações, geração após geração [56].

A evolução, na natureza, ou em qualquer lugar, não é um processo proposital ou dirigido, isto é, não há evidências de que a meta da evolução seja a produção do homem. No entanto, os processos da natureza parecem se constituir em diferentes indivíduos, competindo por recursos no ambiente. Alguns são melhores do que os outros. Aqueles que são melhores estão mais habilitados a sobreviver e propagar sua carga genética.

Na natureza, a perpetuação da informação genética é mantida pelo genoma, de modo que ela é realizada através da reprodução assexuada e sexuada. A codificação é um sistema igual em todos os seres vivos, porém com variações, e ocorre no DNA, o que confere a diversidade.

Na reprodução sexuada, a produção de descendentes se dá pela união de dois seres diferentes (cujo material genético é organizado em pares de unidades mais simples, os cromossomos), da mesma espécie, denominados macho e fêmea, os quais, diferem pela carga genética que contêm os cromossomos sexuais, de forma a produzirem células chamadas de gametas. O macho produz gametas masculinos e a fêmea produz gametas femininos. Um novo ser é produzido, pela união de um gameta masculino com um gameta feminino, que se divide, sucessivamente, até formar-se todos os órgãos e sistemas do indivíduo. Assim, um grupo dessas células, chamadas germinativas, é capaz de produzir um novo macho ou uma fêmea. É o processo de reprodução dos seres mais elevados na escala filogenética. Até o momento, não há algoritmos evolucionários baseados neste processo de reprodução.

Por outro lado, na reprodução assexuada não há diferença genética entre seres da mesma espécie, no que se refere ao seu papel na reprodução. Na descendência o material genético parental é trocado entre seus cromossomos (*crossover*), resultando

em cromossomos filhos, cujo material genético é a combinação dos materiais dos dois progenitores. Este processo é o imitado pela maioria dos AE.

Convém ainda notar, quanto a evolução biológica, que esta exige diversidade. Na natureza, a diversidade deriva de mutações.

Em relação aos AE, normalmente se inicia o algoritmo já com uma grande diversidade, utilizando-se um método aleatório para produzir genótipos diferentes na população inicial. Conseqüentemente, a importância do mecanismo de mutação, neste caso, é de menor importância e mesmo discutível. O mesmo ocorre em mecanismos de reprodução sexuada. Acredita-se que neste caso, as mutações podem servir a manter uma diversidade, que poderia ter desaparecido, ao longo de várias gerações, e apenas indivíduos, com alto nível de aptidão, teriam sobrevivido[17] [10].

2.4.2 Algoritmos Genéticos

AG constituem uma técnica de busca, inspirada no processo de evolução dos seres vivos, baseada na seleção natural de Darwin.

Considerando os sistemas biológicos como um todo, observa-se que os mesmos desenvolveram, ao longo da sua evolução, estratégias de adaptação de comportamento, que possibilitaram a sua sobrevivência e a perpetuação de suas espécies. As pressões do ambiente fizeram com que estas estratégias tivessem um forte impacto sobre os organismos biológicos, gerando profundas mudanças nos mesmos. Manifestações destas mudanças podem ser observadas nas especializações estruturais e funcionais, na organização da informação e nas representações internas do conhecimento [8].

Baseado nesta analogia, com o processo de evolução biológica das espécies, chamada de metáfora biológica, os AG mantêm a informação sobre o ambiente, acumulando-a durante o período de adaptação. Eles utilizam, tal informação acumulada, para podar o espaço de busca e gerar novas soluções plausíveis dentro do domínio.

Dentro de uma perspectiva histórica, os AG foram desenvolvidos, inicialmente, por John H. Holland, em 1975, no trabalho intitulado *Adaption in Natural and Artificial Systems* [92]. Holland, inspirou-se no mecanismo de evolução das espécies e na genética natural. De acordo com a teoria Darwiniana de evolução das espécies,

[17] Esta opinião não é compartilhada por alguns pesquisadores.

uma população sujeita a um ambiente qualquer sofrerá influências desse, de tal forma que, os mais aptos terão maior probabilidade de sobreviverem a tal ambiente. Desta forma, a cada geração haverá uma população mais apta ao ambiente em questão [52].

Busca e Otimização

Os AG podem ser enquadrados em grande parte dos problemas científicos, a serem formulados como problemas de busca e otimização.

Basicamente, existe uma série de fatores influenciando o desempenho de um dado sistema. Tais fatores podem assumir um número limitado ou ilimitado de valores e podem ser sujeitos a certas restrições. O objetivo é encontrar a melhor combinação dos fatores, ou seja, a combinação de fatores, que proporcione o melhor desempenho possível para o sistema em questão. Em termos técnicos, o conjunto de todas as combinações possíveis para os fatores constitui o chamado espaço de busca. Não é difícil perceber, que existe uma dualidade, entre os conceitos de busca e otimização, de tal modo que todo problema de busca pode ser considerado um problema de otimização e vice-versa [73][177].

O problema de otimização pode ser solucionado por meio de métodos probabilísticos, numéricos ou enumerativos, ou por hibridismo destes métodos.

Os métodos numéricos podem ser divididos em analíticos, cuja função $f(x)$ é explicitamente conhecida e derivável, ou pode ser aproximada, por alguma função derivável até o grau desejado de precisão, enquanto que, nos baseados em cálculo numérico, caso o espaço de busca seja linear, técnicas de Programação Linear, como o método *simplex*, são suficientes [177]. Contudo, em ambientes não-lineares, técnicas de gradiente ou de estatística de ordem superior são geralmente empregadas. Já os métodos enumerativos de otimização examinam cada ponto do espaço de busca, um por um, em busca dos pontos ótimos. Por outro lado, os métodos probabilísticos são métodos que empregam a idéia de busca probabilística, isto é, descrevem a variação de sistemas, que se realizam, essencialmente, sob condições inalteradas. Esses sistemas são chamados de sistemas aleatórios, de forma que a teoria de probabilidade permite modelar seu comportamento.

Os AG fazem parte da classe correspondente aos métodos probabilísticos de busca e otimização, apesar de não serem aleatórios. Os AG usam o conceito de probabili-

dade, mas não são simples buscas aleatórias. Pelo contrário, os AG tentam direcionar a busca para regiões onde é provável que os pontos ótimos estejam.

Além disso, em relação as técnicas de busca convencionais, os AG diferem nos seguintes pontos:

- A busca da melhor solução para o problema é feita sobre uma população de pontos, e não sobre um único ponto, reduzindo sensivelmente o risco da solução recair sobre um máximo (ou mínimo) local;
- Os AG realizam uma busca cega. A única exigência é o conhecimento do valor da função de custo (ou objetivo) de cada indivíduo. Não há necessidade de qualquer outra informação, ou heurística dependente do problema.
- Os AG usam operadores estocásticos e não regras determinísticas para guiar uma busca altamente exploratória e estruturada, onde informações acumuladas nas iterações (gerações) anteriores são usadas para direcionar essa busca.

Apesar de sua simplicidade, os resultados obtidos com a aplicação do método, segundo Goldberg [73], permitem concluir que os AG são um método de busca robusto, eficiente e eficaz em uma grande variedade de problemas.

Definição

Uma das definições de AG, tomada como base, é referida por J.J. Grefenstette [74], como sendo um procedimento iterativo que mantém uma população de estruturas (chamadas indivíduos ou *string*), que representam possíveis soluções de um determinado problema. A cada incremento temporal (chamado geração), os indivíduos na população atual são avaliados de acordo com o valor de sua aptidão para solução de problema. Tendo como base essa avaliação, uma nova população de soluções candidatas é formada, utilizando-se Operadores Genéticos (OG) específicos, tais como, *crossover* e mutação.

Conceitos Fundamentais de AG

A nível biológico, um indivíduo é formado por um conjunto de cromossomos. No entanto, pode-se fazer uma analogia, neste contexto, entre indivíduo e cromossomo, tendo em vista que, um indivíduo pode ser formado por apenas um cromossomo, o que é comum em AG. Por isso, os dois termos são utilizados indistintamente, neste contexto.

Assim, um indivíduo é definido por um *string*, usando-se um alfabeto finito, de modo que cada *string* represente um conjunto de valores para o conjunto de parâmetros do problema. Um exemplo de alfabeto é o conjunto {0,1}, ou o conjunto de números inteiros. Deste modo, cada posição do *string* representa um gene.

O cromossomo é composto de genes, sendo que, cada gene possui um local fixo no cromossomo, local este, denominado de *locus*. Cada gene pode assumir um certo valor, pertencente a um certo conjunto de valores, os quais, são denominados de *alelo* [170]. Em termos de AG, o gene é denominado de *bit* e o *locus*, de posição do *bit* no indivíduo. Já o termo alelo, refere-se ao conjunto de valores possíveis de serem atribuídos a um determinado *bit*, ou seja, por exemplo, o alfabeto binário {0,1}.

Ao conjunto de cromossomos, genes e alelos, denomina-se de genótipo, e as características conferidas por este, denomina-se de fenótipo. Em termos de AG, o genótipo é a variável independente, x, e o fenótipo, a variável dependente ou função, $f(x)$ [52].

Normalmente, os AG trabalham com um conjunto de indivíduos (população)[18], no qual, cada elemento é candidato a ser a solução desejada. Cada indivíduo é codificado em uma cadeia de *bits*, denominada de cromossomo. A função a ser otimizada é o ambiente, no qual, a população inicial vai ser posta. Espera-se, que através dos mecanismos de evolução das espécies e a genética natural, somente os mais aptos, se reproduzam e, também, que cada nova geração esteja mais apta ao ambiente (função a ser otimizada).

Contudo, deve-se notar que, a representação genética pode diferir consideravelmente da forma natural dos parâmetros da solução. A cadeia de cromossomos, representada por números binários e de comprimento fixo, tem dominado a pesquisa de AG, no sentido que eles concedem o número máximo da cadeia de *bits*, bem como, tornam-se acessíveis à implementação simples.

O grau de aptidão de cada indivíduo é obtido pela avaliação de tal indivíduo, através da função a ser otimizada. Se o objetivo for maximizar, a aptidão é diretamente proporcional ao valor da função. Caso o objetivo seja a minimização da função, a aptidão será inversamente proporcional ao valor da função. Contudo, o

[18]No caso de se utilizar AG para otimização da topologia de rede neural, o conjunto de indivíduos será representado por uma população de redes.

termo minimização não é bem aceito por alguns pesquidadores, por não ter inspiração biológica, haja visto que, somente o mais apto é que deve sobreviver [177].

Assim, quando já se tem realizado o teste de todos os indivíduos da população, na função a ser otimizada, obtem-se a aptidão para cada um, ou seja, o seu grau de aptidão.

A próxima geração será uma evolução da anterior e, para que isso ocorra, os mais aptos, os de melhor aptidão, deverão possuir maior probabilidade de serem selecionados para dar origem à nova geração. Com isso, se o processo for bem conduzido, espera-se que a nova geração seja, em média, melhor do que a que lhe deu origem.

A seleção dos indivíduos da geração anterior, que vão participar da formação da nova geração, pode ser realizada através da roleta ponderada. Na roleta ponderada, os indivíduos que obtiveram melhor valor de aptidão, recebem maior nota. Além disso, os valores são acumulativos, como é exemplificado na Tabela 2.1.

Tabela 2.1: Roleta Ponderada

Elemento	Nota	Nota Acumulada
X_3	5	5
X_2	4	9
X_1	3	12
X_4	2	14
X_5	1	15

Como pode ser visto na Tabela 2.1, ao mais apto foi dado a nota máxima, 5, e ao menos apto, foi dada a nota mínima, 1. Em valores acumulados, tem-se 5 para o mais apto e 15 para o menos apto. O ponto crucial da roleta ponderada é a diferença entre as notas acumuladas dos elementos da população. Nota-se, que a diferença do mais apto, com nota acumulada de 5, para o segundo, com nota acumulada de 9, é de 4 unidades de espaçamento. Por outro lado, tomando o menos apto, com nota acumulada de 15, e o penúltimo, com nota acumulada de 14, nota-se, que a distância entre eles é de somente 1 unidade. Tal característica torna maior a chance dos mais aptos serem sorteados em relação aos menos aptos. Nesse caso, na razão de 4:1 [52].

Além disso, na roleta ponderada, o sorteio é realizado pela geração de um número aleatório, segundo uma distribuição uniforme ou distribuída, dependendo do tipo de aplicação. A escolha dos melhores indivíduos para participar da reprodução, faz com que a média da população caminhe na direção mais promissora da solução desejada [122][120][177].

Realizada a seleção, o próximo passo é a aplicação dos mecanismos de busca, também conhecidos como OG. Entre tais mecanismos, os mais comumente empregados em AG, são: *crossover* e mutação. Estes operadores serão descritos com maiores detalhes, na próxima seção.

Um outro ponto relevante a ser mencionado, diz respeito aos parâmetros do AG, ou seja, os valores que influenciam o desempenho do AG. Seguindo a relação proposta por S. Austin [8], estes parâmentros são: tamanho da população, taxa de operadores, intervalo de geração, seleção de estratégia e fator de escalada.

O tamanho da população de cromossomos afeta o desempenho global dos AG. Uma população pequena é insuficiente para cobrir o espaço de busca do problema. Uma população grande é mais representativa do domínio, além de evitar, a convergência prematura para soluções locais, em vez de soluções globais.

As taxas de operadores medem a freqüência com que cada tipo de OG é utilizado. Representam também, a influência que cada tipo de OG exerce sobre a população de cromossomos. Além do que, se a taxa de operadores de *crossover* for muito alta, alguns cromossomos de bom desempenho podem ser removidos mais rapidamente do que a seleção possa desenvolvê-los. Se a taxa de *crossover* for muito baixa, a busca pode estagnar. Entretanto, se a taxa dos operadores de mutação for baixa, evita-se que uma dada posição estabilize-se em um único valor. Uma taxa de mutação alta resulta essencialmente numa busca aleatória.

O intervalo de geração controla o percentual da população, a ser substituído durante cada ciclo de geração. Por exemplo: $N \times G$ cromossomos da população $P(t)$ são escolhidos para serem substituídos na população $P(t+1)$. Se o valor de G for igual a 1, significa, que toda a população é substituída durante cada geração.

As estratégias de seleção correspondem aos critérios utilizados para a escolha de cromossomos durante a reprodução. Um exemplo de estratégia de reprodução, ba-

seado em S. Austin [8], é a seleção pura, onde os cromossomos são reproduzidos em função da sua aptidão.

O fator de escalada mede a manutenção da diversidade genética da população de cromossomos durante a evolução. Um cromossomo ou um grupo de cromossomos pode ter uma aptidão bastante forte, a ponto de dominar o processo de reprodução, reduzindo-se a diversidade da população. Uma maneira de se controlar este processo é ordenando os cromossomos, escalonando o seu desempenho, para refletir sua aptidão relativa dentro da população, e utilizando as operações genéticas de mutação para reduzir a homogeneidade da população de cromossomos.

Operadores Genéticos

Indivíduos novos são criados, usando-se dois principais operadores de recombinação genética, conhecidos como *crossover* e mutação.

O *crossover* se dá pela aproximação dos cromossomos dos dois indivíduos (pais), que trocam entre si partes de seus cromossomos. Isso resulta em dois cromossomos diferentes que, porém, ainda guardam influências dos pais [10]. Há várias formas possíveis de se fazer o cruzamento [52][111][146][171]. O operador *crossover*, mais simples, é o chamado *crossover* de um ponto (*One-Point*), onde, primeiro um local de cruzamento é escolhido com probabilidade uniforme sobre o comprimento do cromossomo, então, as *strings* correspondentes são permutadas, como é mostrado na Figura 2.10. Há, ainda, muitas outras técnicas de *crossover*, como é o caso do *crossover* de dois pontos (*Two-Point*), e dos tipos uniformes [70][171]. Contudo, não há consenso sobre qual é a melhor técnica a ser usada.

A mutação consiste em perturbações na cadeia dos cromossomos dando origem a uma nova cadeia, que guardará pouca ou nenhuma informação da cadeia mãe. Na realidade, mutação é a denominação dada a vários mecanismos de alteração genética, os quais, têm em comum, o fato de fazerem o novo cromossomo apresentar pouca informação dos pais [2]. Esta alteração ocorre de forma que cada gene em cada cromossomo é um candidato à mutação, enquanto que, a seleção é determinada pela probabilidade de mutação. Esta probabilidade é mantida, usualmente, em um valor baixo, para evitar-se a perda de um número grande de cromossomos bons [146]. O operador de mutação pode ser implementado de várias maneiras. A codificação binária de *string* é o modo mais fácil para executá-la.

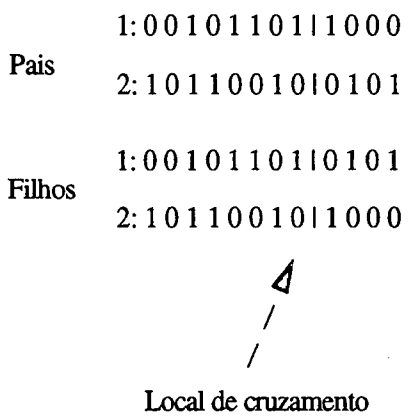

Figura 2.10: Operador *crossover* de um ponto

A tarefa da mutação em AG tem sido a de restituir a perda ou material genético inexplorado na população, com o objetivo de prevenir a convergência prematura do AG para soluções sub-ótimas [171].

Dentre os principais mecanismos de alteração genética, que recebem a denominação global de mutação, destacam-se: troca simples, translocação, inversão, deleção e adição.

Na adição, ocorre a inserção de mais um gene na cadeia, e na deleção, é justamente o oposto, ou seja, ocorre a retirada de um gene da cadeia. Geralmente, estes mecanismos não são utilizados em algoritmos genéticos, pois, alteram o comprimento da cadeia do cromossomo.

A troca simples consiste de um erro de cópia, de um ou mais genes da cadeia. Se um gene for considerado como sendo um *bit*, com valor lógico 1, a ocorrência de troca simples, levaria este *bit* (gene) para nível lógico 0 e se fosse 0, levaria para 1 (Figura 2.11). Já a inversão consiste na retirada e inserção de um pedaço da cadeia, porém, na ordem inversa da que foi retirada. Ao contrário da inversão - onde um pedaço do código é retirado e colocado no mesmo local com ordem inversa -, a translocação retira uma parte do cromossomo e coloca em outra posição do mesmo cromossomo. Estes três últimos mecanismos não alteram o comprimento original da cadeia e, como a maior parte dos trabalhos, em algoritmo genético, utilizam ca-

deia de comprimento fixo, estes são os mais comumente utilizados [2]. No entanto, como na maioria dos trabalhos com AG, este também, usa o termo mutação, como sinônimo de troca simples [2][52][92][169].

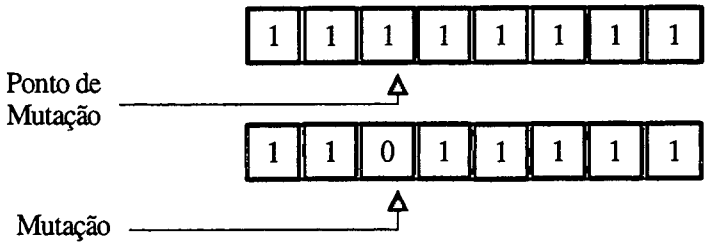

Figura 2.11: Exemplo de mutação (troca simples)

Por último, após a seleção e a aplicação dos OG, tem-se uma nova geração, a qual deve ser avaliada, visando comparar o seu grau de aptidão em relação a geração anterior. Caso tal geração não esteja apta o suficiente, deve-se repetir o processo de seleção e reprodução, até que o grau de aptidão seja aceitável [2][52].

Algoritmo Genético Simples

O Trabalho original de Holland (1975) propõe os seguintes passos principais para um algoritmo genético simples [168]:

- Geração da população inicial;
- Validação dos elementos da população e análise de convergência;
- Seleção;
- Manipulação genética.

Geração da População Inicial

A população inicial pode ser obtida através da geração aleatória de indivíduos, obedecendo condições de contorno, previamente estabelecidas pelo usuário. O usuário estabelece estas condições, tendo em vista, o seu conhecimento prévio do problema a ser otimizado. Quanto mais restringente forem as condições de contorno, mais rápida será a convergência, isso porque, os valores gerados aleatoriamente estarão

mais próximos da solução desejada [52].

O número de elementos, que comporá a população, ainda é motivo de estudos, pois existem várias heurísticas, ou seja, depende muito da experiência do usuário e do seu conhecimento prévio sobre a função a ser otimizada [40]. É claro que, quanto maior o número de elementos na população, maior é a probabilidade de convergência, tendo em vista, que aumenta a probabilidade da solução desejada ser constatada entre os elementos da população. Em contrapartida, o tempo de processamento, também aumenta. Já, no caso da população inicial ser muito pequena, ela terá o problema da perda de diversidade, isto é, o espaço de busca seria muito pequeno para ser avaliado. Desta forma, a solução obtida poderia não estar dentro do ótimo global. Conseqüentemente, a convergência seria prematura.

A população inicial, necessariamente, não precisa de ser gerada aleatoriamente, tendo em vista que, o objetivo é gerar uma população dentro de certo intervalo onde se acredita conter a resposta. Também, pode-se obter a população inicial através de um escalonamento do número de indivíduos, que compõem no intervalo especificado, isto é: se a população é de 50 indivíduos e o intervalo inicial é de 0 a 10, os indivíduos da população inicial deverão ser distribuídos uniformemente neste intervalo.

O número de elementos na população, a probabilidade de ocorrer cruzamento e a probabilidade de acontecer mutação, são denominados de parâmetros de controle dos AG [168].

Validação dos Elementos da População e Análise de Convergência

A validação é o processo de expor cada elemento da população a função de custo (objetivo) e, ao final, ordená-los de acordo com a aptidão, à esta função.

Na convergência, analisa-se o desempenho da população para ver se o objetivo foi atingido. Isto pode ser feito através de vários fatores, tais como: valores máximo, mínimo e médio da função de aptidão. Também, é relativamente comum utilizar-se o desvio padrão dos valores da função de aptidão, como forma de análise da convergência [73].

Como o AG é regido por população, se na população inicial tiver um elemento que seja a resposta exata do problema, o AG ainda assim não finalizará o processo de busca da solução. A finalização ou convergência só ocorrerá quando a aptidão

média da população estiver suficientemente estável, ou seja, quando houver pouca variação da aptidão média da população atual em relação a anterior. Isto indica que a população se adaptou ao meio, isto é, os elementos da população levam a função ao valor otimizado/desejado [168].

Utiliza-se memorizar o indivíduo mais apto, independentemente, deste fazer, ou não, parte da população atual. Assim, ao final, este será o resultado esperado [177].

Contudo, na utilização de AG pode ocorrer uma rápida convergência prematura para uma solução sub-ótima, porém, não o esperado ótimo global. Este problema é denominado convergência prematura, podendo ocorrer devido a população reduzida ou a má distribuição da população inicial, em torno do ponto sub-ótimo. Ou seja, um indivíduo próximo de um ótimo local, possui um valor de aptidão superior aos demais indivíduos da população. Conseqüentemente, o processo de seleção fará com que este indivíduo tenha grande chance de dominar a próxima geração e, assim sucessivamente, se não aparecerem outros indivíduos com melhores valores de aptidão [177][170].

Conforme pode ser visto, a convergência prematura pode ocorrer devido a uma má distribuição dos indivíduos no espaço de busca. Esta má distribuição, também recebe a denominação de perda da diversidade [177][73]. Segundo Júlio Tanomaru [177], o conceito de diversidade indica o grau em que as mais diversas regiões estão representadas no espaço de busca. Este problema pode ser amenizado através da escolha criteriosa do número de indivíduos na população, melhora da distribuição dos indivíduos da população inicial no espaço de busca e, também, impedindo a perda de diversidade nas primeiras gerações.

Seleção

A seleção tem por objetivo fazer com que somente os elementos mais aptos da geração anterior, participem do processo que irá gerar a nova população.

O processo de seleção tem início após a verificação do grau de aptidão de cada elemento, à função de custo e a verificação da não convergência dos valores.

O processo de validação fornece os elementos da população, em ordem de aptidão. Uma das formas empregadas na seleção, para pegar somente os mais aptos, é o da roleta ponderada [56]. Na roleta ponderada, imagina-se uma roleta em que

cada casa, corresponde a um indivíduo, sendo a área da casa proporcional ao valor de aptidão de cada indivíduo, de modo que, os indivíduos mais aptos têm maior probabilidade de serem selecionados pela roleta. Desta forma, a aptidão de cada indivíduo é usada para aumentar sua probabilidade de sobrevivência, e não utilizada de forma determinística [10].

Manipulação Genética

A etapa de manipulação genética consiste na aplicação de OG, isto é dos operadores *crossover* e/ou mutação, somente em alguns elementos que tiveram maior valor de aptidão, quando sorteados por meio da roleta ponderada. Ao término desta etapa terá sido gerada uma nova população, que deverá repetir os passos anteriores até que a aptidão da população seja aceitável.

Inicialmente é necessário estabelecer alguns pontos importantes, tais como: manter o tamanho da população fixo e garantir que a metade da nova população[19] seja composta por elementos obtidos por seleção da população anterior, além do que, o complemento seja de elementos manipulados pelos OG.

Como exemplo, se a população for composta de 40 elementos, na etapa de seleção deverão ser sorteados, com o auxílio da roleta, 20 elementos que passam diretamente a fazer parte da nova geração, enquanto isso, os outros 20 elementos poderão sofrer manipulação pelos OG. Com o auxílio da roleta, torna-se claro que entre os 40 elementos sorteados os mais aptos, ou melhor, os de maiores notas acumuladas aparecerão mais vezes.

Quanto aos OG, costuma-se executá-los em seqüência nos AG simples, isto é, inicialmente aplica-se o OG de *crossover* e após, os de mutação [73].

A aplicação do *crossover* implica na composição de 10 casais a partir dos 20 elementos, sendo que alguns serão acasalados e outros não. Para tanto gera-se dois números aleatórios: o primeiro, entre 0 e 1, indicará a probabilidade de ocorrer *crossover* e o segundo o local da realização do *crossover*. Caso o primeiro número gerado seja inferior ao definido pelo usuário, como probabilidade de *crossover*, realiza-se o *crossover* propriamente dito, caso contrário copia-se os pais para a nova geração

[19]Sabe-se que existem vários procedimentos para se determinar uma nova geração de população de indivíduos. Entretanto, neste trabalho optou-se por gerar a primeira metade desta população através do processo de seleção, enquanto que a metade restante é obtida pelo uso dos OG.

[177]. O segundo número aleatório, gerado no caso da ocorrência de *crossover*, indicará a posição de corte do cromossomo para efetuar o *crossover*. Para tanto o número aleatório gerado deverá estar entre 1 e $g-1$, onde g é o número de genes ou *bits* do cromossomo.

Para a aplicação do OG de mutação, há necessidade de gerar um número aleatório para cada *bit* de cada indivíduo. Este número randômico é denominado de probabilidade de mutação e, deverá ser comparado com a probabilidade de mutação estipulada pelo usuário, para o problema em questão. Caso seja inferior a esta, executa-se a mutação, caso contrário repete-se o processo para o próximo *bit* do indivíduo, até que todos os indivíduos tenham sido analisados [73][177].

A probabilidade de ocorrer mutação é sempre bem menor que a de ocorrer *crossover*. Segundo M. Sirivas e L.M. Patnaik [168], existe um compromisso entre os três parâmentros de controle do AG simples: tamanho da população, probabilidade de *crossover* e probabilidade de mutação. Muitos autores têm proposto valores para estes parâmetros visando garantir uma boa performance do AG, porém, estes valores ainda fazem parte de várias heurísticas [100].

2.5 Problema do Raciocínio Médico

Segundo Kassirer et al. [107], antes de 1950, investigações realizadas na área de psicologia, preocuparam-se em focalizar principalmente no aprendizado simples e na percepção, especialmente nos animais inferiores, mas com uma crescente ênfase no estudo dessas habilidades nos humanos. Com umas poucas exceções estes estudos produziram somente, em sua grande maioria, as teorias rudimentares, para *explicar* as várias capacidades intelectuais, observadas em relação aos humanos. Mais recentemente, alguns psicólogos começaram a devotar grande atenção as teorias de resolução de problemas humanos e aos mecanismos de processamento de informação, para explicar algumas das características observadas da resolução de problemas por humanos.

A resolução de problemas pode ser definida como uma busca no espaço de estados deste problema, enquanto que, o espaço de estados de um problema pode ser considerado como o conjunto de configurações possíveis deste problema [154]. Sabe-se que o humano em sua tarefa de resolver problemas, tem um único objetivo em mente, que é traçar um caminho entre os estados iniciais e os finais (ou metas), da solução

do mesmo. Desta forma, os processos de diagnóstico médico podem ser vistos como um passo para a resolução de problemas. Contudo, os processos que fundamentam esta realização intelectual são entendidos pobremente. Tentativas para explicar os componentes do processo de diagnóstico, - um dos elementos chaves da resolução de problemas clínicos - derivam-se, sobretudo, das reflexões de clínicos experientes em seus próprios processos de resolução de problemas. Tais reflexões têm fornecido princípios muito importantes e um enfoque geral, dentro do qual, os estudantes podem desenvolver uma compreensão das tarefas de tomada de decisão da profissão. Entretanto, devido a estes princípios serem gerais e, em muitas situações surgirem com muitas abordagens possíveis diferentes, a aplicação destes princípios é um ponto inicial neste processo, além do que eles são simples e raramente especificados. Esta imprecisão favorece a crença de que os processos de resolução de problemas, usados pelos clínicos, não podem ser examinados e, além do que, devido ao conceito de julgamento clínico ser de difícil compreensão, o médico experiente se utiliza do processo de deduzir da observação, o que ele próprio pratica, mas é incapaz de articular [107].

Além disso torna-se difícil também investigar o processo de diagnóstico, particularmente por causa do amplo escopo, incluído pelo material submetido e as incertezas inerentes na informação médica. Assim os mecanismos mentais e o processo de raciocínio, pelo qual, os clínicos chegam ao diagnóstico, é ainda mal conhecido. Ele envolve, simultaneamente, processos lógicos, avaliação probabilística, encadeamento causal, e muitos outros processos parcialmente entendidos [107]. Por isso é difícil tentar simular num computador, um modelo complexo como é o do raciocínio humano. Portanto, o diagnóstico médico é considerado um caso especial e crítico no desenvolvimento de um sistema automatizado, principalmente, no que diz respeito ao processo de elicitação e representação do conhecimento.

O médico tem dificuldade de articular de maneira lógica, direta e consistente, como ele chegou a uma determinada conclusão (diagnóstico), devido a uma série de fatores. Talvez o principal deles seja que o diagnóstico médico não trata com um assunto conhecido, pois diz respeito a um sistema natural (sistema humano), o qual se conhece apenas parcialmente. Não é algo, que se domina verdadeiramente. Além disso não se conhece a maioria das variáveis, como nem todas, ou quase todas as soluções possíveis, envolvidas neste processo. O médico tem uma idéia da situação mas não consegue dizer com clareza como ele chegou a uma determinada conclusão. Sabe-se, que ele utiliza da sua experiência anterior, bem como, seu *feeling*. E ainda

há elementos específicos para realizar o processo de diagnóstico[20]. Contudo, este tipo de sistema não é só difícil para o médico, como também, para o engenheiro de conhecimento.

Durante o processo de diagnóstico, o médico realiza uma série de inferências sobre a natureza das disfunções do corpo. Estas inferências são derivadas das observações existentes, ou seja, dados consistentes sobre a história do paciente, sintomas, sinais, testes de rotinas, testes invasivos, respostas a várias manipulações, o tempo de curso de alguns eventos, os conhecimentos clínicos, fisiológicos, bioquímicos, anatômicos e patológicos, sobre casos semelhantes e a sua biologia subjacente, a experiência prévia do médico em realizar diagnósticos do mesmo tipo, bem como, o senso comum e a intuição (o *olho clínico*). O raciocínio inferencial continua até que o médico obtenha uma classe de diagnóstico, suficientemente aceitável, para estabelecer um diagnóstico, fornecendo uma ação terapêutica. Quando feita as inferências de diagnóstico de dados clínicos, o médico usa várias estratégias para combinar, integrar e interpretar os dados. Os médicos utilizam amplamente suas experiências passadas, ou seja, as heurísticas, no processo de reunir e interpretar informações. Estas heurísticas são essenciais, pois reduzem a necessidade de se ter um número grande de questões desnecessárias, ordenar testes de diagnósticos supérfluos e tornar a tarefa de informação manejável e eficiente [106].

Estudos de conhecimento humano sugerem que os tipos de heurísticas usadas dependem da natureza do problema clínico a ser tratado e na experiência anterior do médico. No caso dos não-especialistas, eles tendem a usar estratégias de pesquisas não-seletivas, enquanto que os especialistas tipicamente empregam as abordagens de diagnósticos adaptadas à experiência anterior deles.

Entretanto, o processo de diagnóstico ainda focaliza desenvolver uma ou mais hipóteses de diagnóstico. Uma hipótese pode ser geral ou específica. Pode tomar várias formas, incluindo um estado, uma desordem clínica, uma síndrome ou uma doença específica. A geração de hípoteses - que é um processo descrito como o ativador de diagnóstico -, é a primeira, no início de um julgamento de diagnóstico. Assim, acionar a formulação de uma hipótese preliminar baseado somente em umas poucas observações, é criticamente independente da capacidade cognitiva para relatar uma situação nova à experiência passada. Tipicamente, os médicos acionam as hipóteses iniciais, meramente em relação a idade, sexo, raça e queixas presentes em pacientes,

[20] O diagnóstico é a conseqüência da validação de uma hipótese.

mas algumas vezes, tais hipóteses, emergem exclusivamente de julgamentos físicos ou dados laboratoriais. Hipóteses adicionais são ativadas como novos julgamentos emergentes [107][106].

Ainda, de acordo com Kassirer et al. [106], o raciocínio médico procede por modificação e refinamento de hipóteses, progressivamente. Algumas hipóteses são feitas mais especificamente, outras são ativadas previamente, de modo que são eliminadas, e outras são adicionadas. Na literatura, estudos anteriores, indicam que o ser humano tem capacidade de ativar cerca de sete hipóteses por vez, restringida pela capacidade limitada da memória atual ser reduzida. Também, não é claro, quanto do processo de diagnóstico é dirigido por hipóteses e, quanto é dirigido pela avaliação de dados de história, de exames físicos ou laboratoriais.

Um outro ponto, diz respeito a verificação da hipótese de diagnóstico, a qual é a próxima e penúltima tarefa. Devido ao processo de diagnóstico ser inferencial, todas as hipóteses de diagnóstico, necessariamente, refletem uma confiança ou uma convicção para o médico, no sentido da natureza da condição do paciente. Verificar uma hipótese envolve avaliar sua coerência, sua adequabilidade e sua natureza parcimoniosa. Também requer eliminar hipóteses, competindo em um processo análogo à hipóteses científicas não comprovadas. Este processo produz uma ou mais hipóteses, que formam a base para a próxima etapa no manuseio do paciente, ou seja, observar o paciente, ordenar testes adicionais ou tratar o paciente [112].

Desta forma, o diagrama em blocos da Figura 2.12, mostrará os principais elementos envolvidos neste processo de diagnóstico.

Contudo, há ainda o problema relacionado ao tipo de raciocínio que o médico usa. Conforme Kassirer et al. [106], três estratégias de raciocínio podem ser consideradas: probabilístico, causal e determinístico. Raciocínio probabilístico lida com as relações estatísticas entre variáveis clínicas, e é freqüentemente, usado em cálculos formais de probabilidades de doenças. Ele é especialmente útil em evocar hipóteses de diagnóstico, avaliar o significado de julgamentos clínicos e resultados de testes. Raciocínio causal é baseado no modelo fisiológico, ou relações de causa e efeito entre variáveis clínicas e avaliação do paciente, no sentido de coerência e completude contrária ao do modelo. Ele funciona, especialmente e efetivamente, na verificação de hipóteses de diagnóstico. Já o raciocínio determinístico consiste de conjuntos de regras compiladas e geradas de rotinas bem definidas. Neste caso, são usadas para

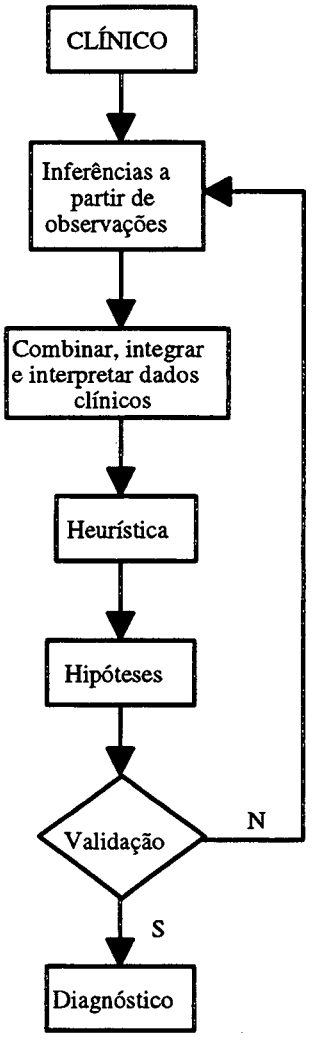

Figura 2.12: Elementos do processo de diagnóstico médico baseado em heurística

solucionar problemas humanos que podem derivar da ativação e implementação de tais regras.

Além disso, muitas das tarefas associadas à decisão médica são de difícil sistematização, e caem no domínio do que chamamos problemas mal-definidos, ou seja, que não podem ser expressos por formalismos matemáticos, estatísticos ou lógicos. O reconhecimento de padrões espaciais ou temporais complexos (por exemplo, classificação de um traçado de EEG), é um desses problemas, muito freqüentes em medicina, que são de domínio do chamado *raciocínio abdutivo*[21] [155].

[21] *Abductive* - Esta palavra, no sentido deste trabalho, não tem o seu correspondente em por-

Abdução[22] é um tipo importante de inferência. Raciocínio de diagnóstico médico, por exemplo, é provável ser abdutivo, preferivelmente do que dedutivo. Um médico observa que o paciente tem certos sintomas e conclue dessas observações que o paciente está provavelmente sofrendo de uma doença particular. Esta inferência não é dedutiva porque algumas vezes um paciente com um dado sintoma A não necessariamente sofre de uma doença B. O mesmo sintoma pode ser causado por mais do que uma doença. A implicação por outro lado é uma outra forma de ver que: se um paciente sofre de uma doença B, então, ele tem o sintoma A. Daí, o médico conclui a presença de uma doença porque ele tem observado o sintoma e, assim, o raciocínio usado por ele encaixa-se no abdutivo [156].

Contudo, abdução não é uma regra de inferência lógica válida. Mais precisamente, o processo de raciocínio abdutivo pode ser descrito como: *If* B é um conseqüente de A e B é verdadeiro, inferir que A é verdadeiro (isto é, como um *modus ponens* invertido). Formalmente, abdução pode ser representado por: $A \rightarrow B, B \vdash A$ [46].

O raciocínio abdutivo pode ser exemplificado da seguinte forma: um determinado indivíduo bateu seu carro, em conseqüência amassou um dos seus parachoques. Assim, se o carro está amassado, conclui-se que ele bateu. Porém, o que se está concluindo como verdade pode levar a várias outras causas, sendo que, talvés, esta seja a mais comum de todas as possíveis. Desta forma, pode-se observar que não é utilizado um raciocínio lógico válido.

Um outro exemplo prático de uma inferência para o raciocínio abdutivo pode ser dado da seguinte forma: se você vê que as ruas estão úmidas, então pode inferir que choveu. Você sabe então, que se choveu as ruas estão úmidas, utiliza assim esta parte do conhecimento para inferir que a melhor explicação para esta observação, ou seja, que as ruas estão úmidas, é a suposição de que tenha chovido. Claramente, isto não é uma inferência dedutiva. Então é inteiramente possível as ruas estarem úmidas mesmo que não tenha chovido. A inferência é abdutiva: a suposição que tenha chovido parece provar a melhor explicação.

Em muitos domínios, o raciocínio abdutivo é particularmente útil, caso alguma medida de certeza seja anexada às expressões resultantes. Estas medidas de certeza,

---

tuguês. Então, adotar-se-á *Abdutivo*.

[22] *Abduction* - Esta palavra, no sentido deste trabalho, não tem o seu correspondente em português. Então, adotar-se-á *Abdução*.

quantificam o risco do processo de raciocínio abdutivo estar errado [157]. Para tanto, há dois casos a serem tratados. O primeiro, diz respeito, por exemplo, quando se tem $A \rightarrow B$ (com uma medida de certeza igual a um valor K), $B \vdash A$. Porém, além deste tipo de formalismo não ser uma regra de inferência lógica válida, com qual medida de certeza, pode-se concluir A? Segundo, dados outros antecedentes além de A, que podem ter produzido B, como $X \rightarrow B$ (com uma medida de certeza igual a um valor L), $B \vdash X$ e $Y \rightarrow B$ (com uma medida de certeza igual a um valor M), $B \vdash Y$, quais os critérios para concluir que, por exemplo, a terceira proposição é a solução obtida? Será que aquela proposição que apresente, em seu conseqüente, o maior valor de medida de certeza, é considerada a solução plausível do problema em questão? Desta forma, o processo de raciocínio abdutivo pode ser concluído corretamente ou com um certo grau de erro.

Contudo, ainda o conhecimento do processo de diagnóstico permanece tão limitado para definir precisamente as situações clínicas com respeito ao tipo de raciocínio que é mais desejável. Pode-se opinar que, um médico especialista usa todas ou algumas das abordagens mencionadas anteriormente sobre os tipos de raciocínio para diagnosticar uma doença, mesmo em algum conflito de diagnóstico. Nós ainda temos que aprender muito sobre diagnóstico para fazer esta assertiva.

Capítulo 3

Sistema Proposto

Inicialmente vamos apresentar os motivos que nos levaram a propor a criação de um HES, onde uma série de conceitos básicos importantes foram empregados e estes justificam as decisões tomadas para o desenvolvimento deste trabalho. Após será apresentada uma descrição geral do sistema, de seus componentes e de como são realizadas as diferentes tarefas implementadas.

3.1 Fundamentação do Sistema

Uma série de fatores nos levou a optar pelos diferentes componentes do sistema, fatores estes que são explicados a seguir.

Uma das tarefas mais árduas na área de IA é o chamado processo de AC, mais especificamente a fase de EC. Este tipo de processo é considerado um dos mais importantes no desenvolvimento de um SE (ver a Figura 3.1). Caso esta etapa seja feita de uma maneira não eficaz, o SE fica prejudicado, pois, provavelmente não atingirá o objetivo proposto, ou seja, obter um SE que responda com um certo grau de confiança, o conhecimento elicitado de um especialista de domínio para uma determinada tarefa. Existem várias técnicas de EC que são empregadas para minimizar o problema de AC, porém nem sempre estas técnicas, como referenciadas em [18], são as melhores para um dado caso, pois dependem do grau de conhecimento, da dinâmica, da disponibilidade e do interesse por parte do especialista, bem como do engenheiro de conhecimento. Há casos em que normalmente o especialista precisa ceder um tempo razoável para cada seção de elicitação de conhecimento, além do que, para alcançar um produto final adequado, há a necessidade de se ter uma interface especialista/engenheiro de conhecimento/máquina que apresente um nível razoável para se atingir o objetivo final. Contudo, o tempo gasto durante o processo

de extração de conhecimento é demasiado grande e muitas vezes é cansativo, o que faz com que algumas vezes o especialista fique desmotivado a dar continuidade ao trabalho.

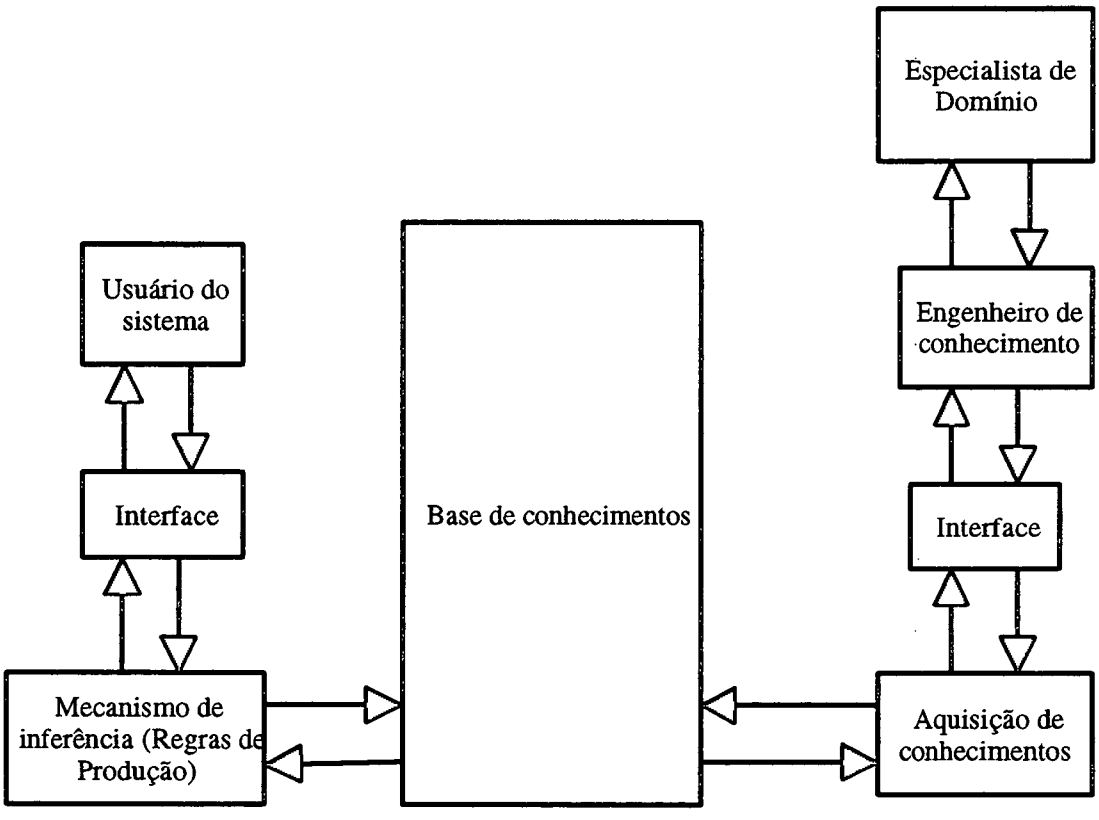

Figura 3.1: Diagrama de blocos de um SE simbólico básico

Um outro problema enfrentado no desenvolvimento de SE, diz respeito ao modo de se representar o conhecimento do especialista. Existem várias formas de representar o raciocínio humano. Uma delas é por meio de regras de produção, como já mencionado nos Capítulos 1 e 2, porém, há alguns problemas com este tipo de representação de conhecimento. Por exemplo, o número de regras necessárias para descrever o problema em questão, o problema de conflito de regras e a atualização constante da base de conhecimento.

Uma outra maneira de se minimizar os problemas mencionados no início deste capítulo é utilizar o paradigma conexionista (ver Figura 3.2). Como já descrito nos Capítulos 1 e 2, é uma outra forma de se representar o conhecimento de um especi-

alista de um determinado domínio. Porém, mesmo que a extração de conhecimento do especialista seja por meio de um conjunto de exemplos e que, comparado com o paradigma simbólico o tempo de convívio com o especialista é bem menor, mesmo assim há uma dificuldade no uso deste tipo de paradigma. Dificuldade esta relacionada ao conhecimento do especialista humano estar embutido em uma grande massa de pesos e conexões. Desta forma, quando se deseja uma explicação da solução obtida, a RNA falha em poder fornecer a linha de raciocínio empregada, uma vez que o seu conhecimento é opaco.

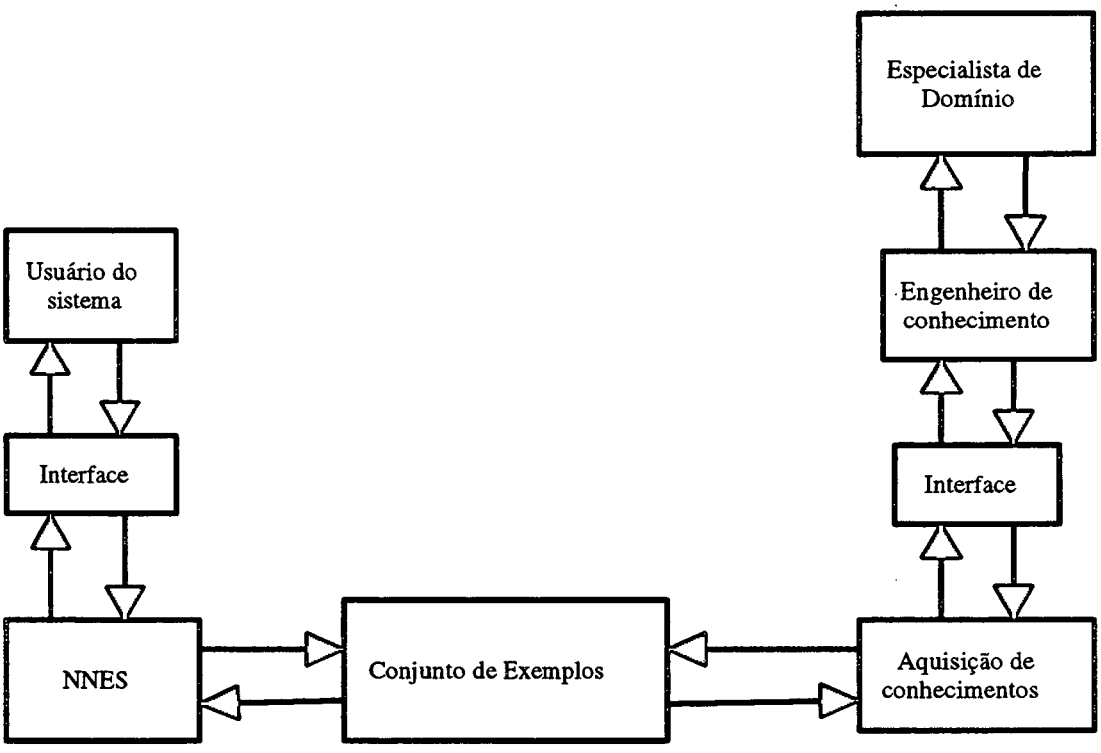

Figura 3.2: Diagrama de blocos de um NNES típico

Assim, antes de se partir para a próxima seção, segue abaixo uma breve lista das vantagens e desvantagens de cada um dos dois paradigmas mencionados anteriormente e que demonstra claramente a sua complementariedade [144]:

- Vantagens do paradigma simbólico:
  - Permite a inserção direta dos conhecimentos *à priori* sobre o problema;

  - Pode fornecer com uma certa facilidade uma explicação direta para as respostas obtidas pelo sistema (seqüência de regras aplicadas);

- Desvantagens do paradigma simbólico:
  - Aprendizado não é uma de suas capacidades intrínsecas;
  - Dificuldade de construção (explicitação) da base de conhecimentos;
  - Fragilidade frente as informações aproximadas e incompletas;
  - Processamento usualmente seqüencial e lento;
  - Dificuldade no processo de AC;
  - Problemas no modo de representar o conhecimento de um dado domínio, por exemplo, por regras de produção, no sentido de determinar o número de regras necessárias para descrever o problema em questão, bem como o problema de conflito de regras e a atualização constante da base de conhecimento;

- Vantagens do paradigma conexionista:
  - Capacidade de aprendizado à partir de exemplos, o que permite uma aquisição fácil de conhecimentos;
  - Robustez das respostas devido a sua capacidade de generalização;
  - Possibilidade de exploração do paralelismo. A recuperação das informações é extremamente rápida;

- Desvantagens do paradigma conexionista:
  - Dificuldade de determinação dos parâmetros de aprendizagem e da topologia da rede em função do problema;
  - O conhecimento adquirido é codificado em pesos e ligações e de difícil interpretação (*caixa-preta*);
  - Aprendizado dependente dos estados iniciais da rede.

A partir deste estudo inicial nossa intenção foi a de se propor e exemplificar um sistema especialista híbrido, cuja característica principal seja a capacidade de aprender a extrair conhecimento a partir de uma estrutura inicial de base de conhecimento e de um conjunto de exemplos. Isto é, utilizar ambos os paradigmas conexionista e simbólico, combinando-os de forma que se possa absorver o melhor de ambas as tecnologias, de modo que se consiga, principalmente, facilitar a tarefa de AC durante a construção de um SE.

3.2 Descrição do Sistema

As Tabelas 3.1 e 3.2 nos mostram resumidamente as características gerais dos HES descritos no Capítulo 1. Estes sistemas híbridos foram encontrados na literatura e demonstram a tendência a uma unificação entre os paradigmas conexionista e simbólico, através de um HES, o qual tem por objetivo aproveitar as melhores características de cada um, no sentido de compensar as suas limitações em certos pontos.

Algumas definições se fazem necessárias para a compreensão das Tabelas 3.1 e 3.2, conforme segue:

Abruptas - A

Aprendizado - APR

Aprendizado Supervisionado - AS

Aprendizado Não-Supervisionado - ANS

Algoritmo de Aprendizado - AP

Baseado/Retropropagação - BRP

Booleanas - B

Completas - C

Conexões - CX

Direta Estática - DE

Direta com Dinâmica - DD

Distribuição Probabilística - DP

Esparsas - E

Extração/Regras - ER

Função de Pertinência - FP

Fuzzy - F

Gaussiana - G

HES com Abordagem de - HES/A

Implementado - IMP

Lingüística - L

Lógica *Fuzzy* - LF

Não Mencionada - NM

Número de Camadas - NC

Probabilidade - P

Probabilidade/*Fuzzy* - P/F

Proposto - PRO

Sigmóide - SG

Sigmóide/Gaussiana - SG/G

Sistema é apenas Proposto ou foi Implementado - SP/I

Tangente Hiperbólica - TH

Tangente Hiperbólica/Sigmóide - TH/SG

Tratamento de Incertezas ou Imprecisões - TII

Tipos de Neurônios - TN

Tipo de Função de Ativação - TFA

Topologia da RNA - TRNA

Valores de Entrada - VE

Tabela 3.1: Características gerais de algumas abordagens para HES

Características							
HES/A	TRNA	APR	AP	NC	VE	TN	CX
L.M. Fu	DE	AS	BRP	4	B	A	E
R. Sun	DD	AS	BRP	3	DP	A	C
M.M. Gupta	DE	AS	BRP	3 ou mais	FP	OU	C
S. Mitra et al.	DE	AS	BRP	3 ou mais	FP	E/OU	C
W. Pedrycz	DE	AS	BRP	3	NM	E/OU	C
H.C. Fu et al.	DE	AS	BRP	5	FP	E/OU	E
Y. Hayashi	DE	AS	BRP	3	FP	A	E
S.K. Halgamuge et al.	DE	AS	BRP	3	L	E/OU	C
J.-S.R. Jang	DE	AS	BRP	3 ou mais	FP	A	C
J. Zimmermann et al.	RBF	AS	BRP	4	L	E/OU	C
R. Machado	DE	AS	BRP	3 ou mais	L/B	E/OU	C

Tabela 3.2: Características gerais de algumas abordagens para HES

Características						
HES/A	TII	TFA	Grafos E/OU	ER	AG	SP/I?
L.M. Fu	FC	Equação 1.1	Sim	A	Nao	IMP
R. Sun	P/F	NM	Não	Não	Não	IMP
M.M. Gupta	F	TH/SG	Não	Não	Não	IMP
S. Mitra et al.	F	SG	Não	A/F	Não	IMP
W. Pedrycz	F	NM	Não	Não	Sim	IMP
H.C. Fu et al.	F	NM	Não	Não	Não	IMP
Y. Hayashi	F	SG	Não	A/F	Não	IMP
S.K. Halgamuge et al.	F	SG	Não	Não	Não	IMP
J.-S.R. Jang	F	SG	Não	Não	Não	IMP
J. Zimmermann et al.	F	SG/G	Não	Não	Não	IMP
R. Machado	F	SG	Sim	Não	Sim	PRO

Como pode ser observado nas Tabelas 3.1 e 3.2, as abordagens dos HES de R. Sun e H.C. Fu et al. não se aplicam aos objetivos deste trabalho, porque a primeira utiliza uma RNA com retardo, os valores de entrada são tratados como distribuição probabilística e as conexões entre os neurônios são esparsas. Na segunda as conexões entre neurônios levam em conta o uso incompleto de ligações. Além disso, ambas as abordagens não foram muito significativas, porque a metodologia a ser aplicada para o sistema aqui proposto diz respeito a utilizar uma RNA com topologia direta estática, com ligações entre neurônios completas e com os dados de entrada da RNA acessíveis para valores lingüísticos, numéricos e booleanos.

As abordagens de L.M. Fu e M.M. Gupta são bastante interessantes, mas na primeira somente são utilizados dados de entrada para a RNA na forma booleana e o tratamento de incertezas ou imprecisões é realizado pelo uso de FC. Na segunda, mesmo que esta se utilize de neurônios *fuzzy* e dos conectivos da lógica *fuzzy*, esta se aplica somente à camadas de neurônios OU. Devido a árdua tarefa de EC de um dado domínio, não se deve prender somente ao raciocínio determinístico, ou seja, sim ou não, mas sim, se aproximar ao raciocínio dos seres humanos. Neste aspecto sendo mais maleáveis de modo a usar outros tipos de entradas, por exemplo, numéricas e/ou lingüísticas. Além disso, como existe o interesse em traduzir um conjunto de regras iniciais em uma RNA, acredita-se que uma RNA com neurônios *fuzzy* do tipo

E na camada intermediária e OU na camada de saída poder-se-ia mapear melhor este conhecimento. Então, descartou-se a aplicabilidade destas abordagens, mesmo que estas sejam aplicáveis à RNA diretas estáticas com algoritmo de aprendizado baseado no de retropropagação.

Quanto a abordagem sugerida por S.K. Halgamuge, não se considerou adequada para se utilizar no desenvolvimento do sistema híbrido aqui proposto. Isto se deve ao fato desta se basear em uma filosofia matemática específica para RNA que apresenta mais de três camadas e que não poder-se-ía adequá-la ao nosso sistema, o qual possui apenas três camadas, devido ser este formalismo matemático intrínseco para este tipo de RNA particular. Além disso, não se vê vantagem de se utilizar uma RNA com mais de três camadas, pelos seguintes motivos: já foi comprovado pela comunidade científica (Veja a referência [196]) que a maioria dos problemas de IA usando o paradigma conexionista são solucionados com uma RNA de apenas três camadas, o tempo de consumo computacional normalmente será muito maior (nos artigos referentes a esta abordagem, não se comenta sobre este tópico) e o número de neurônios para as camadas intermediárias, ou seja, quantos neurônios serão necessários para cada uma das camadas intermediárias de forma a se obter um resultado plausível na saída de tal RNA, ainda é um assunto de estudo. É de nosso conhecimento algumas heurísticas para se amenizar esta dificuldade, mas se para uma única camada intermediária já é um pouco complicado, imagina para mais de uma. Isto se aplica também a abordagem de H.C. Fu, que utiliza cinco camadas na RNA e a treina com algoritmo baseado no de retropropagação.

A abordagem dada por J. Zimmermann et al. também não se aplica aos objetivos deste trabalho por usar uma RNA do tipo RBF, assim como também a abordagem de R. Machado, por apenas ter sido proposta Além do que, pouco se menciona sobre o desenvolvimento matemático do algoritmo de aprendizado proposto por ele.

Em relação a abordagem sugerida por W. Pedrycz, não se achou viável devido a esta se aplicar a um conjunto de dados de entrada sem no entanto ter definido o tipo, isto é, se poderiam ser booleanos, numéricos ou lingüísticos, ou ambas. Além disso, está fora do objetivo atual deste trabalho usar o AG para o treinamento, mas sim para melhorar a topologia da RNA. Um outro motivo diz respeito a implementação deste sistema, por não estarem muito claros os seus resultados.

Por fim, a abordagem de Y. Hayashi também não se aplicou ao desenvolvimento

do HES aqui proposto devido esta utilizar na entrada da RNA grupos de células distintas na forma de abruptas e *fuzzy*. Na metodologia proposta para esta abordagem existem conexões diretas da entrada para a saída da RNA, de forma a serem consideradas esparsas. Quanto a sugerida por S. Mitra et al., foi uma das que se aproximou mais ao interesse desta pesquisa, apesar de usar funções de pertinência na entrada da RNA no sentido de mapear os dados de entrada como números *fuzzy*-π. Além disso, é de interesse da pesquisa extrair regras após a RNA ser treinada, testada e refinada, inspirada, principalmente, no algoritmo proposto para a extração de regras *fuzzy* desta última abordagem.

Com as vantagens e as desvantagens levantadas anteriormente sobre cada HES, a Figura 3.3 mostra a arquitetura do SE proposto neste trabalho. Ele é composto por dois módulos principais: NNES e RBES.

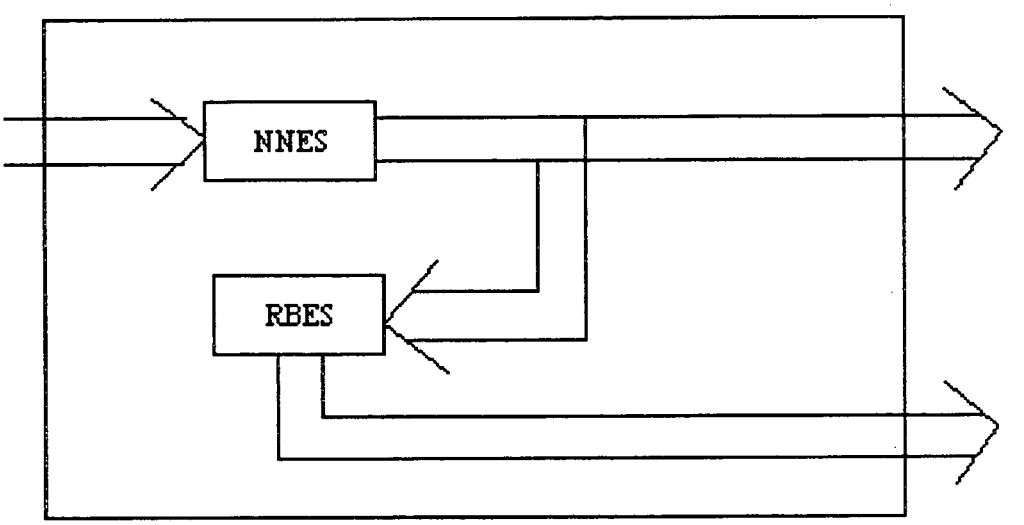

Figura 3.3: Diagrama de blocos do HES

Para construir o primeiro bloco, isto é, o NNES, é necessário que se tenha algumas regras iniciais e um conjunto de exemplos adquiridos da etapa de AC. Então, enquanto a rede inicial representa relações entre conceitos e conexões, o conjunto de exemplos refinará o NNES. Neste último processo o algoritmo prevê modificações não somente nos pesos das conexões, mas também, na estrutura da rede. Ele usa esta topologia e gera e/ou elimina conexões, que não tenham estado nas regras iniciais. Além do que, ele pode também, ocasionalmente, gerar mais conceitos que não esta-

vam nas regras iniciais. Portanto o sistema traduz como regras, aquelas regras iniciais que o especialista não foi capaz de articular. As regras iniciais, após à extração, sofrem um tratamento devido ao tipo de variáveis aplicadas nas entradas da rede, onde elas representam tipos diferentes de conceitos, isto é, quantitativas, lingüísticas ou booleanas [34][19][69][134]. Além disso, uma modificação estrutural da RNA consiste em determinar o número de neurônios da camada intermediária usando-se um dos algoritmos evolucionários propostos pela Computação Evolucionária, ou seja, o AG [69].

Em resumo, o sistema proposto tem dois tipos de dados: regras iniciais e conjunto de exemplos. As regras iniciais são usadas para criar o NNES inicial, o qual é refinado através de exemplos. A partir do NNES refinado, regras podem ser inferidas.

Como já mencionado anteriormente, devido a dificuldade que um sistema conexionista tem em dar uma explicação sobre uma dada conclusão, sugere-se para auxiliar neste problema um sistema simbólico. Métodos de extração de regras à partir de RNA também foram estudados, resultando em uma nova proposta que possibilita uma integração bidirecional completa entre os paradigmas conexionista e simbólico. Veja o Capítulo 1 que trata em parte sobre este assunto e o Capítulo 6 que especifica com maiores detalhes o método escolhido para a extração de regras.

O conhecimento adquirido no NNES é mapeado na saída deste para um outro sistema, ou seja, o RBES. Acredita-se que com o conhecimento abstraído do NNES, por este já estar refinado, ou seja, ter sido composto por algumas regras elicitadas do especialista de um domínio e mais um conjunto de exemplos, bem como treinado e testado, o conhecimento obtido na saída do NNES e que vai ser usado para formar a base de conhecimento do próximo sistema, RBES, seja melhorado. Além de que o RBES servirá para efetuar a explanação sobre a resposta atingida na saída do sistema conexionista, justificando ao usuário o porquê de tal resultado. Todavia, caso fossem utilizadas apenas as regras extraídas no processo de EC, na etapa de AC para formar diretamente a base de conhecimento do RBES, sem ser mapeado primeiramente em um NNES, acredita-se que o desempenho do sistema seria inferior. Tem-se que considerar também que nas regras elicitadas inicialmente são incluídas imprecisões, onde o método escolhido foi o da modelagem *fuzzy*. A teoria de lógica *fuzzy* concede um enfoque matemático de bom nível, para representar o conhecimento impreciso, o qual, é também, um dos objetivos deste trabalho [34][19][76][77][195].

Desta forma, como pode ser observado nas Tabelas 3.1 e 3.2, o HES aqui proposto engloba algumas das características gerais daqueles mencionados e comentados neste Capítulo, nos seguintes pontos:

- Das abordagens sobre os HES descritos aqui somente a do L.M. Fu utiliza os grafos E/OU como uma etapa intermediária na montagem de uma RNA inicial. Acreditamos que com esta etapa o desenvolvimento do NNES inicial seja facilitado. Neste sentido, é possível se determinar inicialmente o número de neurônios da camada de entrada, da saída e da intermediária do NNES;

- Quanto ao tratamento de incertezas e imprecisões do raciocínio humano, optou-se por modelar as imprecisões por lógica *fuzzy*, por se acreditar que esta tem maiores chances de ser similar ao conhecimento de um especialista de um dado domínio;

- Como um dos objetivos deste trabalho é elicitar o conhecimento de um especialista na forma de regras de produção e mapeá-las em um sistema conexionista, a melhor forma que se encontrou foi traduzir estas regras em uma RNA E/OU e não em uma RNA apenas, por exemplo, com neurônios OU, como é o caso da abordagem de M.M. Gupta;

- Um dos interesses deste trabalho é lidar com redes diretas estáticas e não com aquelas com ciclos e com dinâmicas. Desta forma, as abordagens dadas por R. Sun e J. Zimmermann et al., saem deste escopo;

- Com respeito ao tipo de dados de entrada a serem fornecidos ou recebidos pelo sistema conexionista foi optado para serem: lingüísticos, booleanos e numéricos, no sentido de dar um maior grau de liberdade na representação do conhecimento de um dado especialista de um domínio. Neste aspecto pode ser observado, com as abordagens dos HES vistas aqui, somente a da S. Mitra et al. utilizou estes vários tipos de entradas para o paradigma conexionista;

- Devido as particularidades da RNA ser *fuzzy*, o NNES desenvolvido neste trabalho possui interligações entre as unidades como sendo totalmente conectadas;

- Utilizou somente três camadas para o NNES ao invé de mais;

- Com relação ao tipo de aprendizado da RNA utilizou-se o supervisionado pelo motivo de ser pré-determinado um conjunto de entradas/saídas. Por

exemplo, a necessidade de se realizar o diagnóstico de uma dada doença, onde um conjunto de sintomas e diagnósticos são fornecidos por um determinado especialista da área, nos possibilita através de uma RNA supervisionada nos conceder a classificação da referida doença;

- Devido a simplicidade e robustez do algoritmo de aprendizado de retropropagação, utilizou-se para o treinamento do NNES. Porém, inspirado apenas em parte deste. Várias das modificações sugeridas são abordadas com maiores detalhes nos Capítulos 4 e 5;

- O AG foi aqui utilizado como uma ferramenta no auxílio da escolha da melhor topologia do NNES. Das abordagens estudadas pode-se observar que a maioria delas não se preocupou com este detalhe, e quando usado o AG foi para o aprendizado propriamente dito da RNA;

- Quanto a função de ativação não se viu a necessidade de desenvolver matematicamente uma outra que não fosse a tangente hiperbólica. Para os objetivos deste trabalho acredita-se que esta seja plausível.

- Optou-se por fazer um tratamento dos dados de entrada para o NNES em valores *fuzzy* representados por graus de pertinência. Isto se deve ao fato de que não se está interessado em usufruir das funções de pertinência nas entradas do NNES, mas sim de valores (graus de pertinência) que descrevam os dados fornecidos por um especialista de um dado domínio, no sentido de que se aproxime o mais perto possível do que é o real;

- De todas as abordagens dos HES comentadas neste Capítulo apenas a da S. Mitra et al. e a de Y. Hayashi sugerem algum método de extração de regras *fuzzy* após uma RNA é treinada e refinada. Desta forma, o algoritmo proposto neste trabalho para a extração de regras *fuzzy* tem inspiração principalmente no da S. Mitra et al. por ser fácil e estar próximo aos interesses deste trabalho. Maiores detalhes podem ser vistos no Capítulo 6.

Capítulo 4

Metodologia da Pesquisa

Uma das dificuldades em elicitar o conhecimento de especialistas de domínio é quando se deseja obter um conjunto adequado de regras. Primeiramente, em muitos campos, os especialistas não são capazes de perceber ou articular qual (ou quais) conhecimento(s) eles usam para solucionar seus problemas. Em segundo, freqüentemente vários especialistas têm diferentes explicações para suas decisões e, algumas vezes, eles discordam sobre as decisões. E por último, o desempenho do RBES não pode ser melhor do que aquele do especialista a partir do qual foi modelado [42].

Por outro lado, um NNES necessita somente de um conjunto de exemplos para aprender a representar o problema considerado. Contudo, como o conhecimento é freqüentemente distribuído nas conexões da rede, explicar como o NNES alcançou uma conclusão é muito difícil, exceto se a representação de conhecimento é localizada. Isto é extremamente importante para o caso deste trabalho, onde os médicos são os usuários em potencial. Assim, a metodologia proposta lida com uma abordagem híbrida, explorando as características e propriedades desejáveis dos paradigmas conexionista e simbólico (Figura 4.1).

Nas próximas seções, serão descritos com detalhes, cada módulo apresentado na Figura 4.1, referente ao sistema geral proposto neste trabalho.

4.1 Aquisição de Conhecimento

A tarefa de EC consiste em extrair o conhecimento do especialista de domínio (neste caso, o especialista médico). Neste caso, a meta principal é minimizar as dificuldades intrínsicas do processo de AC, pois, quando se está trabalhando com

um sistema simbólico, tenta-se extrair regras, que é a maneira mais usual de se representar o conhecimento. Quando se trata de um sistema conexionista (neural), tenta-se obter exemplos. Já o sistema a ser tratado neste trabalho é um sistema híbrido. Ele envolve estes dois paradigmas: extrai-se as regras que forem possíveis do médico sem forçá-lo a um trabalho muito longo, e uma série de exemplos (Figura 4.2).

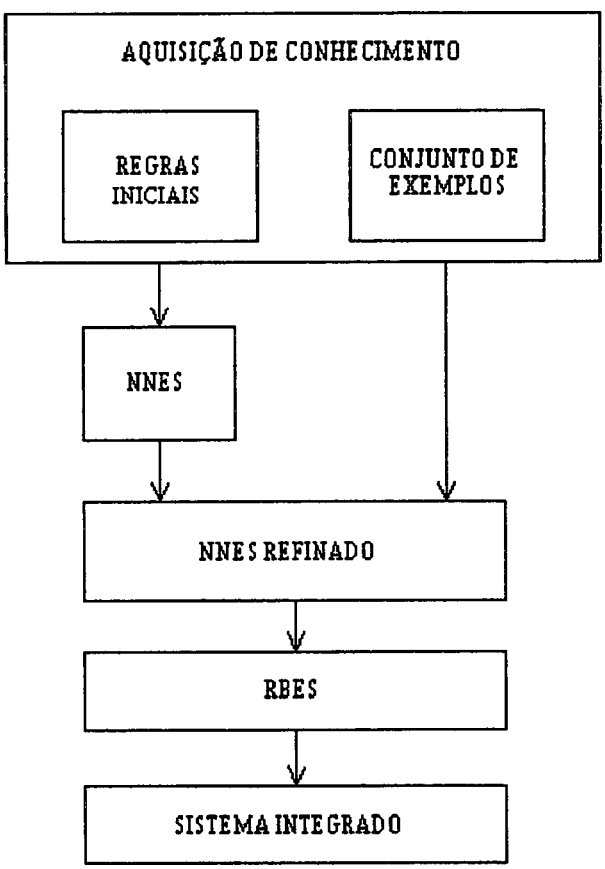

Figura 4.1: Diagrama de bloco do sistema geral

Durante o processo de EC é possível obter-se algumas regras do especialista de um domínio, as quais aqui são consideradas como *regras iniciais*, isto é, são aquelas regras que foram possíveis de ser elicitadas de um dado domínio de um especialista referentes ao seu conhecimento. Conjectura-se, ainda que, se estas regras iniciais correspondem ao conhecimento mínimo deste especialista, ter-se-ia então o conhecimento mínimo descrito nestas regras e que estas seriam consideradas ortogonais. Desta forma, quando se fosse mapeá-las em uma RNA, este conhecimento estaria em-

butido nas suas conexões, de maneira a representá-lo como o mínimo conhecimento extraído de um especialista para uma dada solução de um problema. Portanto, caso se utilizasse um processo de otimização para esta RNA, por exemplo, com respeito ao número de neurônios na camada intermediária, não se poderia obter uma RNA final (vencedora) com o número de neurônios na camada intermediária com um valor menor do que a da inicial. Isto quer dizer que a RNA final estaria perdendo as informações extraídas do referido especialista através das regras iniciais. Entretanto, acredita-se que as regras iniciais elicitadas do especialista sejam apenas algumas informações que o especialista conseguiu articular para o engenheiro de conhecimento. Desta forma, ele estaria explicando o porquê dele inferir aquela decisão ou conclusão com respeito a um determinado domínio de um problema. Poderia ser, também, que estas regras já descrevam o mínimo conhecimento da tomada de decisão de uma dada conclusão por parte do especialista. Porém, não existe nada atualmente que comprove isso. Assim, neste trabalho se optou em usar, ao invés de regras básicas, a conotação de regras iniciais.

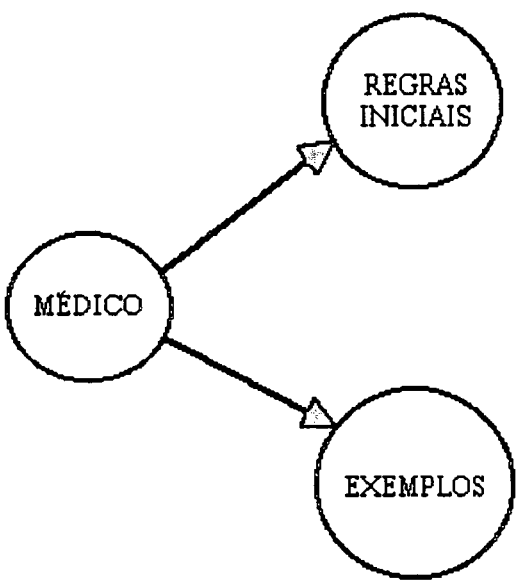

Figura 4.2: Relação especialista/engenheiro de conhecimento

As regras iniciais devem levar em consideração, ainda, as imprecisões associadas aos processos cognitivos humanos, tais como, o raciocínio e o pensamento. O modelo proposto para tratar as imprecisões é auxiliado pelos métodos de raciocínio *fuzzy*, os quais usam a lógica *fuzzy* para descrever as regras iniciais na forma de regras *fuzzy*

[76][77][192][195].

4.2 Tradução das Regras Iniciais Fuzzy em Grafos E/OU

As regras iniciais *fuzzy* são traduzidas em grafos E/OU, os quais definem a estrutura inicial da RNA do NNES. Em outras palavras, um grafo E/OU, o qual representa conceitos e relações, indica o número de neurônios nas camadas de entrada, da intermediária e de saída de uma RNA. O grafo também mostra a existência de conceitos intermediários e suas conexões, os quais são traduzidos na camada intermediária da RNA, de acordo com a Figura 4.3.

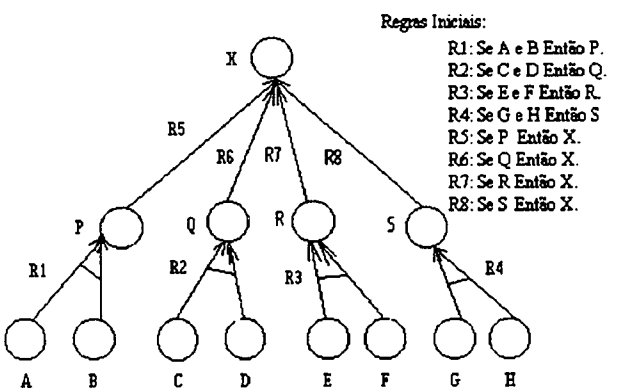

Figura 4.3: Organização da base de regras como uma RNA

Antes de se partir para a montagem, propriamente dita, dos grafos E/OU é necessário que se leve em consideração certas abordagens, com respeito ao formalismo das regras iniciais *fuzzy* extraídas de um especialista de domínio, conforme segue:

a) As regras iniciais estão formalizadas como cláusulas de Horn.

Caso as regras elicitadas do especialista de domínio não estejam no formato de claúsulas de Horn, elas podem ser reescritas, nesta forma, aplicando-se os seguintes equivalentes [64]:

- *Se* p *Então* q *E* r é substituído por *Se* p *Então* q e *Se* p *Então* r.

- *Se* p *OU* q *Então* r é substituído por *Se* p *Então* r e *Se* q *Então* r.
- *Se* p *E* (q *OU* r) *Então* s é substituído por *Se* p *E* q *Então* s e *Se* p *E* r *Então* s.

Assim, cada regra inicial *fuzzy* tem um antecendente consistindo de uma ou mais condições, bem como, um único conseqüente.

b) Nos grafos de inferências, os nós correspondem aos conceitos, os quais representam as *premissas* ou *conclusões* de uma regra. As regras iniciais *fuzzy* consistem de uma parte *Se* que indica o antecedente e expressa, no caso do nosso exemplo de aplicação[1], os sintomas de uma doença, enquanto a parte *Então* lida com o conseqüente e expressa os diagnósticos possíveis.

c) Além do que, cada uma dessas regras apresenta um grau de pertinência. Ele corresponde ao valor que será colocado nas entradas da RNA inicial, após o tratamento das variáveis semânticas [61][60][62][63][59][122][120]. Já as conexões ou ligações correspondem as *relações* entre dois nós. É nelas que estão embutidos os pesos da força de ligação entre os nós.

A seguir, é mostrado um caso de aplicação na área médica para classificar algumas doenças contagiosas infantis. Foram extraídas em torno de quatro a seis regras iniciais por doença. As Figuras 4.4, 4.5, 4.6 e 4.7, mostram os grafos E/OU utilizados para representar o formalismo destas regras iniciais *fuzzy* [184].

4.2.1 Exemplo

O exemplo[2] mostrado a seguir é direcionado para uma das áreas médicas, ou seja, para o diagnóstico médico de algumas doenças infantis. Para tanto se obteve do especialista em questão 41 sintomas e 4 diagnósticos. Após, formalizando-o em um NNES inicial, este compreenderá 41 neurônios (sintomas) na entrada da rede e 4 neurônios (diagnósticos) na saída da rede. As variáveis linguísticas, booleanas e numéricas, formalizadas para os sintomas das respectivas doenças infantis, são traduzidas com um certo grau de pertinência em função do que o médico conseguiu

[1]Como já citado anteriormente, o HES proposto neste trabalho tem como objetivo ser aplicado à Sistema de Apoio (ou Auxílio) ao Diagnóstico na Área Médica. No entanto, supõe-se que ele apresenta um certo grau de generalidade de forma a poder ser aplicado à problemas de outros domínios.

[2]O exemplo aqui apresentado não deve ser analisado do ponto de vista de precisão médica. Ele está sendo utilizado apenas como um exemplo didático.

articular durante o processo de EC. Além do que, os graus de pertinência que aparecem nos conseqüentes das regras relacionadas às doenças infantis, correspondem para todas as figuras apresentadas nesta Seção, como a saída do NNES, ou seja, o conjunto de padrões de saída do NNES.

Doença: Varicela

R1: *Se* febre alta *E* estável *E* erupção na pele com nevus *E* seguidas de bolhas *E* acnes *Então* A.

R2: *Se* dor no corpo forte *E* prurido *E* calafrios *Então* B.

R3: *Se* fraqueza no corpo *E* vômito *E* diarréia *E* cefaléia intensa *Então* C.

R4: *Se* A *Então* paciente com 0.95 de chance de ter contraído varicela.

R5: *Se* B *Então* paciente com 0.8 de chance de ter contraído varicela.

R6: *Se* C *Então* paciente com 0.7 de chance de ter contraído varicela.

A Figura 4.4 mostra o grafo de inferência para a doença infantil conhecida como varicela.

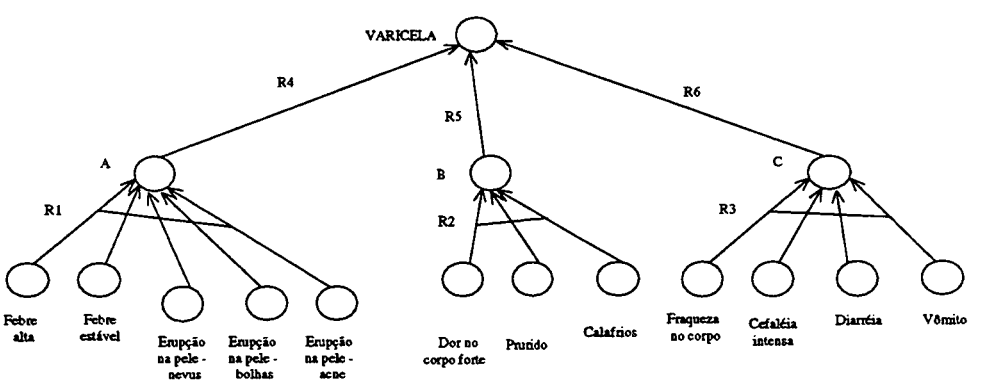

Figura 4.4: Grafo de inferência - varicela

Doença: Catapora

R7: *Se* febre leve *E* avermelhada na luz *E* na face *E* seguida pelo corpo todo (universal) *Então* D.

R8: *Se* aparecimento de glânglios cervicais *E* odinofagia *E* rigidez articular *E* cefaléia branda *E* corisa *Então* E.

R9: *Se* D *Então* a possibilidade do paciente ter contraído catapora é de 0.95.

R10: *Se* E *Então* a possibilidade do paciente ter contraído catapora é de 0.9.

A Figura 4.5, mostra o grafo de inferência para a doença infantil conhecida como catapora.

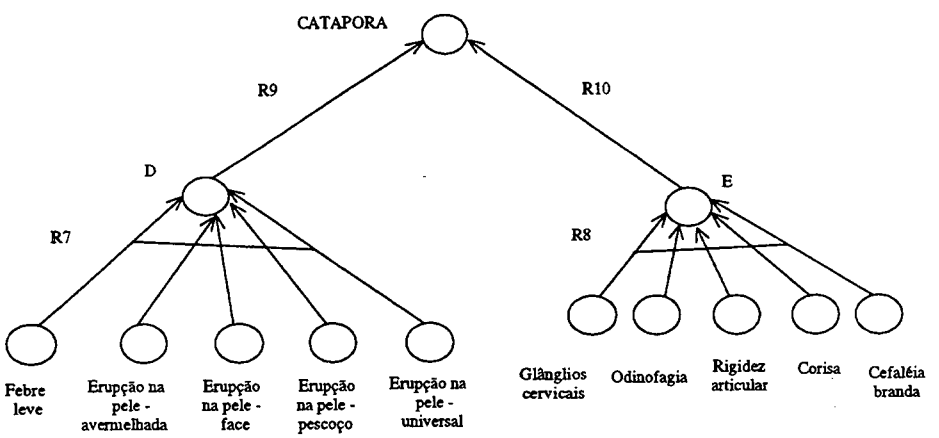

Figura 4.5: Grafo de inferência - catapora

Doença: Sarampo

R11: *Se* febre alta *E* em elevação *E* erupção na pele apresentando um colorido *pink*-castanho *E* iniciando no couro cabeludo *E* passando para pescoço *E* depois se expandido para o corpo todo *E* com pintas brancas nas bochechas *Então* F.

R12: *Se* há fotofobia *E* tosse *E* conjuntivite *E* corisa *E* olhos com pontos vermelhos *Então* G.

R13: *Se* F *Então* o paciente possui a chance de ter sarampo com 0.95.

R14: *Se* G *Então* o paciente possui a chance de ter sarampo com 0.80.

A Figura 4.6, mostra o grafo de inferência para a doença infantil conhecida como sarampo.

Doença: Caxumba

R15: *Se* febre moderada *E* apresenta glânglios cervicais *Então* H.

R16: *Se* salivação excessiva *E* dificuldade de chupar limão *E* boca seca *E* calafrios *Então* I.

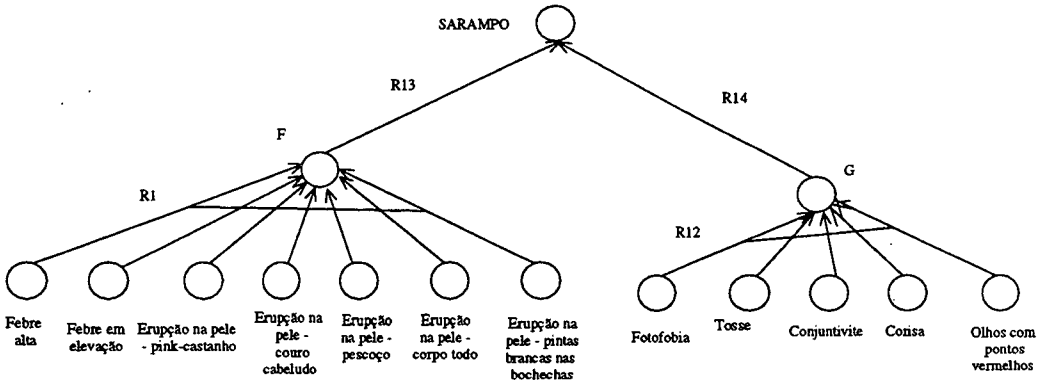

Figura 4.6: Grafo de inferência - sarampo

R17: *Se* H *Então* o paciente apresenta a chance de ter contraído caxumba com 0.9.

R18: *Se* I *Então* o paciente apresenta a chance de ter contraído caxumba com 0.85.

A Figura 4.7, mostra o grafo de inferência para a doença infantil conhecida como caxumba.

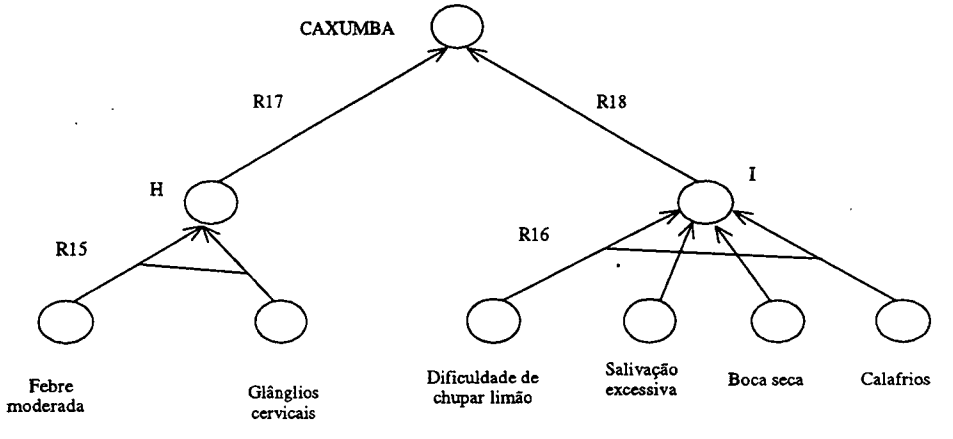

Figura 4.7: Grafo de inferência - caxumba

4.3 Tratamento das Variáveis de Entrada da RNA

Uma estrutura do NNES que tem condições para receber vários tipos de variáveis semânticas, isto é, entradas booleanas, lingüísticas e quantitativas (numéricas), é

considerada (Figura 4.8).

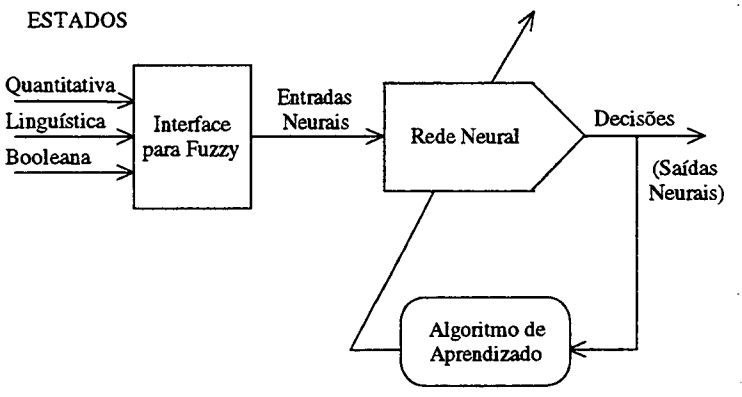

Figura 4.8: Tipos de variáveis de entrada

As entradas de dados da RNA podem ser tratadas de duas maneiras possíveis. Uma delas é subdividir o sistema neural em:

- Tudo que é variável lingüística é tratada por uma RNA *fuzzy*;
- Tudo que é variável booleana é tratada por outra RNA;
- Tudo que é variável numérica é tratada em uma terceira RNA.

Desta maneira, a saída de cada RNA é conectada a outra RNA que faz uma última classificação. Contudo, este tipo de estrutura, por ser particionada, apresenta algumas desvantagens, sendo que uma delas diz respeito ao algoritmo de mudança da estrutura. Isto é, este algoritmo teria que ser aplicado a cada uma destas redes particionadas, o que complicaria o desenvolvimento da implementação do algoritmo de aprendizado do NNES e se gastaria tempo de treinamento.

A segunda alternativa considera uma estrutura de NNES que tenha condições de permitir a presença, na entrada da rede, de diversos tipos de neurônios, sejam eles booleanos, lingüísticos ou quantitativos. Eles seriam tratados, internamente, como um único tipo de variável. Acredita-se que com este tipo de estrutura se consiga modelar uma estrutura de um modo mais simples possível e, também, de consumo de tempo de aprendizagem menor do que em uma estrutura tradicional clássica.

O modelo proposto representa o vetor de entrada em termos de variáveis lingüísticas,

booleanas e quantitativas (numéricas). Cada uma destas variáveis é traduzida pelo programa para uma única variável em comum, no caso associando-as a um grau de pertinência, isto é, no intervalo [0,1].

Sejam, por exemplo, os dados adquiridos do especialista médico na forma booleana[3]. Um paciente é questionado sobre se ele tem febre ou não. Caso afirmativo, ele assumirá o valor *fuzzy* 1 e, se negativo, o valor 0.

Outro exemplo é para o caso de dados linguísticos: o paciente é questionado sobre a intensidade da febre. Esta pode apresentar-se de três maneiras: febrícula, moderada e alta. Uma das maneiras usuais de se representar estes dados é através de números *fuzzy*, na forma triangular ou trapezoidal, por exemplo. Contudo, há o problema em que a solução obtida é uma função de pertinência e não apenas um número representando um grau de pertinência, pois o que se deseja são valores numéricos na entrada da RNA. Por isso optou-se por atribuir um valor *fuzzy*, isto é, um valor[4] entre [0,1] para cada uma das variáveis lingüísticas usadas no problema em questão. Assim, exemplificando, a variável linguística febrícula assume um valor de grau de pertinência com $\mu_F(x) = 0,2$, enquanto a moderada, com $\mu_M(x) = 0,6$ e a alta com $\mu_A(x) = 0,9$.

Suponha agora que a mesma variável (febre) usada anteriormente para representar um dado lingüístico seja extraída na forma quantitativa (numérica). A intensidade da febre se apresenta em torno de $38°c$. Comumente, a febre pode variar em uma faixa de um valor máximo ($42°c$) e um valor mínimo ($36,5°c$). Então, estes valores podem ser inferidos como 1, para o valor máximo e 0 para o valor mínimo. Assim, o valor correspondente a $38°c$ pode ser interpolado com o auxílio da equação da reta (temperatura da febre versus grau de pertinência), ou seja, utilizando o gráfico da Figura 4.9 obtem-se o correspondente grau de pertinência.

4.4 Modelo Matemático do Neurônio

A Figura 4.10, mostra o modelo matemático de um neurônio estático, onde:

[3]Na lógica *fuzzy*, a forma booleana é conhecida também como conjunto abrupto (*crisp*).

[4]Este valor poderia estar contido em outra gama de valores, por exemplo, [-1, 1], que é o caso da entrada bipolar.

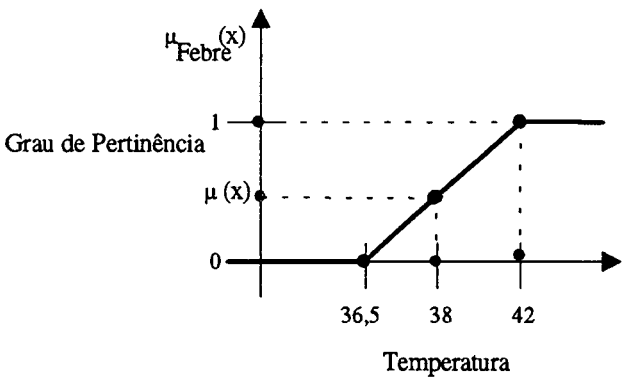

Figura 4.9: Exemplo de variável quantitativa

X(t) = Vetor de entrada do neurônio ou saídas dos neurônios excitando o neurônio considerado de n-dimensões

$$X(t) = [x_1(t), x_2(t), \cdots, x_i(t), \cdots, x_n(t)]^T \epsilon \Re^n \tag{4.1}$$

y(t) = Saída escalar do neurônio

$y(t) \epsilon \Re^1$

N = Função de mapeamento não-linear, $X \rightarrow Y, x(t) \mid\rightarrow y(t)$

Onde:

$$X : Z^+ \subset \Re^n \tag{4.2}$$

$$Y : Z^+ \subset \Re^1 \tag{4.3}$$

Este mapeamento pode ser notificado como *N*. Então:

$$y(t) = N[X(t) \epsilon \Re^n] \epsilon \Re^1 \tag{4.4}$$

Matematicamente, a função de mapeamento não-linear neural, *N*, é dividida em duas partes [33][34][19][21][31][22][24][23][20][32][30][28][27][26][25][29]:

- Confluência (©)
- Operação de ativação não-linear (Ψ)

4.4.1 Confluência

A operação de confluência © é o nome dado para uma operação geral, a qual tem como argumentos os pesos sinápticos e as entradas [76][77].

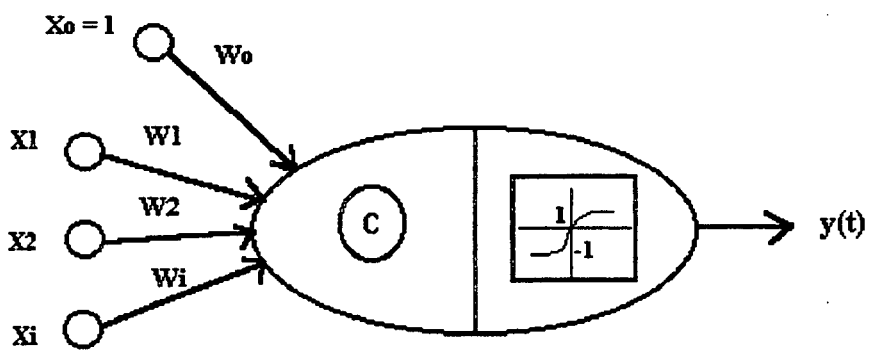

Figura 4.10: Modelo matemático de um neurônio estático

Assim, define-se os vetores de entradas neurais e pesos sinápticos como a seguir:

$$X_a(t) = [x_0(t), x_1(t), \cdots, x_i(t), \cdots, x_n(t)]^T \epsilon \Re^{n+1}, x_0(t) = 1 \tag{4.5}$$

$$W_a(t) = [w_0(t), w_1(t), \cdots, w_i(t), \cdots, w_n(t)]^T \epsilon \Re^{n+1} \tag{4.6}$$

Onde:

$x_0(t) = 1$ e $w_0(t)$ - introduzem o termo *bias* (limiar) na operação de confluência.
$X_a(t)$ = Vetor de entrada neural (nova informação)
$W_a(t)$ = Vetor de peso sináptico (base de conhecimento acumulada)

Há duas maneiras de se resolver a operação de confluência. Uma delas é pelo produto escalar dos vetores $X_a(t)$ e $W_a(t)$. A outra é pela distância Euclideana entre os vetores $X_a(t)$ e $W_a(t)$ [77]. Porém, os neurônios computacionais da maioria das RNA descritas na literatura assumem uma operação de confluência dada pelo produto escalar.

Assim, este mapeamento fornece uma saída escalar u(t), a qual representa uma medida de similaridade entre o vetor de entrada $X_a(t)$ e o conhecimento armazenado no vetor de peso sináptico $W_a(t)$.

Então:

$$u(t) = X_a(t) \copyright W_a(t) \tag{4.7}$$

Onde:

$$u(t) \epsilon \Re^1$$

O produto escalar de $X_a(t)$ e $W_a(t)$ é definido geometricamente, como a projeção das entradas neurais $X_a(t)$ (nova informação) em relação aos pesos sinápticos $W_a(t)$ (conhecimento acumulado), no espaço vetorial, como mostrado na Figura 4.11.

Então:

$$u(t) = W_a(t)^T X_a(t) = \sum_{i=1}^{n} W_i Xi \tag{4.8}$$

4.4.2 Função de Ativação Não-Linear

A função de ativação não-linear Ψ mapeia o valor de confluência $u(t) \epsilon$ $[-\infty, \infty]$ e, eventualmente, é um valor de ativação para uma saída neural limitada. Em geral, a saída neural se apresenta na faixa de [0, 1] para sinais unipolares e [-1, 1] para sinais bipolares. O operador de ativação não-linear transforma o sinal u(t) em uma saída neural limitada y(t), isto é:

$$y(t) = \Psi[u(t)] \tag{4.9}$$

$$y(t) = \Psi[X_a(t) \copyright W_a(t)] \tag{4.10}$$

Há várias maneiras de modelar a função de ativação não-linear. A adotada neste trabalho é a função tangente hiperbólica visto a necessidade de uma função bipolar:

$$\Psi[x(t)] = \frac{[e^{gx(t)} - e^{-gx(t)}]}{[e^{gx(t)} + e^{-gx(t)}]} = \tanh[x(t)] \tag{4.11}$$

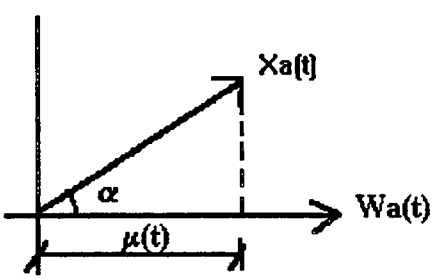

Figura 4.11: Medida de similaridade ($u(t) = X_a(t) \copyright W_a(t)$)

Onde:

g = parâmetro que controla a inclinação da função tangente hiperbólica (ganho de ativação).

Esta função torna-se uma sigmoidal unipolar para o intervalo [0,1], definida como:

$$\Psi[u(t)] = \frac{1}{[1 + e^{(-gu(t))}]}, 0 < g < 1 \tag{4.12}$$

Enquanto que, para uma sigmoidal bipolar é adotada a seguinte expressão matemática:

$$\Psi[u(t)] = \tanh[gu(t)], 0 < g < 1 \tag{4.13}$$

4.4.3 Mapeamento Matemático da Entrada/Saída do NNES

O mapeamento da entrada/saída do NNES, mostrado na Figura 4.12, pode ser representado matematicamente por:

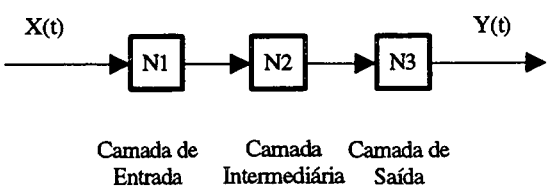

Figura 4.12: Diagrama de bloco das camadas da RNA

$$Y(t) = N_3[N_2[N_1[X(t) \epsilon \Re^n]]] \epsilon \Re^m \tag{4.14}$$

A Equação 4.14, em termos dos operadores de confluência e ativação não-linear, pode ser reescrita como:

$$Y(t) = \Psi_3[W_{3a}(t) © \Psi_2[W_{2a}(t) © \Psi_1[W_{1a}(t) © X_a(t)]]] \tag{4.15}$$

Onde:

Ψ_i = operador de ativação não-linear

© = operador de confluência

$W_{1a}(t), W_{2a}(t), W_{3a}(t)$ = vetores de pesos sinápticos para as camadas de entrada, intermediária e saída, respectivamente.

4.4.4 Modelagem fuzzy

Em uma arquitetura direta estática o neurônio responde, instantaneamente, às entradas *fuzzy*, tendo em vista a ausência de elementos dinâmicos na estrutura. As operações matemáticas, em uma rede direta, podem ser desempenhadas por aritmética *fuzzy* ou operações lógicas *fuzzy*. Então, a função de um neurônio não-*fuzzy* pode ser modelada como:

$$y(t) = \Psi\Big(\sum_{i=1}^{n} W_i X_i\Big) \tag{4.16}$$

Onde:

$[x_1, \cdots, x_n]$ = entradas neurais
$[w_1, \cdots, w_n]$ = pesos sinápticos
y(t) = saída neural
Ψ = função ativação não-linear

Da Equação 4.16, pode-se observar que a operação matemática envolvida em um neurônio computacional é: o produto escalar entre as entradas neurais e os pesos sinápticos de modo a se obter o somatório desses produtos.

Desta forma, o produto escalar na Equação 4.16 pode ser substituído por multiplicação *fuzzy* e a operação de somatório por adição *fuzzy*. Porém, outro método utilizado é através de outras operações lógicas *fuzzy*, tais como, Max/Min, referentes aos conectivos OU e E. Este último método é o adotado para o desenvolvimento matemático desta etapa. A seguir, ele é descrito com maiores detalhes.

Se expressarmos os sinais de entradas neurais em termos de suas funções de pertinência no intervalo [0,1], então, pode-se escrever o vetor de entradas neurais como:

$$X_a(t) = [X_0(t), X_1(t), \cdots, X_i(t), \cdots, X_n(t)]^T \epsilon\, [0,1]^{n+1} \tag{4.17}$$

Onde os sinais neurais (incluindo o termo *bias*, x_0), são limitados por $(n+1)$ do hipercubo dimensional $[0,1]^{n+1}$. Similarmente, o vetor de peso sináptico $W_a(t)$

é expresso através da unidade de hipercubo $[0,1]^{n+1}$. As operações matemáticas utilizadas englobam os operadores lógicos (conectivos), como E/OU, que são aplicados nesses sinais. Eles são solucionados por meio de uma das formas generalizadas dos conectivos para interseção e união de conjuntos *fuzzy*, ou seja, como normas triangulares e co-normas, as quais modelam os conectivos para os conjuntos *fuzzy*, entre t-normas e t-co-normas (ou s-normas). Além disso, são aplicados a estes sinais também o conectivo NÃO [195][193]. A seguir, é visto como cada um destes conectivos é tratado.

Sejam expressas as entradas x_1 e x_2 através de [0,1]. Então, é definido o conectivo E generalizado (operação T) como uma função de mapeamento T:

$$T : [0,1] \times [0,1] \rightarrow [0,1]$$

Dado por:

$$y_1 = [x_1 \; E \; x_2] \equiv [x_1 \; T \; x_2] = T[x_1, x_2] \tag{4.18}$$

Similarmente, define-se o conectivo OU generalizado (co-norma T) como a função de mapeamento S.

$$S : [0,1] \times [0,1] \rightarrow [0,1]$$

Dado por:

$$y_2 = [x_1 \; OU \; x_2] \equiv [x_1 \; S \; x_2] = S[x_1, x_2] \tag{4.19}$$

A negação N, em $x_1 \epsilon [0,1]$, é definida como um mapeamento onde:

$$N : [0,1] \rightarrow [0,1]$$

As propriedades referentes a este tipo de conectivo lógico são descritas a seguir:

$$y_3 = N[x_1] = 1 - x_1 \tag{4.20}$$

Então:

N(0) = 1
N(1) = 0
N(N(x)) = x

Ainda há algumas propriedades importantes dos operadores T e S, conforme segue:

T(0, 0) = 0;
T(1, x) = x;
T(1, 1) = 1;
T(x, y) = T(y, x)

S(0, 0) = 0;
S(0, x) = x;
S(1, 1) = 1;
S(x, y) = S(y, x)

Também os teoremas de De Morgan são expressos como a seguir:

$$T(x_1, x_2) = 1 - S(1 - x_1, 1 - x_2) \tag{4.21}$$

$$S(x_1, x_2) = 1 - T(1 - x_1, 1 - x_2) \tag{4.22}$$

Desta forma, no desenvolvimento da lógica *fuzzy*, baseada na morfologia neural, usam-se as seguintes operações sinápticas e do soma combinadas: seja o vetor de entradas e pesos sinápticos representados por $X_a(t) \epsilon [0,1]^{n+1}$ e $W_a(t) \epsilon [0,1]^{n+1}$, respetivamente.

Assim, na Equação 4.7, substituindo-se a operação-© pela operação-T e a operação-$\sum$ pela operação-S, obtém-se para os neurônios OU:

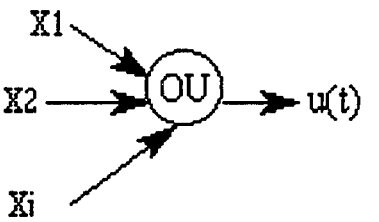

Figura 4.13: Neurônio *fuzzy* OU

$$u(t) = S_{i=1}^{n}[w_i(t)\ T\ x_i(t)]\ \epsilon\ [0,1] \tag{4.23}$$

Então, na Equação 4.7, substituindo-se a operação-© pela operação produto algébrico e a operação-$\sum$ pela operação-T, obtém-se para os neurônios E:

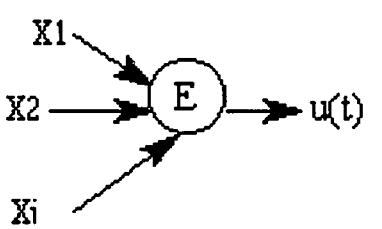

Figura 4.14: Neurônio *fuzzy* E

$$u(t) = T_{i=1}^{n}[w_i(t) * x_i(t)]\ \epsilon\ [0,1] \tag{4.24}$$

$$y(t) = \Psi[u(t)]\ \epsilon\ [0,1] \tag{4.25}$$

Ainda, caso se deseje usar a função de mapeamento não-linear no intervalo bipolar, ou seja, indicando ambas as interações excitatórias (positivas) e inibitórias (negativas) do vetor de entrada neural, deve-se considerar ambos $X_a(t)$ e seus valores negados, $N[X_a(t)]$, de modo que as entradas neurais sejam de dimensão (2n + 2). Então, faz-se a seguinte transformação:

Seja $x(t) \epsilon [0, 1]$ um sinal unipolar (função sigmóide). O correspondente sinal bipolar (função tangente hiperbólica), z(t), é definido como:

$$z(t) = 2x(t) - 1 \tag{4.26}$$

As operações T e S, definidas no intervalo [0,1], podem ser transformadas para o intervalo [-1,1] usando-se a Equação 4.26, enquanto que, para a negação, é definida como: $N[Z] = -Z(t)$.

4.5 NNES Inicial

Através dos grafos E/OU, obtidos na Seção 4.2, tem-se a idéia de como será a topologia da rede de forma a se saber a quantidade de neurônios necessária nas camadas de entrada, na intermediária e na saída da rede, bem como, as ligações entre os neurônios (nós) nas camadas de entrada e intermediária, e entre as camadas intermediárias e saída da rede. Além disso, através destes grafos de inferência, a camada intermediária é definida por nós E, enquanto a camada de saída da rede por nós OU. Desta forma, os operadores lógicos E/OU são incorporados no local do somatório de pesos nos neurônios da rede.

A rede a ser configurada consiste de uma camada de entrada, uma intermediária e uma de saída. A camada de entrada corresponderá aos atributos de dados obtidos da etapa de EC, do especialista de domínio em regras iniciais *fuzzy*, ou seja, a parte *Se* das regras. A camada intermediária[5] representará as hipóteses intermediárias,

[5]Um questionamento que pode ocorrer é se nós podemos omitir o nível intermediário da rede neural e conectarmos os atributos diretamente para o conseqüente (camada de saída). Esta abordagem causaria um problema devido as regras múltiplas, pois, algumas combinações dos atributos envolvidos em diferentes regras podem também ativar o conceito final. Então, para evitar estas possíveis combinações não pretendidas, cada premissa da regra é atribuída a uma unidade conjuntiva (neurônio E) que ativa o conceito final disjuntivamente (neurônio OU).

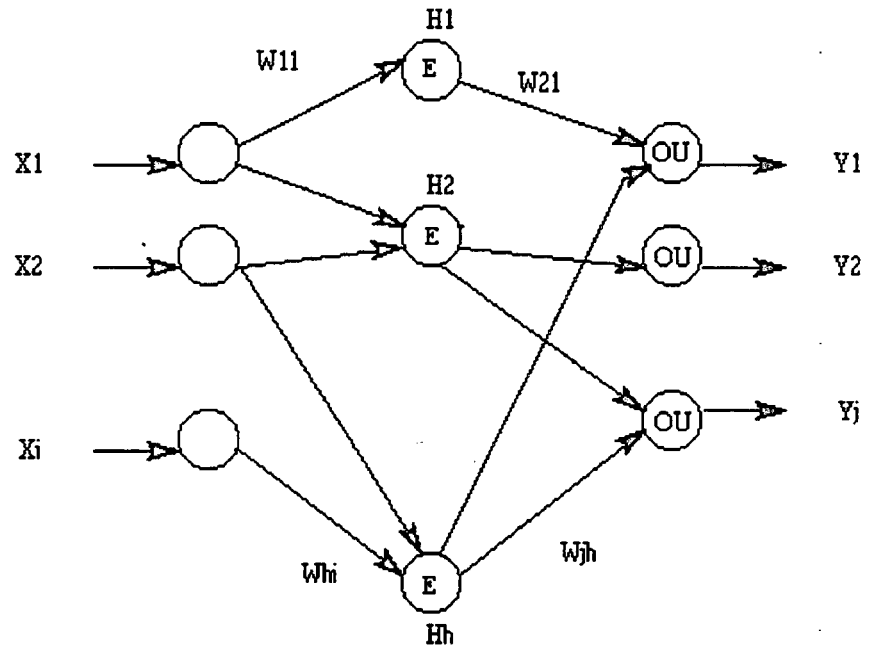

Figura 4.15: Configuração inicial da RNA

isto é, corresponderá as premissas das regras. Enquanto isso, a camada de saída, denotará as hipóteses finais, ou seja, serão as conclusões ou a parte *Então* das regras. A Figura 4.15, mostra a configuração inicial deste NNES, onde x_i representa as i entradas da rede, h_h denota os h nós intermediários e y_j representa as j saídas da rede. Enquanto isso, w_{hi} representa os pesos entre os i neurônios de entrada em relação aos h neurônios da camada intermediária, e w_{jh} indica os pesos entre os h neurônios da camada intermediária em relação aos j neurônios da camada de saída da rede neural.

4.6 RBES

RNA é considerada uma técnica poderosa e geral para o aprendizado de máquina. Entretanto, RNA possuem algumas dificuldades intrínsecas bem conhecidas. Talvez a mais significante delas seja a que está relacionada, essencialmente, com o fato delas serem *caixas-preta*. Isto é, determinar exatamente porque uma RNA *toma* uma decisão particular é uma tarefa árdua. Além disso, para muitas tarefas de aprendizado, é importante apresentar classificadores que não são somente altamente precisos mas, também, facilmente compreendidos pelos humanos. RNA são limitadas neste aspecto, de forma que são usualmente difícieis de serem interpretadas após o seu

treinamento.

Em contraste com as RNA, as soluções formadas por sistemas de aprendizado simbólico são usualmente muito mais acessíveis para a compreensão humana. Uma das maneiras usuais de representação de conhecimento é através de regras de produção para o desenvolvimento da base de conhecimento de um sistema especialista simbólico. Isto se deve ao fato que regras são mais fáceis para serem aceitas e memorizadas por usuários humanos. Analogamente, uma maneira para compreender as representações formadas por RNA consiste em extrair regras simbólicas de RNA treinadas.

A extração de regras diz respeito ao processo de derivar uma descrição de uma RNA treinada. Idealmente, o processo de extração de regras resulta em uma descrição simbólica de modo que tenta simular a conduta da rede em uma forma concisa e compreensiva.

Após o NNES inicial ser obtido, o conjunto de exemplos serve para validar a estrutura da RNA. No entanto, no pior das hipóteses, a rede não representa o conhecimento do problema, tornando-se evidente que as regras iniciais extraídas do especialista não são suficientes, como o esperado. Assim, estes mesmos exemplos são usados pelo algoritmo de aprendizado para refinar a rede. Este algoritmo pode mudar, ou seja, gerar e/ou eliminar conexões ou, pode ainda, gerar e/ou eliminar neurônios na camada intermediária da RNA. Após o refinamento da rede, uma nova discussão é feita com o especialista de domínio para validar as modificações na estrutura inicial da rede. Então, um novo conjunto de exemplos é obtido para testar mais uma vez a rede. No caso que ela tenha desempenhado bem, supõe-se que a rede atinge a meta proposta.

Assim, após o NNES ser refinado, um processo reverso é seguido na direção de inferir as regras *Se/Então*, juntamente com seus graus de pertinências. Então, um RBES é implementado. Desta forma, o RBES serve para explicar porque o NNES alcançou uma dada conclusão.

4.6.1 Descrição do Sistema Simbólico

Neste tipo de SE, ou seja, o RBES, é composto por dois módulos. Um deles corresponde ao mapeamento dos dados obtidos na saída do NNES em um grafo E/OU, de tal forma que este facilita a compreensão destes dados para a implementação deste sistema. O outro é o justificador, onde se encontra a técnica de extração de

regras escolhida. Esta técnica tem a finalidade de dar a explicação da resposta obtida na saída do sistema conexionista, no qual está embutido o algoritmo de extração de regras *fuzzy*, isto é, o *Fuzzy Rule Extraction Algorithm* (FUZZYRULEXT).

4.6.2 Mapeamento em Grafos E/OU

Na saída do NNES tem-se um arquivo dos conjuntos de dados de entrada/saída do sistema, bem como um outro referente aos dos valores das conexões (pesos) após o treinamento e refinamento do NNES. De posse destes, é possível mapear estes dados em um grafo, que neste caso será o do tipo E/OU, já que os dados adquiridos de um dado especialista foi na forma de regras *fuzzy*. Os grafos E/OU servirão como um processo auxiliar na montagem das regras antes delas serem implementadas em um sistema simbólico. Similar ao que foi realizado no mapeamento do NNES.

4.6.3 Justificador

Nesta fase o usuário pode questionar o sistema porque ele inferiu uma conclusão particular. O sistema responde com uma regra *Se/Então*, aplicável para o caso em questão, em termos das características das entradas. Note-se que estas regras *Se/Então* não estão representadas explicitamente na base de conhecimento codificada (RNA treinada). Elas são geradas pelo *sistema de inferência* dos pesos das conexões como e quando necessários para a explanação.

Esta etapa é ainda subdividida em outras duas: geração do caminho por *Backtracking* e a geração das cláusulas.

Com respeito à fase de geração do caminho por *Backtracking*, este consiste em dar uma justificação sobre uma conclusão olhando uma dada saída, de tal forma que o processo continua em um modo *Top-Down* até a camada de entrada ser alcançada. Veja o Capítulo 6 para maiores detalhes.

Na fase de geração das cláusulas, o processo inicia na seleção de um dado neurônio na camada de entrada para a geração das cláusulas e passando pela(s) camada(s) intermediária(s), correspondente a parte do antecedente de uma regra *Se/Então*, até a da saída, quando se obtem o conseqüente de uma regra *Se/Então*. Todo um formalismo matemático e heurístico é usado na determinação desta etapa. No Capítulo 6 estão descritas com detalhes todas estas abordagens.

Nos próximos Capítulos serão também descritos e discutidos os algoritmos propostos, tanto para o treinamento quanto para a extração de regras para o NNES.

Capítulo 5

Algoritmo de Aprendizado (GENBACK)

Desde muitos anos atrás que o interesse por RNA levou vários pesquisadores a desenvolverem não só um modelo matemático para representar o neurônio biológico, mas também regras de aprendizado que simulem o processo de aprendizado de um ser humano (reconhecimento de imagens, operações matemáticas, etc.). No primeiro caso, pode-se mencionar o modelo de McCulloch-Pitts (1943) e o Perceptron (1957) de Rosemblatt, enquanto, para o último, o aprendizado Hebbiano (1949) de Hebb, a regra delta ou Algoritmo do Erro Médio Quadrático (1962) de Widrow-Hoff, a regra delta generalizada (1986) de Rumelhart et al. e outros [159][196][166].

Como pode ser observado acima, exceto Rumelhart et al.[1], todos os outros autores desenvolveram suas regras de aprendizado para redes com apenas uma camada de entrada e outra de saída. Já para o caso das RNA de múltiplas camadas vários algoritmos de treinamento foram desenvolvidos. Uma das contribuições foi dada pelo algoritmo de retropropagação que usa a generalização da regra delta para o aprendizado de RNA multicamadas. Uma outra contribuição dada pelas RNA multicamadas foi a solução do problema da classificação de padrões não-linearmente separáveis. Além disso, depois que surgiram os algoritmos multicamadas foi possível provar que a quantidade de camadas intermediárias que uma rede deve ter, isto é, se uma RNA, por exemplo, com mais de três camadas executando uma determinada tarefa, uma rede com apenas três camadas pode fazer o mesmo, como é mostrado em [196]. Um problema que ainda não obteve uma solução plausível diz respeito

[1] Lembrar que P. Werbos foi o primeiro a desenvolver o algoritmo de retropropagação, seguido por D. Parker e somente alguns anos após que o Grupo de D. Rumelhart foram reconhecidos como os autores deste tipo de algoritmo de treinamento para RNA diretas com múltiplas camadas.

ao número de neurônios que uma camada intermediária deve possuir para executar uma certa tarefa. Existem, sim, heurísticas para se determinar este número, como é o caso de Eberhart que diz: "*O número de neurônios da camada intermediária é igual a raiz quadrada da quantidade de neurônios na camada de entrada somado com o número de neurônios da camada de saída de uma dada RNA*" [54]. Usando-se esta heurística ou outra, às vezes funciona para certos tipos de problemas.

O objetivo principal nesta etapa deste trabalho é desenvolver um algoritmo de aprendizado para uma RNA direta estática com múltiplas camadas, de modo a levar em consideração a problemática da quantidade de neurônios na camada intermediária. Esta dificuldade será abordada nas próximas seções.

5.1 GENBACK

O algoritmo de aprendizado proposto, chamado *Genetic-Backpropagation Based Learning Algorithm* (GENBACK), desenvolvido para o NNES é inspirado no algoritmo de retropropagação clássico, ou seja, o algoritmo desenvolvido por D.E. Rumelhart et al. [159]. A forma como os neurônios (nós) da rede neural estão conectados entre si baseia-se em uma topologia direta. Contudo, o algoritmo de aprendizado proposto neste trabalho é diferente em vários aspectos, quais sejam:

- Otimização da camada intermediária auxiliada por AG;
- Incorporação de conectivos lógicos E/OU no local do somatório de pesos;
- Tratamento das variáveis de entrada por lógica *fuzzy*.

5.1.1 Etapas do GENBACK

Resumindo, a primeira etapa na implementação do algoritmo de uma RNA já existe. Isto é, o número de neurônios na camada de entrada e na de saída é determinado por um dado conjunto de padrões. Desta forma, a estrutura da rede é definida com respeito aos neurônios serem *fuzzy* ou abruptos. Enquanto isso, o número de neurônios na camada intermediária é inferido das regras iniciais extraídas do processo de AC. Assim, o número de regras iniciais elicitadas corresponde ao número de neurônios na camada intermediária. Na próxima etapa, o algoritmo de aprendizado usa o conjunto de exemplos de modo a modificar não somente o peso de cada conexão, mas também a estrutura da rede. No último caso, as conexões são incluídas ou excluídas entre os neurônios, bem como os neurônios podem ser incluídos ou excluídos da camada intermediária pela aplicação do AG [92][56][20][30][28].

5.1.2 Algoritmo Genético

Os AG são baseados no trabalho de Holland [92], que se inspirou na evolução de uma população sujeita à reprodução, mutação e cruzamento (*crossover*) em um ambiente seletivo. Quando o AG é implementado este é executado, usualmente, de modo que envolva o seguinte ciclo: avaliação da aptidão de todos os indivíduos na população; criação de uma nova população através de operações tais como cruzamento, aptidão proporcional à reprodução e mutação nos indivíduos, cuja aptidão tenha já sido medida; descarte da população velha ou parte dela, e repetição usando a população nova juntamente com o que restou da população velha. Uma iteração deste laço é chamada uma geração.

Na Figura 5.1 é ilustrado um exemplo do código genético usado para codificar uma cadeia de cromossomos. A cadeia de cromossomos é formada pelo código binário cujo comprimento é igual a oito (8) *bits*. Esta codificação é a usada no desenvolvimento do GENBACK.

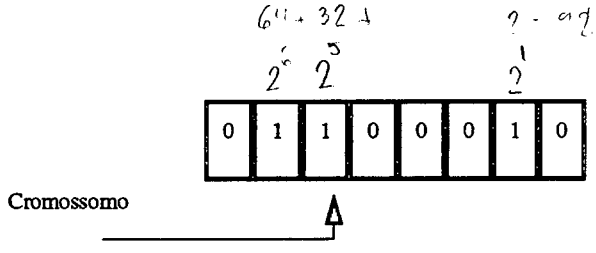

Figura 5.1: Exemplo de código genético do número decimal 98 para codificar uma cadeia de cromossomos

Quanto à função de aptidão que descreva o problema em questão, neste trabalho, tem-se duas. Uma delas é definida como o inverso entre a soma do erro obtido durante o processo de treinamento da RNA pelo algoritmo de retropropagação e o número de neurônios da camada intermediária. A outra, corresponde ao quociente entre o número de neurônios da camada intermediária e o erro obtido durante o processo de treinamento da RNA pelo algoritmo de retropropagação.

A criação de soluções novas é realizada como segue: primeiro, a reprodução é aplicada aos pais existentes. Um método comum para realizar isto é conhecido como a seleção de pais pelo processo da Roleta Ponderada [19][56][52]. A reprodução é

executada até que o tamanho da população da próxima geração seja completada. A população resultante de uma seqüência de caracteres é, então, modificada de acordo com os OG. A seguir, AG usam dois operadores básicos para manipular a composição genética de uma população: mutação de *bit* e cruzamento de 1-partição (ou ponto). Por fim, as novas soluções são geradas através da reprodução, mutação e cruzamento. Este processo é repetido até que o conhecimento da solução do ótimo global seja descoberto, ou até que o tempo disponível do computador seja esgotado.

Seguindo esta idéia, o AG foi escolhido para otimizar o tamanho da camada intermediária de uma dada RNA, o que pode ser justificado pelos seguintes fatos: evitar o ótimo local, experimentar soluções de otimização próximas ao ótimo global e ser de fácil implementação. Entretanto, quando se deseja aplicar o AG nesta direção, é necessário que se tenha alguns cuidados. Um deles, refere-se a quantidade de neurônios mínimo e máximo a serem colocados na camada intermediária. De fato, muitos neurônios geralmente têm como efeito reduzir a capacidade de generalização da rede. Isto implica em uma longa fase de aprendizado. Por outro lado, com pouquíssimos neurônios, a rede pode não aprender a tarefa com a precisão desejada. Assim, há um número de neurônios na camada intermediária que deve ser considerado para evitar-se os problemas acima mencionados [33][34][19][21][31][22][24][23][20][32][30][28][27][26][25][29].

5.1.3 Problemas da Derivabilidade das Funções E/OU

Antes de se abordar o problema da derivabilidade das funções E/OU, sugere-se primeiramente que sejam relembrados alguns conceitos matemáticos relacionados à derivação e à diferenciação de funções. Para tanto, veja a literatura correspondente a estes assuntos, por exemplo, em W.A. Maurer [127] e L. Leitthold [114].

Um segundo ponto, antes de ser abordado diretamente o problema da derivabilidade das funções E/OU, diz respeito a utilização do algoritmo de retropropagação no treinamento de RNA diretas *fuzzy*.

Este tipo de algoritmo tem como condição principal a exigência que as funções de ativação sejam não-lineares e deriváveis. Isto pode ser explicado porque se está minimizando o erro na saída da rede através da pesquisa por gradiente descendente, isto é, o gradiente de $f(x_1, \cdots, x_n)$ em qualquer ponto x é o vetor cujos componentes são as derivadas parciais da função f em um ponto $x = (x_1, \cdots, x_n)$ com respeito a essas variáveis, ou seja,

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \tag{5.1}$$

Pelo método do gradiente, partindo-se de um ponto arbitrário x_0, pode-se caminhar pela superfície $\nabla f(x)$ em direção do ponto de mínimo, bastando para isto evoluir sempre no sentido oposto ao do gradiente naquele ponto. Isto pode ser traduzido no algoritmo conhecido por retropropagação. Maiores detalhes sobre este algoritmo podem ser visto no Capítulo 2, Seção 2.2.5 deste trabalho.

Entretanto, em muitas aplicações as funções E/OU são interessantes de serem utilizadas em neurônios artificiais, ao invés das funções tradicionais. Contudo, a partir do momento que elas são usadas, defronta-se com um problema: o da derivabilidade destas funções. Caso se queira usar para treinamento de RNA algoritmos como o de retropropagação, isto implica que não se pode, no sentido extritamente matemático, aplicar o algoritmo de retropropagação no aprendizado de RNA diretas com neurônios E/OU ou qualquer outro algoritmo que necessite que as funções envolvidas sejam deriváveis.

Um terceiro ponto refere-se as várias abordagens de HES vistas no Capítulo 1 deste trabalho, principalmente X. Zhang et al., que considera a conceituação de derivabilidade de uma função como sendo o mesmo que diferenciabilidade. Como foi visto anteriormente, elas são bem diferentes no sentido da conceituação matemática. Não será o caso ao longo deste trabalho.

Finalmente chegamos ao problema crucial da derivabilidade das funções E/OU. Do ponto de vista matemático ela não existe para este tipo de funções. Contudo, existe uma série de autores, como visto no Capítulo 1, na descrição dos HES, que estão inclusive publicando em revistas altamente conceituadas, como é o caso da *IEEE Transactions on Neural Networks*[2] e o da Neural Networks[3] (ambas consideradas como as melhores do mundo a nível americano e europeu, respectiva-

[2]Nesta revista consta como Editor Chefe o Professor Jacek M. Zurada e o corpo editoral é constituído de pesquisadores tais como: Robert J. Marks II, J. Ghosh, M.H. Hassoun, J. Mao et al.

[3]Nesta revista consta como Editores Chefes o Professor Stephen Grossberg, o Professor Mitsuo Kawato e o Professor John Taylor, e o corpo editoral é composto por Bart Kosko, Lofti Zadeh, Richard Lippmann et al.

mente), que têm publicado artigos abordando o problema da derivabilidade das funções E/OU. Nestes artigos trata-se de contornar este problema de diferentes formas, levando-se em consideração vários formalismos matemáticos como visto em [64][175][120][77][135][150][57][85][80][98][138]. Contudo, concorda-se que matematicamente isto não é possível devido a estas funções apresentarem pontos de descontinuidades (pontos angulosos), ou seja, quando não se verificam simultaneamente as igualdades

$$\lim_{x \to x_0^+} f(x) = \lim_{x \to x_0^-} f(x) = f(x_0) \tag{5.2}$$

Além disso, por elas possuirem valores impróprios em determinados pontos, a razão incremental $f(x_0 + h) - f(x_0)/h$ tende a ∞ para h tendendo a zero, ou seja, não existe propriadamente derivada no ponto, por exemplo, x_0. Então:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h} = \infty \tag{5.3}$$

Um exemplo disso é o *impulso*. A função admite, neste caso, uma derivada imprópria em x_0.

No entanto, como em muitas outras coisas em engenharia e, talvés seja este o espírito desses artigos, na realidade não se pretende encontrar uma *derivada* onde, matematicamente, ela não existe. Mas sim, contornar este problema de forma que o algoritmo tipo o de retropropagação possa, eventualmente, ser aplicado para este tipo de funções e, também eventualmente, adaptarem a rede para encontrar pesos que resolvam, de forma razoavelmente boa, o problema.

Além disso, considerando o nível da revista *IEEE Transactions on Neural Networks*, bem como do editor chefe e referees, nós entendemos que muito provavelmente eles também consideraram a situação deste ponto de vista e não do ponto de vista estritamente matemático. Caso contrário, por exemplo, os trabalho de X. Zhang et al. e S. Mitra et al. não poderiam ter sido aceitos para publicação nesta revista.

Como último ponto, gostaríamos de discutir sobre a convergência das RNA *fuzzy*. Em uma RNA com neurônios E/OU a operação de confluência, normalmente utilizada, é a dos conectivos E/OU usando as funções de Min/Max para solucioná-los. Para estes tipos de funções se leva em conta achar os pontos de máximo ou de mínimo. O que ocorre é que se a função é contínua, verifica-se que uma condição

necessária, mas não suficiente, é que possua um máximo ou mínimo em um determinado ponto, onde a derivada seja nula. No caso das funções máximo e mínimo, ocorre a existência de pontos angulosos, ou seja, descontínuos, implicando que não haja a derivabilidade das referidas funções. Desta forma ao se aplicar o algoritmo de retropropagação no treinamento de uma RNA *fuzzy*, pode acontecer dois fatos:

- Se, no momento do treinamento, este algoritmo não encontrou pontos angulosos, dependendo do intervalo que esteja sendo aplicado, pode-se dizer que a rede tem possibilidades de convergir;
- Caso se encontre pontos angulosos, ou seja, haja pontos de descontinuidades, ocorre que neste ponto não há a derivada. Então, esta RNA tem fortes possibilidades de não convergir.

Assim, uma das alternativas é a de trocar o algoritmo de retropropagação por um outro algoritmo que não exija a derivabilidade de uma função, como é o caso, na disciplina de Programação Não-Linear[4] do método conhecido como *Pesquisa Univariável*, que não utiliza o cálculo da derivada de uma função, mas sim, acha um ponto ótimo de uma dada função, ou seja, uma solução *ótima* para um dado problema, dentro de um espaço de busca pré-determinado pelo especialista juntamente com o engenheiro de conhecimento. A seguir é dado um exemplo de um Pseudo-Código do Algoritmo de Pesquisa Univariável.

% Pseudo-Código %
Inicialização das variáveis;
Divisão Tricotômica;
Intervalo de criação de pesos aleatórios;
Escolha da parcela de trabalho, ou seja, espaço amostral;
Criação de um vetor de pesos;
Determinação do peso total;
Para $I = 1$ até peso total:

Escolha do primeiro peso da fila que será alterado para uma faixa de valores e os demais serão fixados;
Passo Para Frente;
Cálculo do erro na saída da RNA: $E = (saida\_desejada - saida\_calculada)^2$;

[4]Lembrar que os algoritmos que utilizam a pesquisa de gradiente descendente para a minimização de um determinado erro fazem parte também da Programação Não-Linear. Estes algoritmos são classificados pela Programação Não-Linear como aqueles que necessitam determinar a derivada de uma função, como é também o caso do método de Newton ou quase-Newton.

Fim Para

Escolha do menor dos pesos para fixar na fila e passar para o próximo peso na seqüência. Por exemplo, o método bolha de ordenação de variáveis;

$Peso\_Atual = Peso\_Escolhido$;

Término do processo em um número x de iterações ou quando chegar a um erro mínimo pré-determinado.

Um outro método que aqui pode ser citado é chamado Método Direto. Conceitualmente o tipo mais simples para o procedimento deste método é mudar uma variável em um tempo enquanto são mantidos todos os outros constantes até que o mínimo seja alcançado [89]. Por exemplo, um dos procedimentos seria fixar uma das variáveis, digo x_1, constante e variar x_2 até um mínimo ser obtido. Então, mantendo o valor novo de x_2 constante, mudar x_1 até um ótimo para o valor de x_2 ser alcançado e, assim por diante. A Figura 5.2 apresenta um diagrama de fluxo de informação simplificado deste método. Outros métodos utilizados dentro da Programação Não-Linear, além deste, podem ser: Pesquisa do Poliedro Flexível, Método de Powell, Método de Pesquisa Randômica, Algoritmo de Subgradiente e Linearização Externa de $L(u)$. Para mais informações e escolha de outros métodos, veja as referências [124][89][126].

Concluindo, por ser o algoritmo de retropropagação extremamente interessante e simples de implementar, seria o caso de se encontrar uma alternativa para ele. As próximas seções tratam disso. Contudo, a procura dessa alternativa não significa encontrar uma derivada onde ela não pode existir, pois isto não encontra apoio matemático no que diz respeito a derivabilidade de uma função. O que se está tentando realizar aqui é apenas contornar este problema.

5.1.4 Regra Delta para RNA E/OU

Devido a dificuldade na análise de operações do tipo Min e/ou Max, o treinamento de RNA *fuzzy*, especialmente RNA Min/Max, parece não ser abordada rigorosa e sistematicamente. Portanto, na prática, tende-se a escolher a operação de multiplicação e *bounded-addition* para substituir as operações de Min/Max, de modo a contornar esta dificuldade. A despeito do fato que a RNA modificada é treinável para sua natureza analítica, ela é funcionalmente muito diferente da rede original.

Em [194], os autores fizeram uma outra tentativa para desenvolver uma teoria rigorosa para a derivabilidade de funções Min/Max por meio de análise funcio-

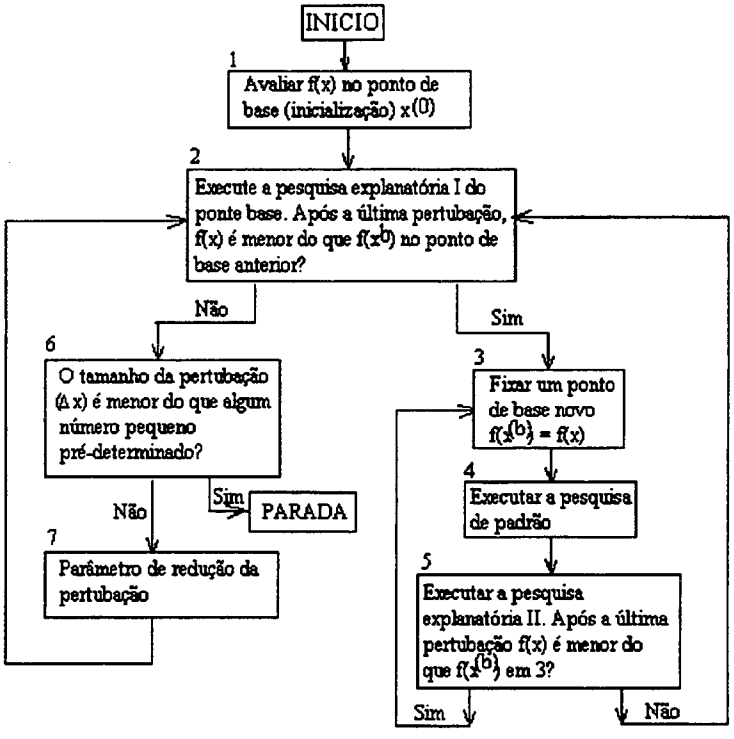

Figura 5.2: Diagrama de fluxo de informação para minimização da pesquisa univariável

nal. Além disso, eles derivaram a regra delta para treinamento de RNA Min/Max (ou RNA *fuzzy* ou RNA E/OU) baseadas na teoria de derivabilidade de funções Min/Max.

Primeiramente, antes de iniciarmos o desenvolvimento matemático para esta etapa, vamos definir, baseado em [194], a função $Lor(x)$ pertecente ao conjunto dos números reais $\Re$, como:

$$Lor(x) = \begin{cases} 1 & \text{if } x > 0 \\ 1/2 & \text{if } x = 0 \\ 0 & \text{if } x < 0 \end{cases}$$

E, de acordo com um dos teoremas dado em [194], supõe-se que $f(x)$, $g(x)$, $h_1(x) = f(x)\ S\ g(x)$ e $h_2(x) = f(x)\ T\ g(x)$ são funções reais. Se todas elas são deriváveis no ponto x, então:

$$\frac{\partial h_1(x)}{\partial x} = \frac{\partial [f(x)\, S\, g(x)]}{\partial x} \tag{5.4}$$

$$\frac{\partial h_1(x)}{\partial x} = Lor[f(x) - g(x)]\frac{\partial f(x)}{\partial x} + Lor[g(x) - f(x)]\frac{\partial g(x)}{\partial x} \tag{5.5}$$

$$\frac{\partial h_2(x)}{\partial x} = \frac{\partial [(x)\, T\, g(x)]}{\partial x} \tag{5.6}$$

$$\frac{\partial h_2(x)}{\partial x} = Lor[g(x) - f(x)]\frac{\partial f(x)}{\partial x} + Lor[f(x) - g(x)]\frac{\partial g(x)}{\partial x} \tag{5.7}$$

Desta forma, as Equações 5.5 e 5.7 servem para auxiliar no cálculo da derivada parcial da minimização do erro médio quadrático na saída da RNA em relação a um conjunto de treinamento, denominado de *Função Custo J*. A seguir será demonstrado matematicamente o desenvolvimento do GENBACK.

5.1.5 Algoritmo de Treinamento - GENBACK

Suponha-se que temos uma RNA clássica com uma camada de entrada, uma de saída e N intermediárias e a função ativação $\Psi(.)$. Usando-se $w_{(n)ji}$ para denotar o peso entre o nó i na camada n e o nó j na camada $n+1$, usando-se T_s e Y_s para denotar a saída desejada e a saída do nó s, respectivamente, na camada de saída $n+1$ e usando-se $U_{(n)i}$ e $Y_{(n)i}$ para denotar a ativação de entrada e a saída do i na camada $n(n = 0, 1, \cdots, n+1)$, tem-se, geralmente, as seguintes fórmulas interligando $U_{(n)i}$ e $Y_{(n)i}$:

$$U_{(n)i} = \sum_k w_{(n-1)ki} Y_{(n-1)k} \tag{5.8}$$

$$Y_{(n)i} = \Psi(U_{(n)i}) \tag{5.9}$$

Onde k é usado para todos os nós na camada $n-1(n = 1, 2, \cdots, n+1)$. Define-se, também, a função custo J como segue:

$$J = \frac{1}{2}\sum_{t=1}^{T}\sum_s (T_s - Y_s)^2 \tag{5.10}$$

Onde s é usado para todos os nós da camada de saída $n+1$ e T é o número de amostras de dados. Agora, suponha-se que $T = 1$, então, através da derivabilidade da função custo J, pode-se chegar no algoritmo de aprendizado utilizando a regra delta para a RNA, como segue:

$$\delta_{(n+1)s} = (T_s - Y_s)\Psi'[U_{(n+1)s}] \tag{5.11}$$

$$\delta_{(n)i} = \Psi'(U_{(n)i}) \sum_j w_{(n)ji}\delta_{(n+1)j} \tag{5.12}$$

Então, a regra delta é derivada como:

$$\Delta w_{(n)ji} = -\eta(\frac{\partial J}{\partial w_{(n)ji}}) = -\eta\delta_{(n+1)j}Y_{(n)i} \tag{5.13}$$

onde η é o parâmetro conhecido como taxa de aprendizado.

Agora a tarefa principal é treinar uma RNA E/OU de modo que ela possa acomodar pares de entrada e saída desejada da rede para uma dada precisão. Para este objetivo, usou-se a idéia convencional de gradiente descendente para designar uma regra delta para minimizar J. Para um p, de acordo com os teoremas provados em [194], todas as derivadas parciais de J com respeito a $w_{(n)ji}$, existem em quase todo $\Re$. Desta forma, tem-se as seguintes representações[5] [22]:

Passo Para Frente

Propagação das ativações das unidades da Camada de Entrada (l) para as das unidades da Camada Intermediária (i)

$$U_{(n)i} = T_l[w_{(n-1)il} * Y_{(n-1)l}] \in [-1, 1] \tag{5.14}$$

Propagação das ativações das unidades da Camada Intermediária (i) para as das unidades da Camada de Saída (j)

[5]Este procedimento matemático não garante a convergência da função custo J, visto que se existirem pontos angulosos, ou seja, a função neste ponto não apresenta derivada, o algoritmo de treinamento para a RNA especificada neste trabalho pode ficar comprometido.

$$U_{(n+1)j} = S_i[w_{(n)ji}\, T\, Y_{(n)i}] \in [-1, 1] \tag{5.15}$$

A saída neural é definida como:

$$Y_{(n+1)j} = \Psi[U_{(n+1)j}] \in [-1, 1] \tag{5.16}$$

$$\Psi[U_{(n+1)j}] = \tanh[U_{(n+1)j}] \tag{5.17}$$

Passo Para Trás

As regras de aprendizado para modificar w_{jin} podem ser desenvolvidas como segue [22][75][76][77][194]:

Camada Intermediária (i)/Camada de Saída (j)

Para um caso específico, isto é, o caso de três camadas, onde a camada intermediária tem neurônios E e a camada de saída tem neurônios OU, estes são auxiliados por uma função ativação do tipo sigmóide bipolar e β é o parâmetro conhecido como momento, então:

Função ativação:

$$\Psi[U_{(n+1)j}] = \tanh[U_{(n+1)j}] \tag{5.18}$$

Derivada da função ativação:

$$\Psi'[U_{(n+1)j}] = 4\frac{e^{-2U_{(n+1)j}}}{[1 + e^{-2U_{(n+1)j}}]^2} \tag{5.19}$$

Então, a regra delta para RNA E/OU é dada para as derivadas parciais de J com respeito aos pesos $w_{(n+1)ji}$ como segue:

$$\Delta w_{(n+1)ji} = -\eta(\frac{\partial J}{\partial w_{(n+1)ji}}) \tag{5.20}$$

$$\frac{\partial J}{\partial w_{(n+1)ji}} = (\frac{\partial J}{\partial U_{(n+1)j}}) \cdot (\frac{\partial U_{(n+1)j}}{\partial w_{(n+1)ji}}) \tag{5.21}$$

Então, das Equações 5.5 e 5.7, tem-se que:

$$\frac{\partial U_{(n+1)j}}{\partial w_{(n+1)ji}} = Lor[(w_{(n+1)ji}\, T\, Y_{(n)i}) - (S_{i' \neq i}\, T\, Y_{(n)i'})] \cdot Lor[Y_{(n)i} - w_{(n+1)ji}] \tag{5.22}$$

Cálculo dos erros das unidades da camada de saída, denotada por $\delta_{(n+1)j}$. Então:

$$\delta_{(n+1)j} = (T_j - Y_j) \cdot \Psi[U_{(n+1)j}] \tag{5.23}$$

Ajuste dos pesos entre a camada intermediária e a de saída é definido como:

$$\frac{\partial J}{\partial U_{(n+1)j}} = -\delta_{(n+1)j} \tag{5.24}$$

$$\Delta w_{(n+1)ji} = \delta_{(n+1)ji} \cdot (\frac{\partial U_{(n+1)j}}{\partial w_{(n+1)ji}}) \tag{5.25}$$

As fórmulas para a atualização dos pesos passam a ser:

$$w_{(n+1)ji} = w_{(n)ji} + \Delta w_{(n+1)ji} \tag{5.26}$$

$$w_{(n+1)ji} = w_{(n)ji} + \eta(\frac{-\partial J}{\partial w_{(n)ji}}) + \beta \Delta w_{(n)ji} \tag{5.27}$$

Camada de Entrada (l)/Camada Intermediária (i)

Função ativação:

$$\Psi[U_{(n)i}] = \tanh[U_{(n)i}] \tag{5.28}$$

Derivada da função ativação:

$$\Psi'[U_{(n)i}] = 4\frac{e^{-2U_{(n)i}}}{[1+e^{-2U_{(n)i}}]^2} \tag{5.29}$$

Então, a regra delta para RNA E/OU é dada para as derivadas parciais de J com respeito aos pesos $w_{(n)il}$ como segue:

$$\Delta w_{(n)il} = -\eta(\frac{\partial J}{\partial w_{(n-1)il}}) \tag{5.30}$$

$$\frac{\partial J}{\partial w_{(n-1)il}} = (\frac{\partial J}{\partial U_{(n)i}}) \cdot (\frac{\partial U_{(n)i}}{\partial w_{(n-1)il}}) \tag{5.31}$$

Então, das Equações 5.5 e 5.7, tem-se que:

$$\frac{\partial U_{(n)i}}{\partial w_{(n-1)il}} = Lor[(T_{l'\neq l}(w_{(n-1)il'} * Y_{(n-1)l'}) - (w_{(n-1)il} * Y_{(n-1)l})] \cdot Y_{(n-1)l} \tag{5.32}$$

Cálculo dos erros das unidades da camada intermediária, denotada por $\delta_{(n)i}$. Então:

$$A = Lor[(w_{(n+1)ji}\, T\, Y_{(n)i}) - (S_{i'\neq i}\, T\, Y_{(n)i'})]$$
$$B = Lor[w_{(n+1)ji} - Y_{(n)i}]$$

$$\delta_{(n)i} = \Psi'[U_{(n)i}] \sum_j \delta_{(n+1)j} \cdot A \cdot B \tag{5.33}$$

Ajuste dos pesos entre a camada intermediária e a da saída é definido como:

$$\frac{\partial J}{\partial U_{(n)i}} = -\delta_{(n)i} \tag{5.34}$$

$$\Delta w_{(n-1)il} = \delta_{(n)i} \cdot (\frac{\partial U_{(n)i}}{\partial w_{(n-1)}il}) \tag{5.35}$$

As fórmulas para a atualização dos pesos passam a ser:

$$w_{(n)il} = w_{(n-1)il} + \Delta w_{(n)il} \tag{5.36}$$

$$w_{(n)ji} = w_{(n-1)il} + \eta(\frac{-\partial J}{\partial w_{(n-1)il}}) + \beta \Delta w_{(n)il} \tag{5.37}$$

5.2 Etapas do NNES

O algoritmo desenvolvido compreende as seguintes etapas:

- Atributos dos dados entrada/saída;
- Conjunto de exemplos;
- Número de neurônios das camadas de entrada, intermediária e saída do NNES inicial;
- Tratamento das variáveis de entrada (linguística, booleana e quantitativa);
- Aplicação do AG (otimização da camada intermediária);
- Treinamento da rede - algoritmo com inspiração no de retropropagação;
- Refinamento.

5.2.1 Atributos dos Dados Entrada/Saída

Nesta etapa, obtem-se dois arquivos, um correspondente aos dados de entrada e outro à saída do NNES inicial, de tal forma a se saber quantos neurônios devem ser aplicados na formação deste sistema.

5.2.2 Conjunto de Exemplos

Cria-se um arquivo de exemplos fornecidos pelo especialista de domínio, para montar o NNES inicial e outro para testar o NNES já refinado.

5.2.3 Número de Neurônios das Camadas de Entrada, Intermediária e Saída do NNES Inicial

O número de neurônios da camada de entrada, da intermediária e o da de saída, são determinados através da etapa de EC de um especialista de domínio. Utiliza-se, nesta etapa, os grafos E/OU.

5.2.4 Etapas do Algoritmo de Treinamento para o NNES

Nesta etapa, o GENBACK não somente treinará a rede inicial mas um número de populações de redes a serem admitidas para a escolha da melhor topologia. Neste ponto o AG servirá para auxiliar na determinação da topologia que melhor se adapta a rede em questão. Além disso, o critério utilizado para a paralização do treinamento da rede será pelo erro e por épocas.

Como em um AG[6] se inicia com um gene qualquer, este gene pode exprimir, de certa forma, o número de elementos que há na camada intermediária de uma RNA. Desta forma, o processo de criação da população inicial de indivíduos, ou seja, da população de redes em relação à camada intermediária da RNA, utiliza o seguinte procedimento para a sua montagem:

1. Inicia-se com todas as redes tendo o mesmo número de neurônios de entrada e de saída;
2. Cada indivíduo (rede) da população terá um número diferente de neurônios na camada intermediária;
3. Usa-se para a criação da população inicial de indivíduos, ou seja, da população de redes, a distribuição uniforme ou distribuição gaussiana [140].

Assim, após a montagem do NNES inicial e a escolha de uma população de redes, inicia-se a escolha da melhor topologia de rede da seguinte forma:

1. Codificação da cadeia de cromossomos por um código binário;
2. Treina-se cada rede, utilizando-se o algoritmo de treinamento GENBACK, através de um número de passos, com certa porcentagem de erro;
3. Avaliação da aptidão de todos os indivíduos da população, por meio de uma função custo;
4. Verificação dos critérios de encerramento por número de gerações;
5. Atribui-se uma nota para cada rede escolhida;
6. Ordena-se cada indivíduo da população, em ordem decrescente ou crescente;
7. Emprega-se o AG para seleção, *crossover* e mutação.

[6]O AG utilizará reprodução assexuada com *crossover* (cruzamento) e mutação de cromossomos.

8. Determinação de uma geração;
9. Volta-se ao passo 2, até se obter uma topologia, que corresponda aos objetivos esperados;
10. Escolhida a topologia adequada passa-se para a etapa de desenvolvimento do RBES.

Entretanto, é necessário que se faça algumas considerações no uso do AG como segue:

Criação da População Inicial

A criação da população inicial foi realizada com opção de distribuição uniforme e gaussiana sendo que, para a distribuição gaussiana, foi utilizada a seguinte equação:

$$y = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)\epsilon^{-0,5\left(x-\frac{\mu}{\sigma}\right)^2}$$

Onde:

y - variável dependente
σ - desvio padrão da distribuição
x - variável independente
μ - média da distribuição

Condição de Parada do AG

A parada do processo de utilização do AG, é realizada pelo número de gerações, podendo ter opção por convergência (desvio padrão ou média aritmética).

Avaliação da Aptidão

A avaliação da aptidão de cada rede criada foi auxiliada através do processo de treinamento da rede pelo GENBACK, para determinar quais redes são mais aptas a sobreviverem. A função custo, utilizada neste processo, foi obtida em função do erro total ocorrido durante o treinamento da rede em questão, isto é, por meio do erro médio quadrático bem como pelo Número de Neurônios na Camada Intermediária (N_{CI}) de cada indivíduo da população gerada. Assim, a função custo foi obtida de modo a descrever o problema em estudo de duas formas:

$$A = \frac{N_{CI}}{Erro\_Total} \tag{5.38}$$

$$A = \frac{1}{Erro\_Total + N_{CI}} \tag{5.39}$$

Observou-se que ambas as Equações 5.38 e 5.39 definem o problema analisado. A Equação 5.38 faz, em geral, com que a rede vencedora tenha mais neurônios que a da rede inicial, enquanto que a Equação 5.39, a rede vencedora passa, em geral, a ter menos neurônios que a rede inicial. Isto pode ser explicado da seguinte forma:

Na Equação 5.38, a aptidão é proporcional ao N_{CI} e inversamente proporcional ao Erro_Total obtido no aprendizado de cada população de redes. Nota-se, caso fosse a aptidão apenas o inverso do Erro_Total, que quanto maior o Erro_Total menos apta seria a população da rede em questão. Contudo, como a aptidão também depende proporcionalmente ao N_{CI}, esta Equação faz com que a rede vencedora, ou seja, a rede mais apta tenha o N_{CI} maior que o N_{CI} da rede inicial. Enquanto isso, na Equação 5.39, a aptidão é inversamente proporcional tanto ao N_{CI} quanto ao Erro_Total. Desta forma, quanto maior o N_{CI} ou o Erro_Total, faz com que a rede vencedora tenha o N_{CI} menor que o N_{CI} da rede inicial.

Torna-se necessário ressaltar que o ideal seria encontrar uma função custo que, para o mesmo problema, execute o processo de otimização tanto para a minimização quanto para a maximização da camada intermediária. Aliás, este é o problema chave neste algoritmo e, apesar de não apresentarmos, ainda, uma alternativa, continuamos o trabalho neste sentido.

A natureza deste problema pode ser compreendida da seguinte maneira:

Imaginemos que uma nova função de custo para descrever este problema seja a combinação linear das Equações 5.38 e 5.39, isto é:

$$A = A_1 * \left(\frac{N_{CI}}{Erro\_Total}\right) + A_2 * \left(\frac{1}{Erro\_Total + N_{CI}}\right) \tag{5.40}$$

Sejam A_1 e A_2 dois coeficientes que se possa colocá-los em um determinado espaço. Se A_1 for igual a zero, somente a segunda parcela da Equação 5.40 permanece. Caso

contrário, somente a primeira parcela na Equação 5.40 é a que resta, já que $A_1 \neq 0$ e $A_2 = 0$. Observou-se que ao ser utilizado esta função custo no problema abordado o desempenho do sistema não foi bom. Contudo, se esta função custo for utilizada com cada uma das parcelas que a compõe separadamente é possível se obter um desempenho razoável do sistema. Então, pode-se inferir que ocorrerá uma região, em termos visuais, em que acontecerá tanto A_1 quanto A_2, que serão diferentes de zero.

Acredita-se então que, este problema de descrição da função custo para o objetivo de um dado projeto é um problema filosófico que ocupa a otimização a muito tempo.

Os primeiros sistemas automatizados surgiram como o resultado das invenções, por exemplo, do sistema de orientação de moinhos de vento e o regulador de Watt. Durante os anos que precederam a Segunda Guerra Mundial, alguns trabalhos foram realizados com os primeiros desenvolvimentos da teoria de sistemas automatizados. Estes trabalhos basearam-se, principalmente, nas ferramentas matemáticas como a Transformada de Laplace e as funções de variáveis complexas, que permitiram a realização de numerosos projetos inteiramente satisfatórios [9].

Com a vulgarização dos meios de cálculo, um outro método de síntese de sistemas automatizados começou a surgir. Este procurava resolver o problema, não do projeto de um sistema que funcionasse, mas de um melhor sistema possível. Para definir melhor o problema no sentido de um novo aspecto, foi necessário *precisar* matematicamente a noção de *melhor*, de modo a se levar em conta os meios que se dispunha. Este *melhor* foi caracterizado pela escolha de uma função custo e os meios que se dispunha [9].

Desta forma, pode-se notar que o interesse e a dificuldade de definir uma função de custo *melhor* continua sendo, ainda, uma tarefa árdua para se descrever matematicamente esta função em relação as variáveis que poderiam influenciar na otimização de um dado problema.

Nesta etapa deste trabalho, o que se obteve foi um ponto de partida para se realizar este projeto. E este impulso inicial permitiu detectar soluções que foram convenientes e soluções que não foram convenientes.

Ordenação da População de Redes

A ordenação de uma população de indivíduos pode ser feita em ordem crescente ou decrescente. Optou-se pela ordem decrescente, por ser esta a forma como ocorre na natureza, isto é, baseia-se no Princípio da Teoria Evolucionista de Darwin: *Os mais aptos sobrevivem, enquanto os outros têm a tendência de morrerem mais do que participarem da nova geração* [73].

Processo de Seleção

Na seleção foi usado o método da roleta ponderada onde os indivíduos mais aptos têm maiores chances de serem escolhidos em relação a toda população de redes.

Processo de OG

No operador *crossover*, escolhe-se um número x de pares de casais para haver o cruzamento dos indivíduos e, após, gerarem novos filhos. A escolha é feita pelo processo da roleta ponderada, onde esta escolha é realizada respeitando-se a condição daqueles indivíduos que possuirem um valor $< ou = 0,6$, por exemplo. Estes são selecionados e se reproduzem gerando novos filhos. Assim, estes indivíduos participam da nova geração da primeira metade da população total obtida.

Quanto ao operador mutação, escolheu-se o processo de troca simples, onde o tamanho da cadeia de cromossomos é mantida fixa. A pertubação ocorre quando o valor booleano 1, torna-se 0 e 0 fica 1, respeitando-se a condição de $< ou = 0,1$, por exemplo.

Nova População

Nesta etapa a formação da nova população de redes é gerada da seguinte forma: a primeira metade da nova população de redes é determinada pelos OG, ou seja, pelos operadores de *crossover* e mutação, enquanto a metade restante é obtida pelo processo de seleção.

Uma outra consideração, com relação a implementação do AG, deve-se ao fato em que nas gerações subseqüentes pode ter ocorrido o surgimento dos novos indivíduos que não sofreram *crossover* e/ou mutação. Nesta etapa, guardam-se os pesos, os erros e o número de épocas de cada rede gerada. Os novos indivíduos são treinados até o número de épocas pré-determinado, desde a primeira geração até a atual, enquanto aqueles que permaneceram inalterados são treinados apenas em relação

ao número de épocas naquela geração. Desta forma, ao final deste procedimento tem-se não somente a RNA otimizada mas também treinada.

5.2.5 Refinamento do NNES

Esta etapa ocorre após o NNES inicial ser determinado, treinado com um algoritmo de aprendizado, que neste caso é o GENBACK, utilizando um conjunto de exemplos e com o processo de otimização da camada intermediária por AG. Então é escolhida a melhor topologia de rede. Assim, um novo conjunto de exemplos é utilizado para testar o NNES refinado, antes que se passe para a etapa seguinte de desenvolvimento do sistema proposto, isto é, a construção do RBES.

Capítulo 6

Algoritmo de Extração de Regras Fuzzy para RNA Fuzzy (FUZZYRULEXT)

As RNA têm demonstrado serem uma importante ferramenta para a realização de tarefas como mapeamento de funções complexas e representação de conhecimento especializado na área de IA. A sua característica de habilidade de aprendizado sobre uma base de exemplos abriu novas perspectivas nas tentativas realizadas por pesquisadores para a automatização da AC especializada. Contudo, o seu comportamento como sistemas denominados *caixa-preta*, em que o processamento interno e a representação do conhecimento não podem ser inspecionados, acompanhados ou mesmo compreendidos pelo usuário [81], tem dificultado significativamente o seu uso para a representação de bases de conhecimentos em sistemas especializados. Isto ocorre porque um processo de explanação torna-se muito difícil quando o processamento de informações para a tomada de uma decisão é realizada por uma RNA. Não existem critérios ou meios claramente definidos, nos quais um usuário possa obter uma clara explanação sobre as respostas obtidas de um sistema especialista baseado em modelos conexionistas.

As RNA baseadas em conhecimento desenvolvidas em [64][42][122][67], apresentam uma condição favorável para serem implementadas em sistemas especialistas híbridos devido a sua capacidade de explanação. Porém, elas apresentam uma limitação quanto à dimensão da estrutura da RNA. Redes que envolvam elevadas dimensões (número de neurônios) apresentam dificuldades de implementação, podendo ocorrer uma explosão combinatória na geração dos neurônios da rede. Neste sentido, como forma de aumentar a potencialidade da integração de sistemas sim-

bolistas e conexionistas, linhas de pesquisas consideram a possibilidade de se realizar, formalmente, um processo de regras a partir dos pesos de uma rede treinada [64][178][39][134][133][50]. Neste caso, a rede pode ter sido treinada a partir de qualquer algoritmo de treinamento, incluindo o de retropropagação.

A aprendizagem e representação de relações e funções não-lineares de uma RNA estão codificadas de uma forma, a princípio, não compreensível, nos vetores de pesos. Estes algoritmos de extração de regras buscam explorar a forma de representação destes pesos, estabelecendo critérios de validação do treinamento de uma RNA [31].

Esta análise considera três características onde uma RNA treinada *codifica* um conhecimento específico [50]:

1. Na própria topologia da RNA;
2. Na função de ativação associada aos neurônios da camada intermediária e de saída;
3. No conjunto de pesos que interligam os neurônios.

A tarefa de extração de conhecimento de uma RNA consiste na interpretação da forma de cooperação existente nas características 1, 2 e 3 de um treinamento.

A Figura 6.1 mostra uma visão geral do sistema proposto neste trabalho, resumindo as ferramentas necessárias para o desenvolvimento deste, onde demonstra a aplicação da técnica de extração de regras *fuzzy*, juntamente com o seu algoritmo, proposta neste Capítulo.

6.1 Descrição da Técnica de Extração de Regras para o NNES Fuzzy

A técnica que aqui vai ser desenvolvida é inspirada nos trabalhos de [64][134][133] [135][83][84][85]. Entretanto, ela difere nos seguintes pontos:

1. É utilizado um processo intermediário entre a saída do NNES para o RBES, isto é, o grafo E/OU;

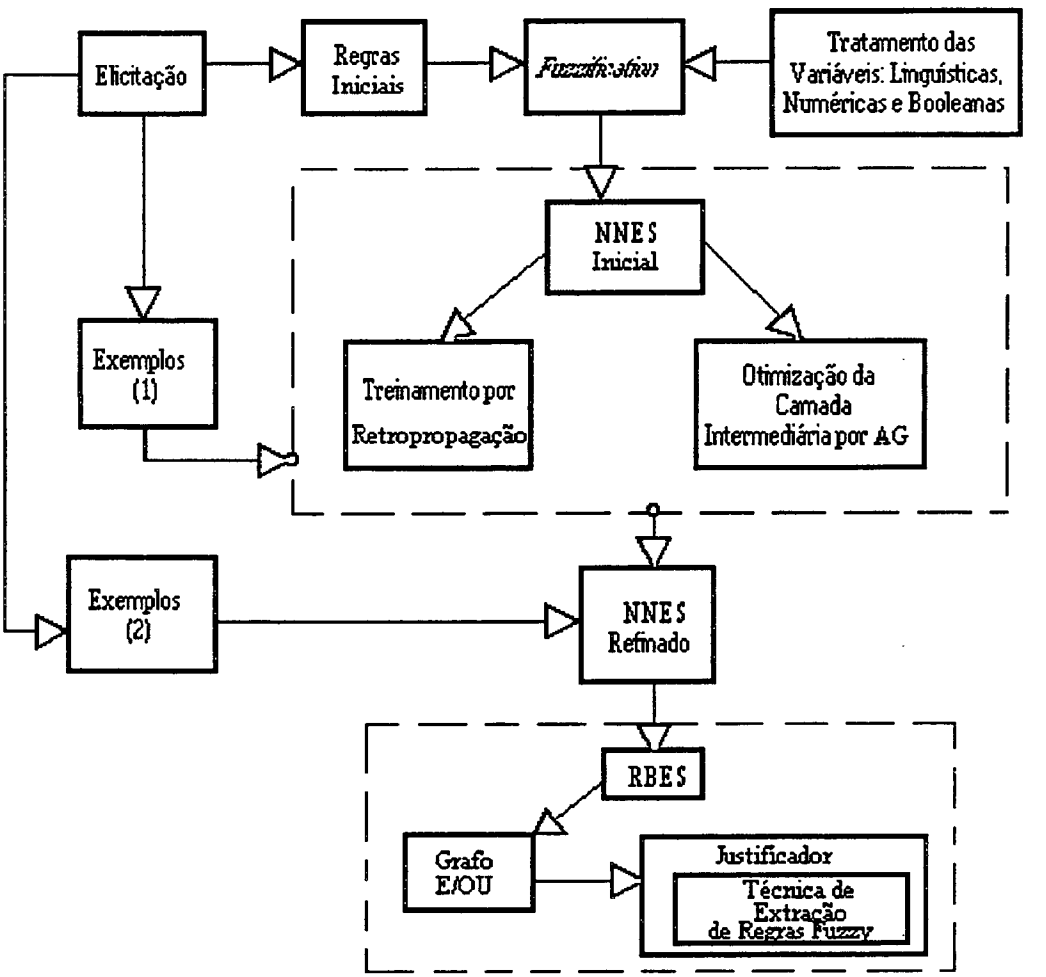

Figura 6.1: Uma visão geral do sistema proposto

2. As entradas não são funções de pertinência definidas como um vetor de três dimensões, como exemplificadas na Equação 1.2. Aqui, as entradas são tratadas como valores *fuzzy*, ou seja, as entradas são representadas por graus de pertinência e não como números *fuzzy*-π;

3. Os pesos acumulados wet_i são tratados pelo *Fuzzy Rule Extraction Algorithm* (FUZZYRULEXT), não somente como os pesos máximos, mas também como os pesos mínimos. Isto quer dizer que são utilizados os wet_i máximos entre as camadas de saída e intermediária, enquanto que os wet_i mínimos são usados entre as camadas intermediária e de entrada, pois isto se deve ao fato de se ter alguns dados de entrada negados;

4. Não é usada aqui a condição de parada de geração do número de regras dada pela Equação 1.11, porque nesta etapa foi usado o processo de eliminação das regras geradas como redundantes. Desta forma, reduziu-se, consideravelmente, este número de regras e, conseqüentemente, a construção do RBES.

5. Como as entradas do NNES não foram utilizadas na forma de função de pertinência, com conjuntos *fuzzy-*π, mas diretamente em graus de pertinência, as propriedades apresentadas para se obter as cláusulas de uma regra em ambos algoritmos de Y. Hayashi e S. Mitra et al. foram simplificadas, como será visto nas próximas subseções.

6.1.1 Grafo E/OU

Nesta etapa o conhecimento das regras iniciais somado ao conhecimento oriundo dos conjuntos de exemplos cedidos por um dado especialista de um domínio, os quais foram mapeados em uma RNA, isto é, o NNES, após o treinamento deste pelo algoritmo GENBACK e apresentado a este um novo conjunto de exemplos para testar o aprendizado e sua capacidade de generalização, são *desmapeados* em um Grafo E/OU, com a finalidade de facilitar a tarefa do mapeamento destes conhecimentos em um sistema simbólico novamente, ou seja, o RBES.

Quanto a maneira de se mapear os resultados obtidos na saída do NNES para os Grafos E/OU, sugere-se que sejam revistos os conceitos desta ferramenta no Capítulo 2, bem como os Capítulos 4 e 5 sobre a metodologia e o algoritmo de aprendizado, respectivamente, empregados na realização deste trabalho, para que se tenha maiores informações sobre o assunto tratado aqui. Além disso, acredita-se que seja desnecessário voltar a este tema aqui por ser o modo de emprego desta ferramenta similar a utilizada no desenvolvimento do GENBACK.

6.1.2 Geração do Caminho por Backtracking

Nesta etapa o usuário pode perguntar ao sistema porque ele inferiu uma conclusão particular. O sistema responde com regras *Se/Então* aplicáveis para o caso em questão. Estas regras não se encontram explícitas na base de conhecimento. Elas são geradas pelo sistema de inferência dos pesos das conexões como necessidade para explanações. Uma conclusão particular, olhando uma dada saída j, é inferida dependendo de uma medida de certeza chamada bel_j. É garantido que os nós (unidades) de saída j com $bel_j > 0$ são escolhidos para obter a justificação.

A seguir serão descritas cada etapa da formalização do algoritmo para extração de regras, bem como a definição de bel_j.

Suponha-se que temos uma RNA *fuzzy* com uma camada de entrada, uma de saída e N intermediárias (Veja a Figura 6.2 para maiores detalhes), usando-se $w_{(n)ji}$ para denotar o peso entre o nó i na camada n ($n = 0, 1, \cdots, n+1$) e o nó j na camada $n+1$, bem como usando-se $Y_{(n+1)j}$ para denotar a saída do nó j na camada de saída $n+1$.

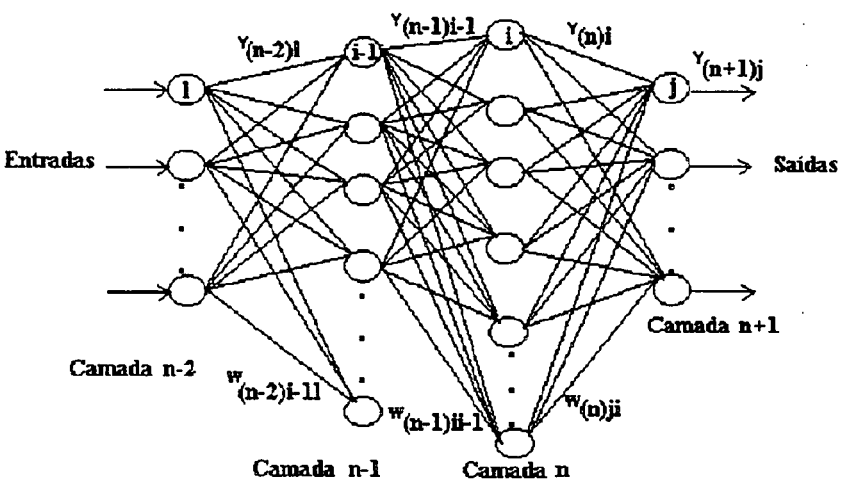

Figura 6.2: Rede neural *fuzzy*

Camada de Saída

Nesta etapa seleciona-se os neurônos i na camada precedente que têm um impacto positivo na conclusão do neurônio de saída j. Então escolhe-se o neurônio i da camada intermediária n se $w_{(n)ji} > 0$. Seja o conjunto de m neurônios da camada intermediária n, assim selecionado:

$$m_{(n)i} = \{a_1, a_2, \cdots, a_{m_{(n+1)j}}\}$$

e seja seus pesos de conexões para o neurônio j na camada de saída dado como:

$$wet_{(n)a_k} = \{w_{(n)ja_1}, w_{(n)ja_2}, \cdots, w_{(n)jam_{(n+1)j}}\}$$

Camada Intermediária

Seleciona-se o neurônio i na camada intermediária n se:

$$Y_{(n)i} > 0$$

$$wet_{(n-1)i} = max[wet_{(n)a_k} + w_{(n-1)a_k i}] \tag{6.1}$$

com $w_{(n-1)a_k i} > 0$ e $Y_{(n)i}$ a saída do neurônio i, ou seja, o estado do neurônio i na camada n.

No caso que uma RNA tenha mais de uma camada intermediária, $Y_{(n)i}$ é determinado no passo para frente quando do uso do algoritmo de treinamento do NNES na etapa de apresentação do conjunto de padrões para teste. Contudo, quando a RNA tiver apenas uma camada intermediária, $Y_{(n)i}$ será o valor do padrão de entrada.

Seja o conjunto de $m_{(n-1)i}$ neurônios da camada intermediária $n-1$ assim escolhido dado como:

$$m_{(n-1)i} = \{a_1, a_2, \cdots, a_{m_{(n)i}}\}$$

e seus pesos de conexões acumulados para o neurônio j na camada de saída ao longo do caminho dos pesos máximos ser, $\{wet_{(n-1)a_1}, wet_{(n-1)a_2}, \cdots, wet_{(n-1)am_{(n)i}}\}$ obtidos pela Equação 6.1. Note que esta heurística assegura que cada um dos $m_{(n-1)i}$ neurônios selecionados tenham uma resposta de saída significante. Em outras palavras, isto ajuda a selecionar caminhos ao logo dos quais cada par de neurônios possui uma correlação positiva significante ou uma influência em cada um dos outros neurônios. Também é possível se dizer que isto implica em escolher um caminho com neurônios que são ativos para decidir a conclusão que está sendo justificada. Pode também habilitar cada neurônio i para ser mantido ao longo de um dos $m_{(n-1)i}$ caminhos selecionados previamente, que provê o maior $wet_{(n-1)i}$ peso da rede.

Camada de Entrada

Seja o processo da Equação 6.1 resultante em $m_{(n-2)l}$ neurônios escolhidos na camada de entrada. Estes neurônios indicam entradas que são conhecidas e têm contribuído para a conclusão final no neurônio j na camada de saída $(n+1)$. Pode acontecer que $m_{(n-2)l} = 0$, tal que nenhuma justificação clara pode ser dada para

um caso particular de entrada/saída[1]. Isto implica que nenhum caminho adequado pôde ser selecionado pela Equação 6.1 e o processo termina.

Seja o conjunto dos neurônios de entrada $m_{(n-2)l}$ selecionados como $m_{(n-2)l} = \{a_1, a_2, \cdots, a_{m_{(n-1)i}}\}$ e seus pesos correspondente ao neurônio j na camada de saída $n+1$ ser dado como $\{wet_{(n-2)a_1}, wet_{(n-2)a_2}, \cdots, wet_{(n-2)am_{(n-1)i}}\}$. Então estes neurônios são arranjados na ordem decrescente dos seus impactos de rede, onde define-se o Impacto da Rede (IR) para o neurônio i na camada $n-1$ como:

$$IR_{(n-1)i} = Y_{(n-1)i} * wet_{(n-1)i} \tag{6.2}$$

Então, geram-se as cláusulas para uma regra *Se/Então* desta lista ordenada, onde l_s indica os neurônios de entrada selecionados para as cláusulas e l_p denota os neurônios de entrada restantes do conjunto $\{a_1, a_2, \cdots, a_{m_{(n-2)l}}\}$, tal que:

$$m_{(n-2)l} = |l_s| + |l_p|$$

e $|l_s|$, $|l_p|$, referem-se, respectivamente, ao número de neurônios selecionados e restantes do referido conjunto. Esta heurística permite a seleção daqueles neurônios de entrada ativos correntes contribuindo para a conclusão final como a parte antecedente de uma regra. Daí, habilita as entradas dos padrões de teste ativos atuais para influenciar a base de conhecimento gerada (pesos das conexões aprendidos durante o treinamento) em produzir uma regra para justificar a inferência atual (decisão olhando o padrão de teste corrente). O modelo pode, portanto, funcionar como um sistema especialista conexionista *fuzzy*. Um especialista humano pode ser usado neste estágio, para verificar a lógica da regra gerada pelo modelo no apoio da conclusão inferida atual. Estas regras poderiam, também, ser usadas para a base de conhecimento para um SE tipo classificação tradicional para o problema em consideração.

[1]Esta condição não se aplica ao exemplo correspondente a classificação dos números de 0 à 15 na forma binária, exemplo este a ser visto no próximo capítulo. Isto se deve ao fato do caso do número zero (0) ser representado no NNES com todas as entradas negadas.

6.1.3 Geração das Cláusulas

Dedução do Antecedente

Para um neurônio l_{s_l} na camada de entrada selecionado para a geração da cláusula, a característica de entrada, sendo booleana ou numérica, u_{s_l} é obtida como:

$$u_{s_l} = (l_{s_l} - 1) + 1$$

onde $-0,8 \leq l_{s_l} \leq n$, $-0,8 \leq u_{s_l} \leq n$ e n é a dimensão do vetor do padrão de entrada.

O antecedente da regra é dado na forma de vetores de graus de pertinência, com as propriedades[2] sendo determinadas como:

$$prop = \begin{cases} -0,8 & \text{Se } u_{s_l} \leq -0,8 \\ 1 & \text{Se } u_{s_l} = 1 \\ -0,8 < x < 1 & \text{Se não} \end{cases}$$

Aqui, a característica de entrada u_{s_l} corresponde a parte apropriada do vetor de padrões de teste.

Para um neurônio l_{s_l} na camada de entrada selecionado para a geração da cláusula, a característica de entrada [3]: sendo lingüística, u_{s_l} é obtida como:

$$u_{s_l} = (l_{s_l} - 1) \, mod \, 3 + 1$$

onde $-.8 \leq l_{s_l} \leq 3n$, $-.8 \leq u_{s_l} \leq n$ e n é a dimensão do vetor do padrão de entrada.

O antecedente da regra é dado na forma lingüística com as propriedades[4] lingüísticas sendo determinadas como, por exemplo:

$$prop = \begin{cases} forte & \text{Se } u_{s_l} \geq .8 \\ \text{moderado} & \text{Se } -.4 < u_{s_l} < .7 \\ fraco & \text{Se } u_{s_l} \leq -.4 \end{cases}$$

[2]Os valores apresentados nestas propriedades são valores heurísticos definidos em função das simulações efetuadas para o desenvolvimento deste algoritmo de extração de regras *fuzzy*.

[3]Entenda-se aqui a notação mod 3 como sendo o resto da divisão da expressão matemática $(l_{s_l} - 1)$ por 3.

[4]Os valores apresentados nestas propriedades são valores heurísticos definidos em função das simulações efetuadas para o desenvolvimento deste algoritmo de extração de regras *fuzzy*.

Este procedimento é repetido para todos os $|l_s|$ selecionados pela Equação 6.2 para gerar um conjunto de cláusulas antecedentes conjuntivas para a regra olhando a inferência do nó de saída j. Todas as características de entrada (do padrão de teste) não necessitam precisamente ser selecionadas por geração da cláusula do antecedente.

Dedução do Conseqüente

Uma forma mais natural de decisão pode ser obtida para a saída j tendo o valor de pertinência significante, considerando o valor de uma medida de certeza (bel_j) como:

$$bel_j = Y_{(n+1)j} - \sum_{i \neq j} Y_{(n-1)i}$$

Então, nota-se que a dificuldade em chegar em uma decisão particular em favor da saída j é dependente não somente do valor de pertinência $Y_{(n+1)j}$, mas também das diferenças com os valores de pertinências $Y_{(n+1)j}$, onde $i \neq j$. Desta forma, o valor[5] de bel_j sendo baixo, a dificuldade em decidir uma saída j é determinado em função da escolha do maior grau de certeza da decisão de saída. Assim:

$$prop = \begin{cases} -0,8 & \text{para } -5 \leq bel_j \leq -0,2 \\ 1 & \text{para } bel_j > -0,2 \\ Incapaz\ de\ reconhecer & \text{para } bel_j < -5 \end{cases}$$

Processo de Eliminação das Regras Redundantes

Após serem geradas as regras na saída do NNES, conforme os procedimentos descritos anteriormente, observou-se o surgimento de várias regras redundantes. É de nosso conhecimento que este tipo de regras não contribuem para a qualidade de um sistema simbólico. Pelo contrário, haverá um maior consumo de tempo computacional, bem como dos processos de busca (por exemplo, encadeamento para trás ou para a frente) para testar quais regras definem o objetivo em questão. Desta forma, desenvolveu-se o seguinte processo para auxiliar na retirada deste tipo de regra:

1. Determinar o número total de regras geradas em função da Subseção 6.1.3;

2. Criar um único vetor para os dados de entrada/saída;

[5]Os valores apresentados nestas propriedades são valores heurísticos definidos em função das simulações efetuadas para o desenvolvimento deste algoritmo de extração de regras *fuzzy*.

3. Comparar o antecedente/conseqüente de cada regra para se determinar quais são iguais;

4. Guardar as regras que são distintas;

5. Eliminar as regras redundantes;

6. Determinar o número total de regras geradas sem redundâncias.

A seguir é demonstrado através de alguns exemplos cada etapa na formalização das regras extraídas na saída do NNES.

Exemplos:

Considere a rede com três camadas dada na Figura 6.3 demonstrando uma geração de regra simples olhando as saídas Y_1, Y_2 e Y_3. Este exemplo corresponde a classificação dos números cardinais de 0 à 15 (binário) em Par, Ímpar e Primo.

Um conjunto de amostra de pesos w_{ji} e w_{il}, ativação das entradas X_l e os correspondentes valores de graus de pertinência são mostrados na Figura 6.3. Além disso vamos considerar para facilidade dos cálculos que: $m_{(n-2)l} = X_l$, $Y_{(n+1)j} = Y_j$, $w_{(n)a_k i} = w_{ji}$, $w_{(n-1)a_k i} = w_{il}$.

1. Escolha do padrão Entrada-Saída

$$X_l = \{-0,8; -0,8; -0,8; +1\}$$
$$Y_j = \{-0,8; +1; +1\}$$

2. Escolha do neurônio i com impacto positivo na conclusão de uma saída j, onde $w_{ji} > 0$. Notar que na Figura 6.3 estão apenas representados os pesos positivos.

$$m_i = \{a_1, a_2, \cdots, a_3\}$$

3. Determinação do conjunto dos pesos acumulados wet_l para o neurônio l.

$$wet_l = max[wet_{a_k} + w_{a_k l}]$$

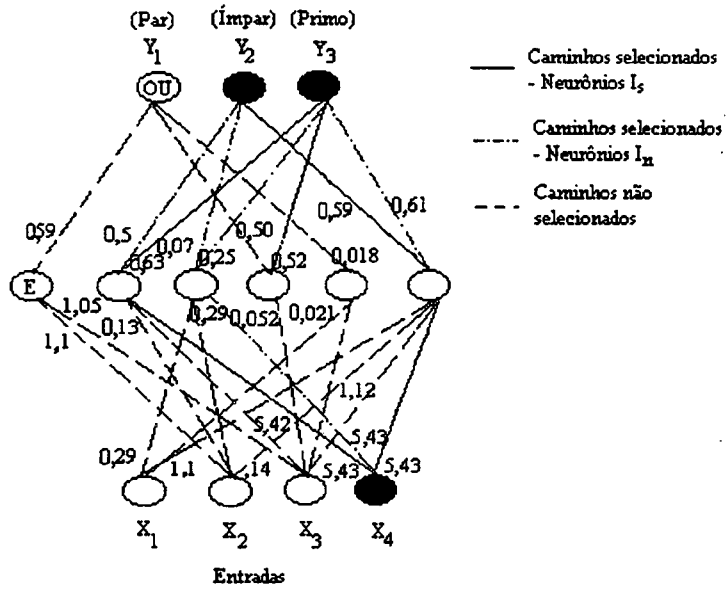

Figura 6.3: Exemplo 1 demonstrando a geração de regra por *backtracking*

Vamos determinar primeiramente os wet_{a_k}, onde $w_{ji} > 0$ e $k = 1, \cdots, 6$. Então:

$$wet_{a_k} = \{w_{ja_1}, w_{ja_2}, \cdots, w_{ja m_3}\}$$

$$w_{a_1} = \{0,59\}$$
$$w_{a_2} = \{0,50; 0,63\}$$
$$w_{a_3} = \{0,07; 0,25\}$$
$$w_{a_4} = \{0,50; 0,52\}$$
$$w_{a_5} = \{0,018\}$$
$$w_{a_6} = \{0,59; 0,61\}$$

Considerando $Y_{(n-2)l} = X_l > 0$, $wet_{(n-1)i} = wet_l$, $w_{(n-1)a_k i} = w_{il}$, $w_{il} > 0$ e $l = 1, \cdots, 4$, então:

$$wet_l = max[wet_{a_k} + w_{il}]$$

Onde:

$$wet_1 = max[0,59; 0,50; 0,63; 0,36; 0,54; 0,50; 0,52; 0,039; 1,62; 1,71] = 1,71$$

$$wet_2 = max[1,59; 1,53; 1,66; 0,20; 0,38; 0,50; 0,52; 0,16; 0,52; 0,61] = 1,66$$

$$wet_3 = max[1,64; 1,57; 1,70; 0,07; 0,25; 5,92; 5,94; 0,07; 1,64; 1,73,] = 5,94$$

$$wet_4 = max[0,59; 5,93; 6,06; 5,50; 5,95; 0,50; 0,52; 0,18; 5,95; 6,04] = 6,06$$

4. Ordenação em ordem decrescente de impacto de rede dos elementos do conjunto de pesos obtidos no item anterior, considerando $(n-1)_i$.

$$IR_l = X_l * wet_l$$

Então:

$$IR_1 = X_1 * wet_1 = -0,8 * 1,71 = -1,368$$

$$IR_2 = X_2 * wet_2 = -0,8 * 1,66 = -1,328$$

$$IR_3 = X_3 * wet_3 = -0,8 * 5,94 = -4,752$$

$$IR_4 = X_4 * wet_4 = 1 * 1,71 = 6,06$$

Em ordem decrescente de impacto de rede, tem -se que:

$$IR_4 = 6,06$$

$$IR_2 = -1,328$$

$$IR_1 = -1,368$$

$$IR_3 = -4,752$$

5. Seleção dos l_s neurônios de entrada e os l_p neurônios restantes para as cláusulas, tal que:

$$X_l = |l_s| + |l_p|$$

Onde l_s é determinado em função de $X_l > 0.1$. Então;

$$l_s = \{X_4\}$$

$$l_p = \{X_1, X_2, X_3\}$$

Assim, na Figura 6.3, os caminhos representados pelas linhas completas e as linhas com traço e ponto (caminhos selecionados) são os que representam os neurônios l_s e l_p, respectivamente, como determinados pela Equação 6.2. As linhas pontilhadas indicam os caminhos não selecionados pela Equação 6.1.

6. Geração das cláusulas

Dedução dos antecedentes:

(a) Determinação de u_{s_l} para uma entrada booleana.

$$u_{s_l} = (l_{s_l} - 1) + 1$$

Então:

$$u_{s_1} = u_{s_2} = u_{s_3} = -0,8$$

e

$$u_{s_4} = 1$$

(b) Determinação do antecedente de uma regra pelas propriedades vista na subseção 6.1.3. Como l_s escolhido foi com respeito a entrada X_4, então o antecedente será:

$$Se\ X_4\ Então \cdots$$

Dedução dos conseqüentes:

(a) Determinação da medida de certeza bel_j

$$bel_j = Y_j - \sum_{l \neq j} X_l$$

Então:

$$bel_1 = Y_1 - \sum_{l \neq j}(X_2, X_3, X_4) = -0,2$$

$$bel_2 = Y_2 - \sum_{l \neq j}(X_1, X_3, X_4) = 1,6$$

$$bel_3 = Y_3 - \sum_{l \neq j}(X_1, X_2, X_4) = 1,6$$

(b) Determinação dos conseqüentes de uma regra através das propriedades vistas na subseção 6.1.3, conclui-se que:

Se $\cdots$ *Então* Y_2

Se $\cdots$ *Então* Y_3

7. Determinar as regras *Se/Então* para o exemplo em questão. As regras geradas pelo modelo para a justificação da conclusão, olhando as saídas Y_2 (Ímpar) e Y_3 (Primo), podem ser:

Se X_4 *Então* Ímpar

Se X_4 *Então* Primo

Pode-se observar, também, que os pesos $w_{ji} = 0,5$ e $w_{ji} = 0,07$, poderiam ser outros caminhos a serem escolhidos para a formalização da justificativa da conclusão correspondente ao neurônio Ímpar, bem como para os pesos $w_{ji} = 0,25$ e $w_{ji} = 0,61$ para o neurônio Par. Optou-se pelos caminhos com pesos maiores, por estes terem maiores influências na ativação dos neurônios na saída da rede em questão.

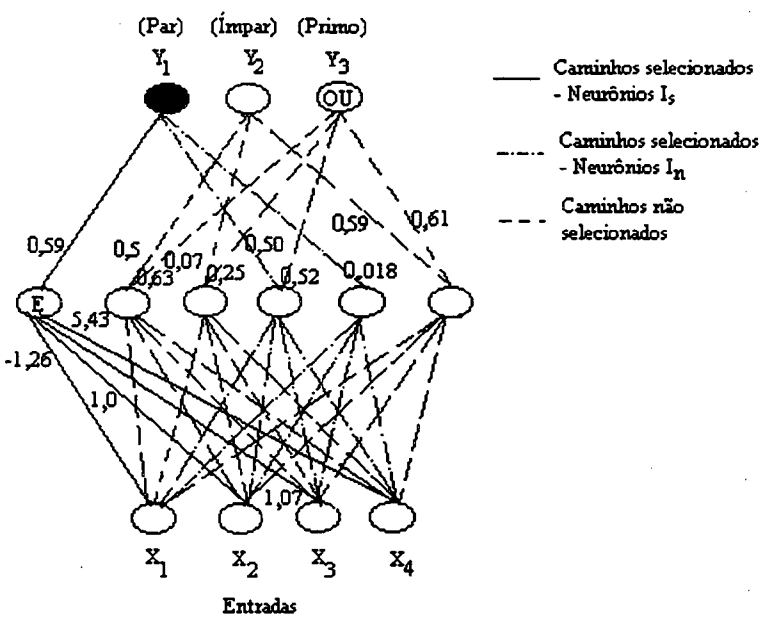

Figura 6.4: Exemplo 2 demonstrando a geração de regra por *backtracking*

O exemplo seguinte corresponde a mesma aplicação do exemplo anterior, contudo se deseja realizar a classificação do número zero (0) em par, ímpar ou primo. Para

este caso, observa-se que todas as entradas são negadas. Então, a Equação 6.1 passa a ser da seguinte forma:

$$wet_l = min[wet_{a_k} + w_{a_k l}] \tag{6.3}$$

com $w_{a_k l} > 0$ não sendo mais considerado, ou seja, usa-se os valores de pesos w_{il} negativos para se obter as premissas negadas. Veja a Figura 6.4 para uma maior compreensão da aplicação proposta[6].

Então, neste caso, a regra gerada pelo modelo para a justificação da conclusão[7], olhando as saídas Y_1 (Par), Y_2 (Ímpar) e Y_3 (Primo), pode ser:

Se não X_1 *E* não X_2 *E* não X_3 *E* não X_4 *Então* Par

No próximo exemplo, usou-se a classificação de casos idealizados por um especialista médico em crises epilépticas. Como pode ser visualizado na Figura 6.5, usando-se os valores das entradas da rede como numéricas, booleanas e lingüísticas, obteve-se a seguinte regra, seguindo os princípios utilizados também no exemplo anterior, como sendo:

Se S_2 *E* S_3 *E* S_6 *Então* D_2

6.2 Algoritmo de Extração de Regras Proposto para RNA Fuzzy

A seguir é apresentado, resumidamente, o algoritmo de extração de regras *fuzzy* (FUZZYRULEXT) proposto neste trabalho.

---

[6]Na Figura 6.4 não foram mostrados os outros valores das conexões devido ao espaçamento entre as linhas ser pequeno. Preferiu-se mostrar apenas os valores das conexões que definiram a justificativa da conclusão.

[7]Para o obtenção da conclusão para este exemplo específico, as propriedades para a dedução do conseqüente sofrem as seguintes alterações:

$$prop = \begin{cases} -0,8 & \text{para } bel_j < 2 \\ 1 & \text{para } 2 \leq bel_j \leq 20 \\ \textit{Incapaz de reconhecer} & \text{para } bel_j < -5 \end{cases}$$

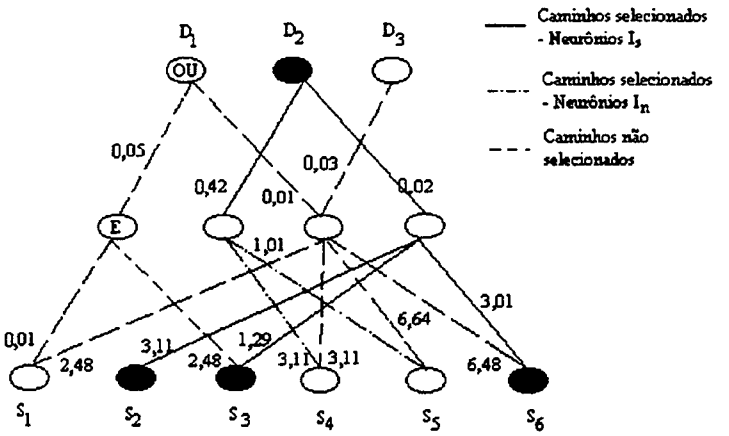

Figura 6.5: Exemplo 3 demonstrando a geração de regra por *backtracking*

1. Geração do caminho por *backtracking*

   (a) Escolha do padrão Entrada-Saída;

   (b) Escolha do neurônio i com impacto positivo na conclusão de uma saída j;

   (c) Determinação do conjunto $m_{(n)}i$ e $m_{(n-1)}i-1$;

   (d) Determinação do conjunto dos pesos acumulados $wet_{(n-1)i}$ para o neurônio i;

   (e) Seleção do conjunto dos neurônios de entrada $m_{(n-2)l}$;

   (f) Determinação dos pesos do caminho referente ao neurônio j;

   (g) Ordenação em ordem decrescente de impacto de rede dos elementos do conjunto de pesos obtidos no item anterior;

   (h) Seleção dos l_s neurônios de entrada e os l_p neurônios restantes para as cláusulas, tal que:

$$m_{(n-2)l} = |l_s| + |l_p|$$

2. Geração das cláusulas

   Dedução dos antecedentes:

   (a) Determinação de u_{s_l} para uma entrada booleana ou numérica;

   (b) Determinação de u_{s_l} para uma entrada lingüística;

(c) Determinação do antecedente de uma regra da forma vista na subseção 6.1.3;

(d) Repetem-se os itens *a* à *d* até que todos os $|l_s|$ sejam selecionados pela Equação 6.2;

Dedução dos conseqüentes:

(a) Determinação da medida de certeza bel_j;

(b) Determinação dos conseqüentes de uma regra através das propriedades vistas na subseção 6.1.3;

3. Volta-se ao item 1 caso se deseje determinar as regras *Se/Então* para justificar todas as saídas apresentadas pelos padrões de Entrada-Saída do NNES vencedor;

4. Eliminação das regras redundantes.

Capítulo 7

Simulações e Resultados

Neste Capítulo serão apresentadas as simulações referentes a três aplicações práticas, sendo que duas delas são do tipo *Toy Problems*[1] e a outra um *Real Problem*, que serão utilizadas para demonstrar a eficiência da metodologia proposta para o desenvolvimento do HES. A primeira delas, refere-se a classificação dos números cardinais de 0 à 15, na forma binária de 4 *bits*, para determinar quais são primos, pares e ímpares. A segunda, corresponde à classificação das vogais, representadas por um reticulado (5 x 7), no código ASCII (7 *bits*). A última, será o estudo de caso que ilustrará a aplicação do HES para o problema de classificação de crises epilépticas. Neste exemplo foram obtidos vários conjuntos de dados: um idealizado pelo médico e dois obtidos de prontuários. Os conjuntos de dados foram fornecidos por especialistas médicos, principalmente do Hospital Universitário da Universidade Federal de Santa Catarina (HU/UFSC). A maior contribuição foi dada pelo doutor Li Shih Min (MD, MSc), que forneceu vários exemplos para a classificação da referida doença através de um conjunto de prontuários e, também, de seu próprio trabalho de dissertação de mestrado [132]. Desta forma, um dos conjuntos de dados usados para formar um NNES inicial foi obtido com 6 sintomas, 3 classes de diagnósticos e 6 regras (caso idealizado), enquanto dois outros exemplos foram de casos extraídos de prontuários, onde o primeiro foi implementado com 32 sintomas, 4 classes de diagnósticos e 9 regras e testados com outros dois conjuntos de testes, com 9 regras e 7 regras, respectivamente. O segundo, com 32 sintomas, 4 classes de diagnósticos e 11 regras e testado para um outro conjunto de 11 regras.

[1]Os *Toy Problems* são aplicações clássicas, como são o caso dos jogos (xadrez, dama, etc.). Eles são muitas vezes usados no desenvolvimento de técnicas de IA (por exemplo, na resolução de problemas), bem como algumas vezes servem para *testar* algumas dessas técnicas. Neste sentido, a classificação dos números cardinais de 0 à 15 em primo, par e ímpar e os das vogais aqui exemplificados, não são, *verdadeiramente Toy Problems*, mas que foram escolhidos por serem simples o suficiente para a verificação do funcionamento dos algoritmos GENBACK E FUZZYRULEXT.

Este Capítulo será dividido em três partes, ou seja, a implementação do sistema proposto, as simulações correspondentes a implementação do paradigma conexionista, NNES *fuzzy*, e as simulações referentes ao desenvolvimento do paradigma simbólico, RBES, como segue.

7.1 Implementação do Sistema Proposto

A Figura 7.1 mostra a tela principal do programa desenvolvido para o treinamento e a extração de regras do NNES *fuzzy*. Foi utilizada a linguagem *Visual Basic*, versão 3.0, na montagem do mesmo. Nesta tela há vários botões: *initialization* indica o início de execução do programa. Quando este é habilitado outra tela surge, na qual são discriminados os dados iniciais do NNES *fuzzy* (número de camadas, quantidade de neurônios para cada camada). Após é necessário se definir que faixa de valores de pesos serão usados para o treinamento do NNES *Fuzzy*. O botão *weight* é o que executará esta função. Há também mais outros três botões que correspondem, respectivamente, a escolha do tipo de função de ativação (*ativation function*) a ser utilizada para o NNES *fuzzy*, a opção do uso da têmpera simulada (*simulated anneling*) e o que executa a função continuar (*continue*), quando o botão *stop*, o qual serve para parar o programa em tempo de execução, é acionado na tela correspondente ao treinamento e refinamento do NNES *fuzzy*. Ainda em relação à tela principal, existe também um *menu* principal composto por: arquivo (*file*), editor (*edit*), execução (*run*), gráficos (*graphics*), opções (*options*) e ajuda (*help*).

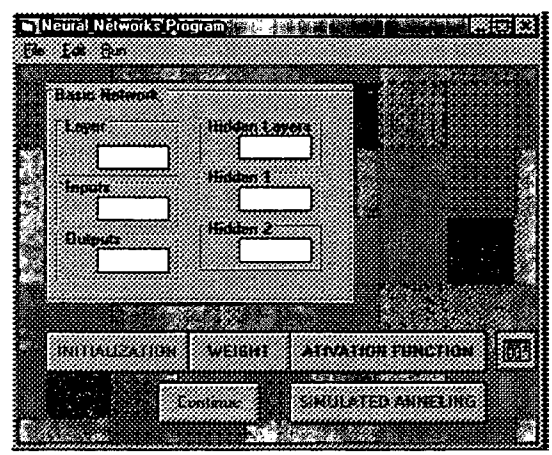

Figura 7.1: Tela principal do sistema proposto

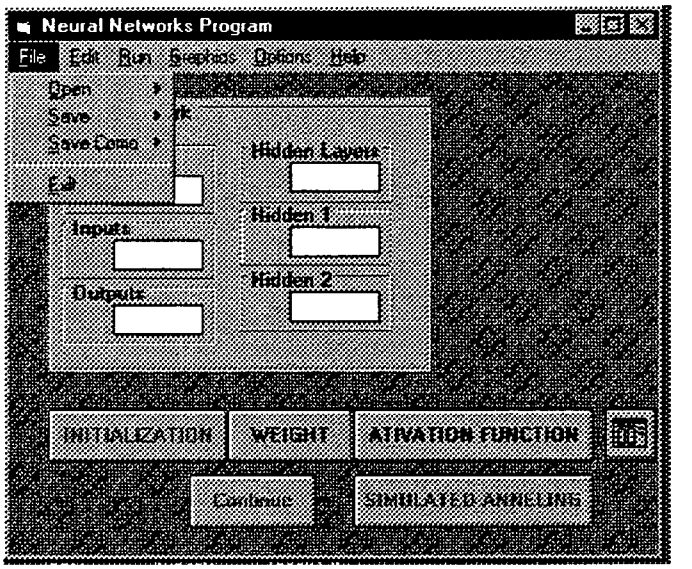

Figura 7.2: Tela do menu principal: *file*

Nas Figuras 7.2, 7.3, 7.4 e 7.5 está ilustrado o *menu* principal com maiores detalhes do funcionamento do sistema desenvolvido.

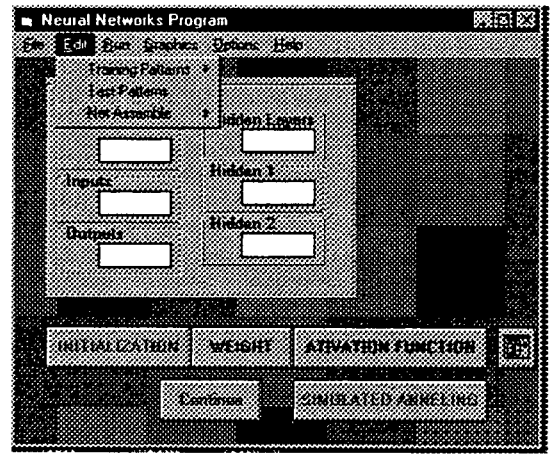

Figura 7.3: Tela do menu principal: *edit*

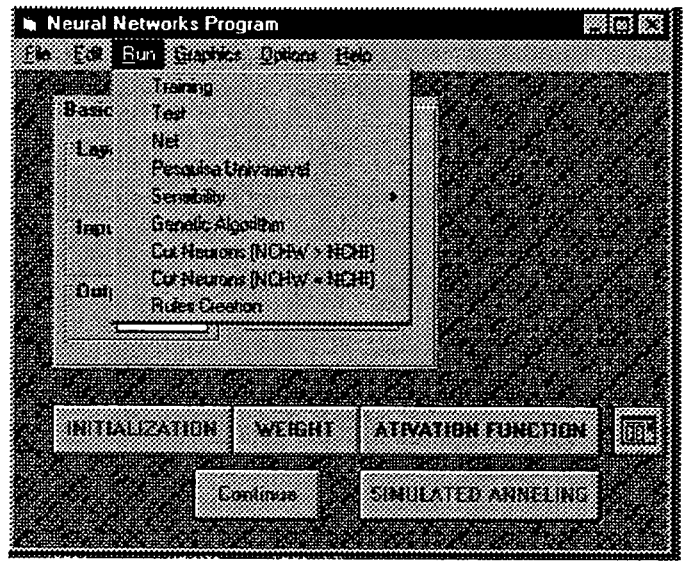

Figura 7.4: Tela do menu principal: *run*

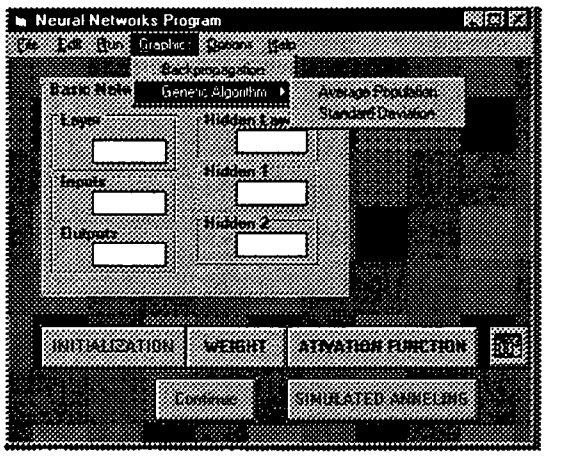

Figura 7.5: Tela do menu principal: *graphics*

A Figura 7.6 apresenta a tela correspondente à etapa de treinamento e refinamento para o NNES *fuzzy*.

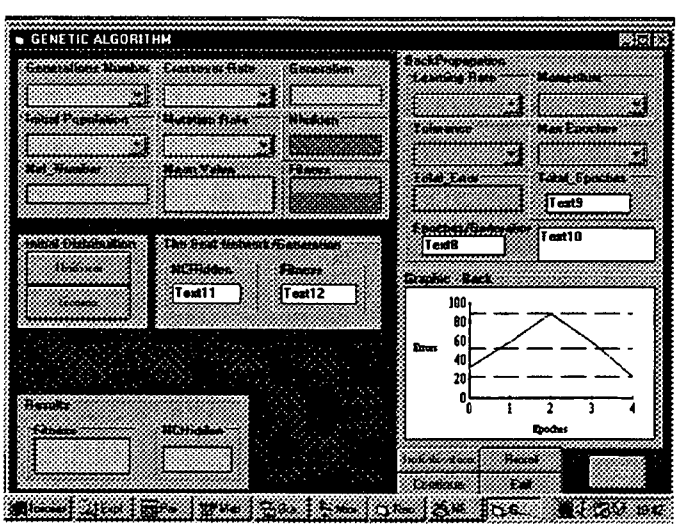

Figura 7.6: Tela para o treinamento e refinamento do NNES *fuzzy*

A Figura 7.7 ilustra a tela para a extração e a eliminação de redundâncias de regras antes de executar o *Shell* usado para a implementação do RBES.

As próximas telas demonstram a implementação para o RBES. Este sistema foi desenvolvido com o auxílio de um *Shell*. Este *Shell* chama-se Sistemas Inteligentes Aplicados (SINTA), desenvolvido pelo Laboratório de Inteligência Artificial (LIA),

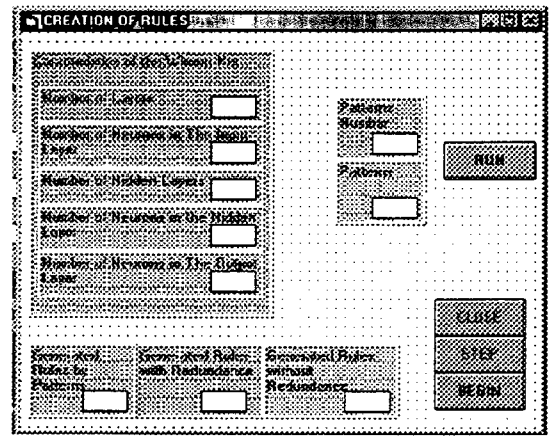

Figura 7.7: Tela de criação e eliminação de redundância de regras

localizado no Departamento de Computação da Universidade Federal do Ceará.

O *Expert* SINTA é um *Shell* implementado na linguagem de programação orientada a objetos *Borland Delphi*, dando um suporte visual de fácil operação, inspirado na arquitetura clássica do MYCIN. O *Expert* SINTA permite o desenvolvimento modular de bases de conhecimento através de uma interface de fácil manipulação e de utilitários criados para depuração. O *Expert* SINTA proporciona uma economia de tempo para os desenvolvedores da base de explicativos sobre as possíveis soluções encontradas pelo sistema. Além destas facilidades, o *Expert* SINTA traz um ambiente de trabalho que possibilita, tanto ao projetista do conhecimento quanto ao usuário final, o uso das facilidades mencionadas sem que seja necessário um conhecimento aprofundado de informática. Isto é conseguido através de um modelo visual, ao contrário de outras ferramentas que utilizam pseudo-linguagens para projeto e adaptação do conhecimento do especialista. A base de conhecimento desenvolvido pelo *Expert* SINTA é através das regras de produção. Ele também possui a opção para tratamento de incertezas ou imprecisões.

Por fim, a Figura 7.8 mostra a tela de conversão e geração das regras *fuzzy*, a Figura 7.9 ilustra um exemplo de questionamento (consulta) feita à base de conhecimento sobre um determinado domínio e as Figuras 7.10 e 7.11 apresentam os resultados alcançados e as regras na forma de premissas e conclusões (*Se/Então*) implementadas em função de uma dada aplicação.

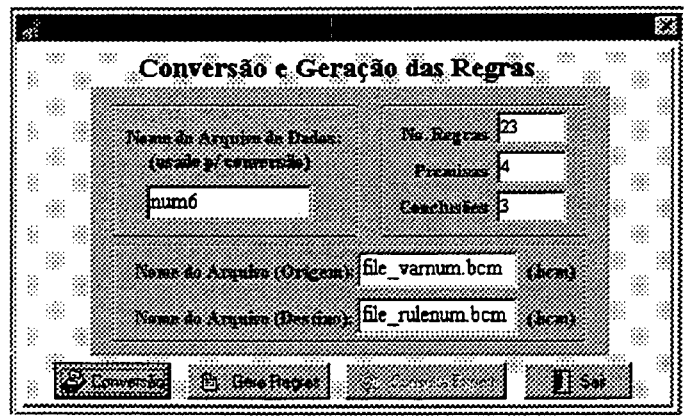

Figura 7.8: Tela do RBES: conversão e geração das regras

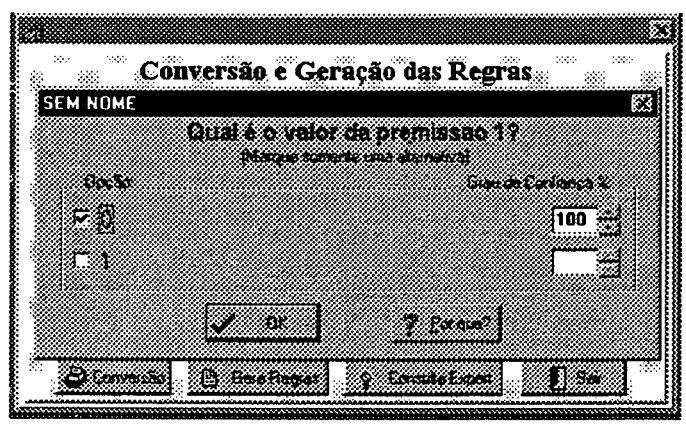

Figura 7.9: Tela do RBES: consulta feita à base de conhecimento

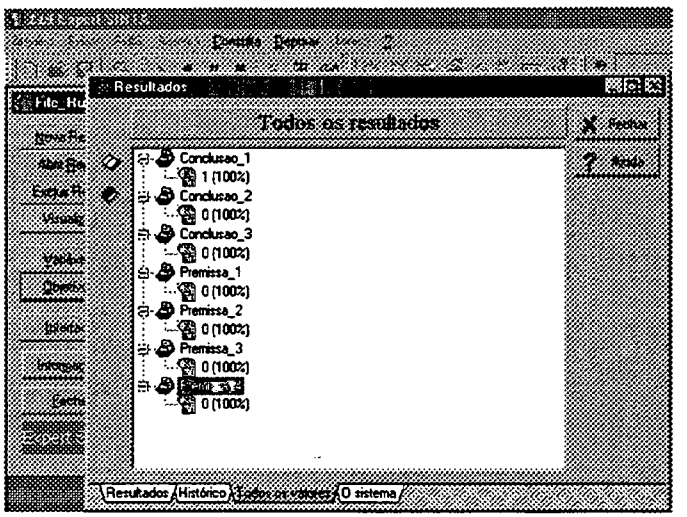

Figura 7.10: Tela do RBES: resultados

7.2 Simulações para o NNES Fuzzy

Dado um NNES *fuzzy* com as seguintes características:

1. Constituído por uma arquitetura direta com três camadas, isto é, uma camada de entrada, uma intermediária e outra de saída. A camada intermediária representando os neurônios E e a camada de saída os neurônios OU;

2. O número de neurônios iniciais na camada intermediária é determinado pelo número de regras iniciais abstraídas durante o processo de EC;

3. Treinado pela regra de aprendizado, isto é, o GENBACK, inspirado no algoritmo de retropropagação clássico. Como já mencionado e descrito no Capítulo 5 e apresentado por Brasil em [22][20][30][28], este difere do algoritmo clássico em vários pontos;

   - Otimização da camada intermediária foi feita utilizando o AG;
   - Para a escolha da melhor topologia da rede foi utilizado duas funções de aptidão que mapearam o problema em questão;
   - A população de indivíduos, ou seja, a população de redes, foi criada por distribuição gaussiana com opção de ser também uniforme;
   - Incorporação de conectivos lógicos E/OU no local do somatório de pesos;
   - Tratamento das variáveis de entrada por lógica *fuzzy*;

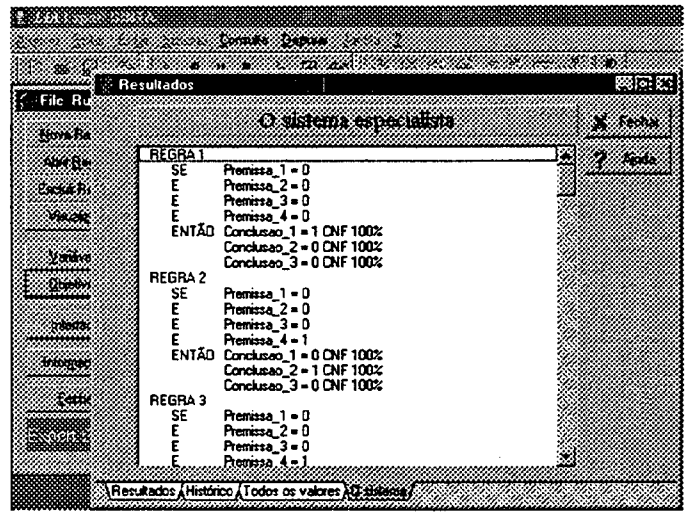

Figura 7.11: Tela do RBES: formalização das regras

4. O sistema neural foi desenvolvido usando-se a linguagem Visual Basic;

5. Faixa de valores das entradas/saídas do NNES *fuzzy* restrita em [-1, 1];

6. Função de ativação para cada neurônio escolhida como a tangente hiperbólica;

7. A faixa de valores usada para as conexões (pesos) do NNES *fuzzy* foram [-0,1; 0,1]. Estes valores foram escolhidos aleatoriamente;

8. Utilizou-se também o mecanismo de Têmpera Simulada (*Simulated Anneling*)[2] com valor igual a 1%.

7.2.1 Exemplo 1: Classificação dos Números de 0-15 em Pares, Primos e Ímpares

A seguir são mostradas algumas simulações para o Exemplo 1. Para esta aplicação estar-se-á se determinando apenas se o sistema conexionista aprende ou não. Não será testada a característica de generalização de um sistema conexionista devido a este exemplo não se aplicar ao caso da generalização.

[2]Em [157] a idéia de se utilizar a Têmpera Simulada é a de explorar suficientemente todo o espaço do problema logo no início do processo, para que a solução seja relativamente insensível ao estado inicial. Isto deve diminuir as chances de ficar preso em um máximo local, platô ou cordilheira. Contudo, sabe-se que não é garantido que o método ao ser utilizado, não encontre um novo máximo, ou mesmo um mínimo local, quando do seu uso.

A estrutura neural utilizada possui: camada de entrada = 4 neurônios, camada intermediária[3] = 16 neurônios e camada de saída = 3 neurônios. As Tabelas 7.1 e 7.2 apresentam vários valores simulados para as variáveis: Número de Gerações (NG), População Inicial (PI), Taxa de Crossover (TC), Taxa de Mutação (TM), Taxa de Aprendizado (α), Momento (β), Tolerância (T), Número de Épocas (NE), Número de Neurônios na Camada Intermediária da Rede Vencedora (N_{CIV}), Aptidão (A), Simulação (S), Padrão de Teste (PT), Porcentagem de Acertos (PA (%)), Número de Acertos ($N°A$). Para estas simulações foram usadas a distribuição gaussiana, bem como a Função de Aptidão (FA) definida como:

$$FA = \frac{1}{Erro\_Total + N_{CI}}$$

Tabela 7.1: Simulação 1: Exemplo 1 com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	O,1	0,01	0,1	0,05	250	3	0,0582
2	5	8	0,6	O,1	0,01	0,3	0,7	250	5	0,0548
3	5	8	0,6	0,1	0,01	0,4	0,2	250	6	0,0546
4	5	8	0,6	0,01	0,01	0,1	0,05	250	4	0,0556
5	5	8	0,6	0,01	0,01	0,3	0,7	250	4	0,057
6	5	8	0,6	0,01	0,01	0,4	0,2	250	6	0,0546
7	5	8	0,8	0,01	0,01	0,3	0,7	250	4	0,0576
8	5	8	0,8	0,1	0,01	0,1	0,05	250	4	0,0559
9	5	8	0,6	0,1	0,01	0,3	0,6	250	6	0,0523
10	5	8	0,6	0,1	0,01	0,3	0,7	500	7	0,0518
11	5	30	0,6	0,01	0,01	0,3	0,7	25	4	0,0572
12	10	30	0,6	0,01	0,01	0,3	0,7	2500	4	0,058

As Tabelas 7.3 e 7.4 mostram as simulações feitas para os mesmos dados mencionados anteriormente. Entretanto, agora a FA utilizada é:

[3]Nesta aplicação foram usados 16 neurônios na camada intermediária devido este valor definir o número de regras necessárias para descrever o problema em questão. Logo, o número de neurônios na camada intermediária foi definido em função do número de regras elicitadas.

Tabela 7.2: Simulação 1: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Quantidade de Padrões de Testes		
Simulações	PT=16	
S	$N°A$	PA (%)
1	13	81,25
2	14	87,50
3	14	87,50
4	14	87,50
5	14	87,50
6	14	87,50
7	14	87,50
8	14	87,50
9	12	75
10	12	75
11	14	87,50
12	14	87,50

$$FA = \frac{N_{CI}}{Erro\_Total}$$

Antes de se efetuar os comentários pertinentes as simulações apresentadas nas Tabelas 7.3 e 7.4, bem como para as demais simulações que envolvam a $FA = N_{CI}/Erro\_Total$, faz-se necessário definir melhor as variáveis PT1, PT2 e PT3. PT1 corresponde aos padrões de teste aplicados antes da rede vencedora ser minimizada, PT2 corresponde aos padrões de teste aplicados, após a rede vencedora ser minimizada para a diferença do N_{CI} do NNES inicial e o N_{CIV}, ficando a rede atual com o N_{CIV} com um valor menor que o do NNES inicial, e PT3 corresponde também aos padrões de teste depois que a rede vencedora é minimizada, mas agora a rede atual passa a ter o N_{CIV} igual ao do NNES inicial.

Como o número de neurônios obtido na camada intermediária da rede vencedora foi maior que o valor da rede inicial, a seguinte condição pôde ser testada: eliminando-se os neurônios e conexões da camada intermediária da rede vencedora até ao valor igual ao do NNES inicial, a rede em questão continuou a reconhecer os

Tabela 7.3: Simulação 2: Exemplo 1 com $FA = \frac{N_{CI}}{Erro\_Total}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	250	19	1,64
2	5	8	0,6	0,1	0,01	0,3	0,7	250	19	2,04
3	5	8	0,6	0,1	0,01	0,4	0,2	250	19	2,18
4	5	8	0,6	0,01	0,01	0,1	0,05	250	18	1,59
5	5	8	0,6	0,01	0,01	0,3	0,7	250	18	2,02
6	5	8	0,6	0,01	0,01	0,4	0,2	250	19	2,18
7	5	8	0,8	0,01	0,01	0,3	0,7	250	20	2,36
8	5	8	0,6	0,1	0,01	0,3	0,7	1000	19	2,28
9	5	8	0,6	0,01	0,01	0,1	0,05	1000	19	1,98
10	5	8	0,6	0,1	0,01	0,3	0,7	500	19	2,16
11	5	8	0,6	0,01	0,01	0,1	0,05	1000	19	1,98
12	5	8	0,6	0,1	0,01	0,3	0,7	500	19	2,15
13	5	20	0,6	0,01	0,01	0,3	0,7	2500	20	2,28

padrões de teste mesmo quando passou a ter o Número de Neurônios da Camada Intermediária (N_{CI}) igual a diferença entre o N_{CI} do NNES inicial e o da rede vencedora. Este caso pode ser observado na Tabela 7.4 para o conjunto de teste PT2. Isto se deve ao fato que a função de aptidão utilizada para este exemplo caracterizou-se por mapear a rede vencedora com o N_{CI} maior que ao do NNES inicial. Uma outra condição também foi testada para esta aplicação: eliminando-se agora os neurônios e conexões excedentes da rede vencedora, comparados com o N_{CI} da rede inicial, deixando-se a rede vencedora atual apenas com o N_{CI} igual ao da rede inicial, esta rede continuou a reconhecer os padrões de teste. Isto pode ser observado na Tabela 7.4 para o conjunto de teste PT3.

As Tabelas 7.5 e 7.6 mostram as simulações feitas para os mesmos dados mencionados anteriormente. Contudo, o que foi alterado diz respeito a utilização da distribuição ser uniforme e a função de aptidão ser:

$$FA = \frac{1}{Erro\_Total + N_{CI}}$$

Tabela 7.4: Simulação 2: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$

Quantidade de Padrões de Testes						
Simulações	PT1=16		PT2=16		PT3=16	
S	$N°A$	%	$N°A$	%	$N°A$	%
1	14	87,50	5	31,25	13	81,25
2	16	100	3	25	14	87,50
3	16	100	7	43,75	16	100
4	14	87,50	8	50	12	75
5	16	100	12	75	15	93,75
6	16	100	7	43,75	16	100
7	16	100	7	43,75	12	75
8	16	100	10	62,50	16	100
9	16	100	8	50	12	75
10	16	100	14	87,5	15	93,75
11	16	100	8	50	12	75
12	16	100	14	87,5	15	97,75
13	15	93,75	6	37,5	15	97,75

As Tabelas 7.7 e 7.8 mostram as simulações feitas para os mesmos dados mencionados anteriormente. Todavia, o que foi alterado diz respeito a utilização da distribuição ser gaussiana, a *bias* com valor de +1 para a camada intermediária e -1 para a camada de saída, e a função de aptidão ser:

$$FA = \frac{1}{Erro\_Total + N_{CI}}$$

As Tabelas 7.9 e 7.10 mostram as simulações feitas para os mesmos dados mencionados anteriormente. Contudo, somente o que foi alterado diz respeito a função de aptidão ser:

$$FA = \frac{N_{CI}}{Erro\_Total}$$

Tabela 7.5: Simulação 3: Exemplo 1 com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	250	8	0,048
2	5	8	0,6	0,1	0,01	0,3	0,7	250	10	0,046
3	5	8	0,6	0,01	0,01	0,1	0,05	250	8	0,048
4	5	8	0,6	0,01	0,01	0,3	0,7	250	10	0,049

Tabela 7.6: Simulação 3: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Quantidade de Padrões de Testes		
Simulações	PT=16	
S	$N^{o}A$	%
1	14	87,5
2	14	87,5
3	14	87,5
4	14	87,5

7.2.2 Resultados do Exemplo 1

- Observando-se as Tabelas 7.1 e 7.2, a maioria dos padrões apresentados à rede vencedora foram reconhecidos, mesmo com um número reduzido de neurônios na camada intermediária;

- Visualizando as Tabelas 7.3 e 7.4, nota-se que vários casos ocorreram em que uma boa parte dos padrões de teste apresentados à RNA, em questão, foram reconhecidos. O que se pode deduzir, também, destas tabelas é que o conhecimento que antes estava localizado na rede inicial passou a estar distribuído não só nas conexões anteriores, bem como nas novas. Nota-se, por exemplo, que ao se apresentar os padrões de teste PT2, quando o $N_{CIV} = N_{CIV} - N_{CI}$ da rede inicial, que vários casos atingiram uma percentagem razoável de acertos, ou melhor, de reconhecimento de padrões. Isto quer dizer que a RNA continuou reconhendo os referidos padrões apenas com um número mínimo de neurônios na camada intermediária da rede atual. Já para os padrões de teste PT3, quando $N_{CIV} = N_{CI}$ da rede inicial, a percentagem foi superior ao caso

Tabela 7.7: Simulação 4: Exemplo 1 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ e *bias*

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	250	7	0,049
2	5	8	0,6	0,1	0,01	0,3	0,7	250	6	0,054
3	5	8	0,6	0,01	0,01	0,1	0,05	250	7	0,049
4	5	8	0,6	0,01	0,01	0,3	0,7	250	6	0,054

Tabela 7.8: Simulação 4: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$ e *bias*

Quantidade de Padrões de Testes		
Simulações	PT=16	
S	$N^{o}A$	%
1	7	43,75
2	8	50
3	1	6,25
4	4	25

anterior;

- As Tabelas 7.5 e 7.6 mostram também que, mesmo se utilizando a distribuição uniforme na criação da população de indivíduos (redes), a rede vencedora reconheceu um número grande de padrões;

Tabela 7.9: Simulação 5: Exemplo 1 com $FA = \frac{N_{CI}}{Erro\_Total}$ e *bias*

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	250	20	2,046
2	5	8	0,6	0,1	0,01	0,3	0,7	250	20	2,14
3	5	8	0,6	0,01	0,01	0,1	0,05	250	20	2,14
4	5	8	0,6	0,01	0,01	0,3	0,7	250	20	2,14

Tabela 7.10: Simulação 5: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$ e *bias*

Quantidade de Padrões de Testes				
Simulações	PT1=16		PT2=16	
S	N^oA	%	N^oA	%
1	13	81,25	13	81,25
2	13	81,25	13	81,25
3	13	81,25	13	81,25
4	13	81,25	13	81,25

- Das Tabelas 7.7 à 7.10, observa-se que ao se colocar a *bias* com os valores lá referenciados, o índice de reconhecimento da maiora dos padrões foi baixo, comparado aos apresentados nas tabelas anteriores. Neste exemplo foram aplicados valores alternativos para as *bias*: na camada intermediária +1 ou -0,5 e na da saída -1 ou -0,5. Contudo, estes também não apresentaram bons resultados.

7.2.3 Exemplo 2: Classificação das Vogais em Código ASCII

A seguir são mostradas algumas simulações para o Exemplo 2. Para tanto estará se determinando se o sistema conexionista, em questão, aprende ou não, e se este tem a capacidade de generalização.

A estrutura neural utilizada tem: camada de entrada = 35 neurônios, camada intermediária[4]= 5 neurônios e camada de saída = 7 neurônios. As Tabelas 7.11 e 7.12 apresentam vários valores simulados para as variáveis: Número de Gerações (NG), População Inicial (PI), Taxa de Crossover (TC), Taxa de Mutação (TM), Taxa de Aprendizado (α), Momento (β), Tolerância (T), Número de Épocas (NE), Número de Neurônios na Camada Intermediária Vencedora (N_{CIV}), Aptidão (A), Simulação (S), Padrão de Teste (PT1, PT2, PT3), Padrão de Teste com Ruído (PRuido), Porcentagem de Acertos (PA (%)), Número de Acertos ($N°A$). Para estas simulações foi usada a distribuição gaussiana, bem como a Função de Aptidão (FA) definida como:

$$FA = \frac{1}{Erro\_Total + N_{CI}}$$

As Tabelas 7.13 e 7.14 mostram as simulações feitas para os mesmos dados mencionados anteriormente. Entretanto, o que foi alterado refere-se a utilização da distribuição ser uniforme e a função de aptidão ser:

$$FA = \frac{1}{Erro\_Total + N_{CI}}$$

As Tabelas 7.15 e 7.16 mostram as simulações feitas para os mesmos dados mencionados anteriormente. O que foi alterado diz respeito: a utilização da distribuição ser gaussiana, a *bias* com valor de +1 para a camada intermediária e -1 para a camada de saída, e a função de aptidão ser:

$$FA = \frac{1}{Erro\_Total + N_{CI}}$$

[4]Na RNA desenvolvida para esta aplicação foram utilizados 5 neurônios na camada intermediária, isto se deve ao fato de que o número de regras definidas para este exemplo ser somente 5.

Tabela 7.11: Simulação 6: Exemplo 2 com $FA = \frac{1}{Erro\_Total + N_{CI}}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	250	3	0,0792
2	5	8	0,6	0,1	0,01	0,3	0,7	500	4	0,0796
3	5	8	0,6	0,1	0,01	0,4	0,2	500	4	0,077
4	5	8	0,6	0,01	0,01	0,1	0,05	500	4	0,772
5	5	8	0,6	0,01	0,01	0,3	0,7	500	4	0,0795
6	5	8	0,6	0,01	0,01	0,4	0,2	500	3	0,0864
7	5	8	0,8	0,01	0,01	0,3	0,7	500	4	0,0828
8	5	8	0,8	0,1	0,01	0,3	0,7	1000	3	0,0813
9	5	8	0,6	0,1	0,01	0,3	0,7	250	3	0,0852
10	5	8	0,6	0,01	0,01	0,3	0,7	1000	3	0,0813
11	5	20	0,6	0,01	0,01	0,3	0,7	2500	5	0,0814

As Tabelas 7.17 e 7.18 mostram as simulações feitas para os mesmos dados mencionados no início desta aplicação. Entretanto, somente no que se refere a utilização da função de aptidão, houve alteração:

$$FA = \frac{N_{CI}}{Erro\_Total}$$

Como no Exemplo 1, Tabelas 7.3 e 7.4, foram realizados testes para a minimização da rede com $N_{CIV} = N_{CIV} - N_{CI}$ do NNES inicial e $N_{CIV} = N_{CI}$ do NNES inicial. Veja as Tabelas 7.17 e 7.18.

As Tabelas 7.19 e 7.20 mostram as simulações feitas para os mesmos dados mencionados anteriormente. A alteração ocorrida diz respeito a utilização da distribuição ser uniforme e a função de aptidão ser:

$$FA = \frac{N_{CI}}{Erro\_Total}$$

Tabela 7.12: Simulação 6: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Quantidade de Padrões de Testes				
Simulações	PT=5		PRuido=5	
S	N^oA	PA (%)	N^oA	PA (%)
1	1	20	1	20
2	-	0	-	0
3	-	0	-	0
4	1	20	1	20
5	-	0	-	0
6	-	0	-	0
7	2	40	2	40
8	-	0	-	0
9	2	40	2	40
10	-	0	-	0
11	3	60	3	60

As Tabelas 7.21 e 7.22 mostram as simulações feitas para os mesmos dados mencionados anteriormente. Entretanto foi alterado a utilização da distribuição para ser gaussiana, o uso da *bias*, com valor de +1 para a camada intermediária e -1 para a camada de saída, e a função de aptidão ser:

$$FA = \frac{N_{CI}}{Erro\_Total}$$

Tabela 7.13: Simulação 7: Exemplo 2 com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	500	4	0,076
2	5	8	0,6	0,1	0,01	0,3	0,7	500	4	0,078
3	5	8	0,6	0,01	0,01	0,1	0,05	500	4	0,074
4	5	8	0,6	0,01	0,01	0,3	0,7	500	4	0,077

Tabela 7.14: Simulação 7: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Quantidade de Padrões de Testes		
Simulações	PT=5	
S	$N°A$	%
1	1	20
2	-	0
3	1	20
4	-	0

Tabela 7.15: Simulação 8: Exemplo 2 com $FA = \frac{1}{Erro\_Total+N_{CI}}$ e *bias*

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	500	3	0,076
2	5	8	0,6	0,01	0,01	0,1	0,05	500	3	0,076
3	5	8	0,6	0,01	0,01	0,3	0,7	500	3	0,081

7.2.4 Resultados do Exemplo 2

- As Tabelas 7.11 à 7.16 demostram que a FA utilizada não foi uma das melhores escolhas para definir o problema abordado, mesmo se optando pela distribuição uniforme e com *bias*. Entretanto, exceto para a simulação 11 das Tabelas 7.11 e 7.12, onde as condições foram mais favoráveis, ou seja, aumentando-se o número de redes (indivíduos) a serem criadas em cada geração e, também, o número de iterações a cada ciclo de treinamento à cada rede gerada, os resultados foram melhores;

- Os resultados apresentados nas Tabelas 7.13 à 7.16 indicam que a RNA não aprendeu o suficiente para fazer uma boa classificação de padrões. Os resultados não foram satisfatórios;

- As Tabelas 7.17 e 7.18 demostram que para a FA em questão, o conjunto de teste apresentado à rede vencedora alcançou valores aceitáveis de reconhecimento dos padrões, bem como de generalização, principalmente ao ser apresentado o conjunto de padrões de teste PT1 e PRuido. Quanto aos casos em que $N_{CIV} = N_{CIV} - N_{CI}$ da rede inicial e $N_{CIV} = N_{CI}$ da rede inicial, ambos

Tabela 7.16: Simulação 8: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$ e *bias*

Quantidade de Padrões de Testes		
Simulações	PT=5	
S	N^oA	%
1	-	0
2	-	0
3	-	0

Tabela 7.17: Simulação 9: Exemplo 2 com $FA = \frac{N_{CI}}{Erro\_Total}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	500	10	0,992
2	5	8	0,6	0,1	0,01	0,3	0,7	500	9	1,46
3	5	8	0,6	0,1	0,01	0,4	0,2	500	9	1,14
4	5	8	0,6	0,01	0,01	0,1	0,05	500	10	1,13
5	5	8	0,6	0,01	0,01	0,3	0,7	500	8	1,13
6	5	8	0,6	0,01	0,01	0,4	0,2	500	9	1,43
7	5	8	0,8	0,01	0,01	0,3	0,7	500	10	1,37
8	5	30	0,6	0,01	0,01	0,3	0,7	2500	10	1,64
9	5	20	0,6	0,01	0,01	0,3	0,7	2500	10	1,47

tiveram uma gama de valores condizentes de reconhecimento e generalização para o caso em estudo;

- Para as Tabelas 7.19 e 7.20, usando-se a distribuição uniforme, obteve-se valores aceitáveis para o reconhecimento e generalização do conjunto de padrões apresentados à rede vencedora. Quanto ao uso da *bias*, a quantidade de padrões a ser reconhecido foram inferiores aos casos apresentados anteriormente, como podem ser visualizados pelas Tabelas 7.21 e 7.22.

Tabela 7.18: Simulação 9: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$

Quantidade de Padrões de Testes												
	PT1=5		PRuido=5		PT2=5		PRuido=5		PT3=5		PRuido=5	
S	$N°A$	%	$N°A$	%	$N°A$	%	$N°A$	%	$N°A$	%	$N°A$	%
1	2	40	2	40	1	20	1	20	3	60	3	60
2	5	100	4	80	2	40	2	40	4	80	3	60
3	4	80	3	60	2	40	2	40	2	40	2	40
4	4	80	3	60	3	60	4	80	1	20	-	0
5	5	100	4	80	3	60	3	60	2	40	2	40
6	4	80	4	80	2	40	2	40	2	40	2	40
7	4	80	3	60	2	40	3	60	2	40	1	20
8	5	100	5	100	2	40	2	40	3	60	3	60
9	5	100	5	100	2	40	2	40	3	60	3	60

Tabela 7.19: Simulação 10: Exemplo 2 com $FA = \frac{N_{CI}}{Erro\_Total}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	500	7	0,762
2	5	8	0,6	0,01	0,01	0,1	0,05	500	7	0,959
3	5	8	0,6	0,01	0,01	0,3	0,7	500	7	0,757

7.2.5 Exemplo 3: Classificação de Crises Epilépticas

Para esta aplicação serão apresentadas algumas simulações, bem como, também se estará determinando se o sistema conexionista em questão aprende ou não, e se este tem a capacidade de generalização. Para se provar as características mencionadas anteriormente a uma RNA, serão utilizados casos idealizados pelo próprio médico, além daqueles adquiridos através de prontuários da referida doença.

Estudo de Casos Reais

Os casos aqui ilustrados são três: o primeiro refere-se ao idealizado pelo próprio médico; o segundo e o terceiro são aqueles adquiridos através de prontuários. A área de estudo destes casos é a da classificação de crises epilépticas. A seguir, cada um

Tabela 7.20: Simulação 11: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$

Quantidade de Padrões de Testes		
Simulações	PT=5	
S	$N^{o}A$	%
1	3	60
2	4	80
3	3	60

Tabela 7.21: Simulação 12: Exemplo 2 com $FA = \frac{N_{CI}}{Erro\_Total}$ e *bias*

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	500	10	1,499
2	5	8	0,6	0,01	0,01	0,1	0,05	500	7	0,794
3	5	8	0,6	0,01	0,01	0,3	0,7	500	9	1,46

destes casos será analisado.

Para o primeiro caso foram obtidos 6 sintomas: alteração de consciência, automatismo, EEG com alteração focal, movimento tônico, movimento tônico-clônico e movimento clônico. Obteve-se também, três tipos de classificação sobre as crises epilépticas (diagnóstico), ou seja, crise parcial simples, crise parcial complexa e crise parcial secundariamente generalizada.

Tabela 7.22: Simulação 12:Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$ e *bias*

Quantidade de Padrões de Testes		
Simulações	PT=5	
S	$N^{o}A$	%
1	3	60
2	3	60
3	3	60

A estrutura neural utilizada possui: camada de entrada (sintomas) = 6 neurônios, camada intermediária[5]= 6 neurônios e camada de saída (diagnósticos) = 3 neurônios. As Tabelas 7.23 e 7.24 apresentam vários valores simulados para as variáveis: Número de Gerações (NG), População Inicial (PI), Taxa de Crossover (TC), Taxa de Mutação (TM), Taxa de Aprendizado (α), Momento (β), Tolerância (T), Número de Épocas (NE), Número de Neurônios na Camada Intermediária Vencedora (N_{CIV}), Aptidão (A), Simulação (S), Padrão de Teste (PT1, PT2, PT3, PT4, PT5, PT6), Porcentagem de Acertos (PA (%)), Número de Acertos (N^oA). Para estas simulações foi usada a distribuição gaussiana, bem como a Função de Aptidão (FA), assim definida:

$$FA = \frac{1}{Erro\_Total + N_{CI}}$$

Tabela 7.23: Simulação 13: Exemplo 3 com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	250	4	0,125
2	5	8	0,6	0,1	0,01	0,3	0,7	250	4	0,127
3	5	8	0,6	0,1	0,01	0,4	0,2	250	4	0,134
4	5	8	0,6	0,01	0,01	0,1	0,05	250	4	0,132
5	5	8	0,6	0,01	0,01	0,3	0,7	250	4	0,134
6	5	8	0,6	0,01	0,01	0,4	0,2	250	5	0,125
7	5	8	0,8	0,01	0,01	0,3	0,7	250	4	0,123
8	10	8	0,6	0,01	0,01	0,1	0,05	500	4	0,118
9	5	10	0,6	0,01	0,01	0,1	0,05	1500	4	0,136
10	5	10	0,6	0,01	0,01	0,3	0,7	1500	4	0,136

As Figuras 7.12 e 7.13 mostram que a qualidade da rede é melhorada. Neste caso, o valor do Coeficiente de Variação Relativa (CVR), isto é, o Desvio Padrão (DV) e o Valor Médio (VM), dado por $CVR = DV/VM$, diminuiu, enquanto a aptidão aumentou. Veja as Tabelas 7.23 e 7.24, Simulações 1 e 4.

[5]Na RNA desenvolvida para esta aplicação foram utilizados 6 neurônios na camada intermediária, isto se deve ao fato de que o número de regras definidas para este exemplo ser somente 6.

Tabela 7.24: Simulação 13: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Quantidade de Padrões de Testes		
Simulações	PT1=6	
S	$N°A$	PA (%)
1	3	50
2	3	50
3	3	50
4	5	83,33
5	3	50
6	5	83,33
7	3	50
8	4	66,66
9	5	83,33
10	5	83,33

As Tabelas 7.25 e 7.26 mostram as simulações feitas para os mesmos dados mencionados anteriormente. Contudo foi alterado a função de aptidão para:

$$FA = \frac{N_{CI}}{Erro\_Total}$$

Aqui também foram testadas as condições: $N_{CIV} = N_{CIV} - N_{CI}$ do NNES inicial e $N_{CIV} = N_{CI}$ do NNES inicial. Veja as Tabelas 7.25 e 7.26.

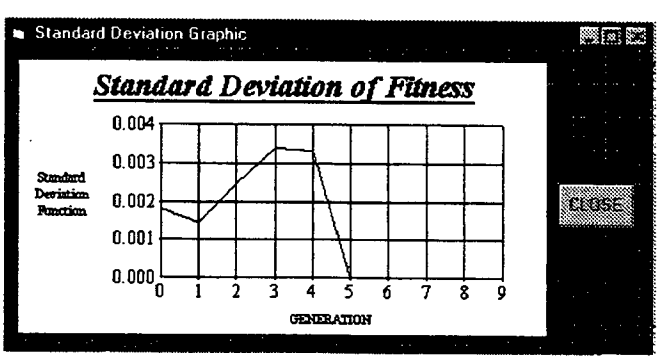

Figura 7.12: Análise do desvio padrão: aptidão menor

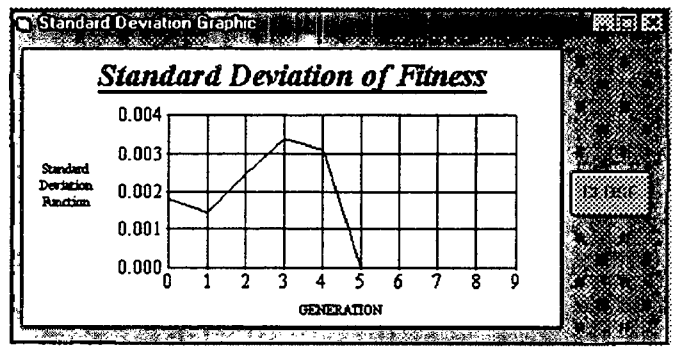

Figura 7.13: Análise do desvio padrão: aptidão maior

Tabela 7.25: Simulação 14: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	250	10	3,111
2	5	8	0,6	0,1	0,01	0,3	0,7	250	10	3,362
3	5	8	0,6	0,1	0,01	0,4	0,2	250	10	3,81
4	5	8	0,6	0,01	0,01	0,1	0,05	250	10	3,06
5	5	8	0,6	0,01	0,01	0,3	0,7	250	10	3,88
6	5	8	0,6	0,01	0,01	0,4	0,2	250	10	3,76
7	5	8	0,8	0,01	0,01	0,3	0,7	250	10	3,34
8	10	8	0,6	0,01	0,01	0,1	0,05	500	10	2,91
9	5	10	0,6	0,01	0,01	0,1	0,05	1500	11	4,18
10	5	10	0,6	0,01	0,01	0,3	0,7	1500	10	3,10

As próximas simulações correspondem ao segundo e ao terceiro caso de estudo. Nesta etapa são apresentados dois conjuntos distintos de casos reais obtidos através de um conjunto de prontuários médicos para formar o NNES inicial. A estrutura neural usada para o primeiro tem: camada de entrada (sintomas) = 32 neurônios, camada intermediária[6]= 9 neurônios e camada de saída (diagnósticos) = 4 neurônios.

[6]Na RNA desenvolvida para esta aplicação foram utilizados 9 neurônios na camada intermediária, isto se deve ao fato de que o número de regras definidas para este exemplo ser somente 9.

Tabela 7.26: Simulação 14: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$

Quantidade de Padrões de Testes						
Simulações	PT1=6		PT2=6		PT3=6	
S	$N°A$	%	$N°A$	%	$N°A$	%
1	6	100	3	50	4	66,67
2	6	100	3	50	5	83,33
3	6	100	3	50	5	83,33
4	6	100	4	66,67	5	83,33
5	6	100	5	83,33	5	83,33
6	6	100	5	83,33	4	66,67
7	6	100	2	33,33	5	83,33
8	5	83,33	5	83,33	5	83,33
9	6	100	6	100	5	83,33
10	6	100	3	50	5	83,33

Veja as Tabelas 7.27, 7.28, 7.31 e 7.32. Para o segundo, a estrutura neural usada foi: camada de entrada (sintomas) = 32 neurônios, intermediária[7] = 11 neurônios e camada de saída (diagnósticos) = 4 neurônios. Veja as Tabelas 7.29, 7.30, 7.33 e 7.34. Ambos os exemplos, do ponto de vista das tabelas, apresentam vários valores para as variáveis já mencionadas no caso anterior desta aplicação.

[7]Na RNA desenvolvida para esta aplicação foram utilizados 11 neurônios na camada intermediária, isto se deve ao fato de que o número de regras definidas para este exemplo ser somente 11.

Os sintomas (neurônios de entrada do NNES inicial) utilizados para as implementações e simulações para esta etapa foram os seguintes:

- Abalos focais;
- Alteração da consciência;
- Alteração da linguagem;
- Alteração quantitativa da consciência;
- Alteração de sensibilidade;
- Alucinações ópticas;
- Alucinações olfativas;
- Alucinações gustativas;
- Alucinações auditivas;
- Amnésia;
- Automatismo;
- Cianose;
- Dores musculares Pós-ictal;
- Duração da perda de consciência;
- EEG com alteração focal;
- EEG com alteração generalizada;
- Incontinência esfincteriana;
- Intensidade da dor de cabeça;
- Manifestações vegetativas;
- Mordedura da língua;
- Monoparalisia;
- Movimento clônico;
- Movimento tônico;

- Movimento tônico-clônico;
- Palidez;
- Perda da consciência;
- Queda;
- Queda com machucadura;
- Sintomas afetivos;
- Sinais focais pós-crise;
- Sonolência pós-crise;
- Versão da cabeça.

As classificações dos tipos de crises epilépticas (neurônios de saída do NNES inicial) utilizadas para as implementações e simulações para esta etapa foram:

- Crise parcial simples;
- Crise parcial complexa;
- Crise parcial secundariamente generalizada;
- Crise generalizada tônico-clônica.

Para estas simulações foi usada a distribuição gaussiana, bem como a Função de Aptidão (FA) definida como:

$$FA = \frac{1}{Erro\_Total + N_{CI}}$$

Tabela 7.27: Simulação 15: Exemplo 3 com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,01	0,01	0,3	0,7	250	8	0,065
2	10	8	0,6	0,01	0,01	0,3	0,7	500	8	0,0658
3	10	8	0,6	0,01	0,01	0,1	0,05	500	8	0,0652
4	5	8	0,6	0,01	0,01	0,1	0,05	250	8	0,0643
5	5	20	0,6	0,01	0,01	0,1	0,05	250	8	0,0643
6	5	10	0,6	0,01	0,01	0,1	0,05	1500	9	0,062
7	5	20	0,6	0,01	0,01	0,3	0,7	1500	8	0,066
8	5	20	0,6	0,1	0,01	0,1	0,05	2000	8	0,066

As Tabelas 7.31 à 7.34 mostram as simulações feitas para os mesmos dados mencionados anteriormente. Contudo foi alterado a função de aptidão:

$$FA = \frac{N_{CI}}{Erro\_Total}$$

Aqui também foram testadas as condições: $N_{CIV} = N_{CIV} - N_{CI}$ do NNES inicial e $N_{CIV} = N_{CI}$ do NNES inicial. Veja as Tabelas 7.31 à 7.34. Além disso, considerar os padrões de testes como sendo para o caso aplicado de $PT3 = PT5 = N_{CIV} - N_{CI}$ da rede inicial e $PT4 = PT6 = N_{CIV} = N_{CI}$ da rede inicial.

7.2.6 Resultados do Exemplo 3

- Visualizando as Tabelas 7.23 e 7.24 é possível se observar que para a FA escolhida muitos padrões deixaram de ser reconhecidos. Um dos motivos pode ter sido uma má escolha da FA;

Tabela 7.28: Simulação 15: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Quantidade de Padrões de Testes				
Simulações	PT1=9		PT2=7	
S	$N^{o}A$	%	$N^{o}A$	%
1	6	66,67	4	57,14
2	6	66,67	4	57,14
3	6	66,67	4	57,14
4	6	66,67	4	57,14
5	6	66,67	4	57,14
6	7	77,78	5	71,43
7	6	66,67	4	57,14
8	6	66,67	4	57,14

- Através das Tabelas 7.25 e 7.26 demostram-se que os valores obtidos foram melhores que ao do caso da aplicação anterior, desta forma foi possível se fazer uma classificação adequada ao caso idealizado de crises epilépticas. Isto se deve a uma escolha melhor da FA;

- Nas Tabelas 7.23 à 7.26, observa-se que o número mínimo de neurônios obtidos na camada intermediária da rede vencedora foi de 4, pois foi possível testar a hipótese de que a rede vencedora não deveria ter o N_{CI} menor que os sintomas prevalentes para uma dada doença, isto é, acredita-se que se a rede vencedora tivesse um N_{CI} menor que estes sintomas prevalentes ela não conseguiria classificar uma dada doença. Isto foi testado e realmente ocorreu. Contudo, quando foi testada a condição de que o N_{CI} da rede vencedora não deveria diminuir mais que os sintomas prevalentes de uma dada doença, esta foi capaz de classificar (diagnosticar) a mesma com um maior grau de certeza, como pode ser visto nas referidas tabelas;

Tabela 7.29: Simulação 16: Exemplo 3 com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,01	0,01	0,1	0,05	250	7	0,048
2	5	8	0,6	0,1	0,01	0,1	0,05	250	7	0,057
3	5	8	0,6	0,1	0,01	0,3	0,7	250	7	0,057
4	5	8	0,6	0,1	0,01	0,3	0,7	1000	7	0,057
5	5	8	0,6	0,1	0,01	0,3	0,7	2000	7	0,051
6	5	15	0,6	0,1	0,01	0,3	0,7	2000	9	0,050
7	5	15	0,6	0,1	0,01	0,3	0,7	2500	9	0,051

- Como pode ser observado nas Tabelas 7.27 e 7.28, devido a FA escolhida, os resultados obtidos foram baixos para os padrões de testes apresentados à rede vencedora;

- Para as Tabelas 7.29 e 7.30, devido também a FA escolhida, os resultados obtidos foram baixos para os padrões de testes apresentados à rede vencedora;

- Para as Tabelas 7.31 e 7.32 os resultados obtidos, após a apresentação dos padrões de testes à rede vencedora, foram melhores. Foi possível se diagnosticar (classificar) com maior precisão a doença correspondente aos sintomas apresentados à referida rede;

- Como no caso anterior, veja as Tabelas 7.27 e 7.28, foi testada a condição de que a rede vencedora não deveria ter o N_{CI} menor que os sintomas prevalentes para uma dada doença. Neste caso, o número de sintomas prevalentes foi de 8, correspondendo a oito neurônios na camada intermediária da rede vencedora. Isto pode ser observado nas tabelas já referenciadas anteriormente. Contudo, quando foi testada a condição de que o N_{CI} da rede vencedora não deveria diminuir mais que os sintomas prevalentes de uma dada doença, a rede vencedora ainda foi capaz de classificá-la (diagnosticar) com um dado grau de certeza;

Tabela 7.30: Simulação 16: Reconhecimento de Padrões com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Quantidade de Padrões de Testes				
Simulações	PT1=11		PT2=11	
S	$N°A$	%	$N°A$	%
1	-	0	-	0
2	4	36,36	4	36,36
3	4	36,36	4	36,36
4	6	54,55	6	54,55
5	6	54,55	6	54,55
6	4	36,36	4	36,36
7	1	9,09	1	9,09

- Visualizando as Tabelas 7.33 e 7.34, pode-se observar que os resultados foram um pouco mais satisfatórios;
- Como no caso anterior, veja as Tabelas 7.29 e 7.30, foi também testada a condição de que a rede vencedora não deveria ter o N_{CI} menor que os sintomas prevalentes para uma dada doença. Neste caso, o número de sintomas prevalentes foi de 7, correspondendo a sete neurônios na camada intermediária da rede vencedora. Contudo, quando foi testada a condição de que o N_{CI} da rede vencedora não deveria diminuir mais que os sintomas prevalentes de uma dada doença, a rede vencedora foi ainda capaz de classificá-la (diagnosticar), mas não com uma percentagem maior que 55%.

Tabela 7.31: Simulação 17: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,01	0,01	0,3	0,7	250	13	2,10
2	10	8	0,6	0,01	0,01	0,3	0,7	500	13	2,11
3	5	8	0,6	0,01	0,01	0,1	0,05	500	13	1,92
4	5	8	0,6	0,01	0,01	0,1	0,05	500	13	1,65
5	5	20	0,6	0,01	0,01	0,1	0,05	500	14	1,78
6	5	8	0,6	0,01	0,01	0,3	0,7	500	14	2,15
7	5	10	0,6	0,01	0,01	0,1	0,05	1500	13	2,07
8	5	10	0,6	0,01	0,01	0,3	0,7	1500	13	2,07
9	5	20	0,6	0,01	0,01	0,1	0,05	1500	14	2,51

7.3 Simulações para o RBES

Dado um RBES com as seguintes características:

1. Este sistema simbólico foi implementado usando-se o *Shell Expert* SINTA[8];
2. O processo de busca utilizado para a máquina de inferência foi o encadeamento para trás;
3. Foi desenvolvido, utilizando a linguagem Delphi, um programa auxiliar interfaceado com o *Shell Expert* SINTA para transformar os dados obtidos na saída do NNES na forma de um conjunto de regras *Se/Então*;
4. Este sistema servirá como um *validador* do conhecimento obtido na saída do NNES.

Para se validar este sistema simbólico serão aplicados os mesmos Exemplos 1 (Classificação dos Números Cardinais), 2 (Classificação das Vogais) e 3 (Casos Reais

[8]A idéia original de dois anos e meio atrás para o desenvolvimento deste trabalho previa a integração de todo o sistema. No entanto, conforme a crítica da banca de qualificação no sentido da proposta ser por demais extensa, resolveu-se dar prioridade ao estudo das metodologias, onde se encontra a originalidade do trabalho em detrimento da interligação dos diversos blocos. Neste sentido foram utilizados alguns sistemas já existentes, como o *Expert* SINTA, de forma a se poupar tempo. A interligação poderia ficar para um trabalho num nível inferior (M.Sc. ou ICI, dependendo das especificações) posto não apresentar, *a priori*, originalidade.

Tabela 7.32: Simulação 17: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$

Quantidade de Padrões de Testes												
	PT1=9		PT2=7		PT3=9		PT4=9		PT5=7		PT6=7	
S	$N°A$	%	$N°A$	%	$N°A$	%	$N°A$	%	$N°A$	%	$N°A$	%
1	8	88,9	4	57,2	3	33,3	6	66,7	3	42,9	4	57,2
2	8	88,9	4	57,2	3	33,3	6	66,7	1	14,3	4	57,2
3	8	88,9	6	85,7	3	33,3	6	66,7	3	42,9	3	42,9
4	6	66,7	4	57,2	-	0	6	66,7	4	57,2	-	0
5	8	88,9	6	85,7	7	77,8	-	0	6	85,7	-	0
6	7	77,8	5	71,4	7	77,8	-	0	5	71,4	-	0
7	8	88,9	6	85,7	8	88,9	-	0	6	85,7	-	0
8	8	88,9	4	57,2	9	100	-	0	6	85,7	-	0
9	9	100	7	100	3	33,3	6	66,7	3	33,3	4	57,2

de Crises Epilépticas) utilizados para desenvolver, refinar, testar e validar o NNES. Obviamente foram escolhidas as melhores simulações ocorridas para o NNES, ou seja, aquelas com um índice de acerto[9] maior ou igual a 60%.

[9]Sabe-se que é difícil encontrar um SE implementado para a área médica com um índice de acerto muito alto. Portanto, em função do que se conhece da área, considera-se que um ídice de 60% já pode ser considerado como razoável.

Tabela 7.33: Simulação 18: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$

Dados										
S	NG	PI	TC	TM	T	α	β	NE	Nciv	A
1	5	8	0,6	0,1	0,01	0,1	0,05	250	15	1,45
2	5	8	0,6	0,1	0,01	0,3	0,7	500	16	1,68
3	5	8	0,6	0,1	0,01	0,1	0,05	1000	16	1,78
4	5	8	0,6	0,1	0,01	0,3	0,7	1000	16	1,76
5	5	8	0,6	0,1	0,01	0,3	0,7	1250	16	1,65
6	5	20	0,6	0,01	0,01	0,3	0,7	2000	16	1,59
7	5	20	0,6	0,1	0,01	0,1	0,05	1500	16	1,79
8	10	20	0,6	0,1	0,01	0,1	0,05	5000	16	1,73

7.3.1 Exemplos

Para todos os exemplos aqui utilizados as seguintes abreviaturas são válidas: Regras Geradas Com Redundâncias (CR), Regras Geradas Sem Redundâncias (SR). As demais abreviaturas que se apresentam aqui são as mesmas utilizadas nas seções anteriores.

Exemplo 1: Números Cardinais (0 à 15)

Exemplo 2: Vogais

Exemplo 3: Casos Reais - Crises Epilépticas

Para as Tabelas 7.39 e 7.41, o N_{CI} da rede inicial é de 6 neurônios.

Tabela 7.34: Simulação 18: Reconhecimento de Padrões com $FA = \frac{N_{CI}}{Erro\_Total}$

Quantidade de Padrões de Testes												
	PT1=11		PT2=11		PT3=11		PT4=11		PT5=11		PT6=11	
S	$N°A$	%	$N°A$	%	$N°A$	%	$N°A$	%	$N°A$	%	$N°A$	%
1	9	81,8	8	72,7	3	27,3	3	27,3	7	63,6	7	63,6
2	6	54,6	7	63,6	6	54,6	3	27,3	4	36,4	4	36,4
3	7	63,6	6	54,6	6	54,6	6	54,6	1	9,1	1	9,1
4	6	54,6	5	45,5	6	54,6	6	54,6	-	0	-	0
5	7	63,6	7	63,6	3	27,3	1	9,1	7	63,6	6	54,6
6	6	54,6	5	45,5	1	9,1	1	9,1	4	36,4	4	36,4
7	8	72,7	8	72,7	1	9,1	1	9,1	7	63,6	7	63,6
8	6	54,6	5	45,5	3	27,3	3	27,3	4	36,4	4	36,4

Tabela 7.35: Simulação 19: Exemplo 1 com $FA = \frac{1}{Erro\_Total + N_{CI}}$

Dados												
S	NG	PI	TC	TM	T	α	β	NE	Nciv	CR	SR	$N°A$	PA (%)
3	5	8	0,6	0,1	0,01	0,4	0,2	250	6	68	23	14	87,5
5	5	8	0,6	0,01	0,01	0,3	0,7	250	4	38	17	10	62,5
6	5	8	0,6	0,01	0,01	0,4	0,2	250	6	68	23	14	87,5
10	5	8	0,6	0,1	0,01	0,3	0,7	500	7	89	21	14	87,5
12	10	30	0,6	0,01	0,01	0,3	0,7	2500	4	68	47	21	81,3

Para as Tabelas 7.41 e 7.42, o N_{CI} da rede inicial é de 9 neurônios.

Para as Tabelas 7.43 e 7.44, o N_{CI} da rede inicial é de 11 neurônios.

Tabela 7.36: Simulação 20: Exemplo 1 com $FA = \frac{N_{CI}}{Erro\_Total}$

Dados													
S	NG	PI	TC	TM	T	α	β	NE	Nciv	CR	SR	$N°A$	PA (%)
2	5	8	0,6	0,1	0,01	0,3	0,7	250	19	168	24	15	93,8
6	5	8	0,6	0,01	0,01	0,4	0,2	250	19	167	24	15	93,8
9	5	8	0,6	0,01	0,01	0,1	0,05	1000	19	181	24	15	93,8
13	5	20	0,6	0,01	0,01	0,3	0,7	2500	20	159	24	15	93,8

Tabela 7.37: Simulação 21: Exemplo 2 com $FA = \frac{1}{Erro\_Total+N_{CI}}$

Dados													
S	NG	PI	TC	TM	T	α	β	NE	Nciv	CR	SR	$N°A$	PA (%)
11	5	20	0,6	0,01	0,01	0,3	0,7	2500	5	63	53	2	40

7.3.2 Resultados das Simulações para o RBES

Neste item serão descritos os resultados obtidos na implementação e nas simulações do sistema simbólico proposto, com os mesmos exemplos aplicados para o sistema conexionista desenvolvido neste trabalho.

- Visualizando as Tabelas 7.35 e 7.36, ambas nos mostram que o número de acertos para classificar os números decimais de 0 à 15 em par, ímpar e primo atingiram bons resultados. Isto quer dizer que o sistema simbólico foi capaz de traduzir o conhecimento embutido nas conexões do NNES em regras, independente de qual tipo de função de aptidão utilizada;

Tabela 7.38: Simulação 22: Exemplo 2 com $FA = \frac{N_{CI}}{Erro\_Total}$

Dados													
S	NG	PI	TC	TM	T	α	β	NE	Nciv	CR	SR	$N°A$	PA (%)
5	5	8	0,6	0,01	0,01	0,3	0,7	500	8	106	79	4	80
9	5	20	0,6	0,01	0,01	0,3	0,7	2500	10	108	81	4	80

Tabela 7.39: Simulação 23: Exemplo 3 com $FA = \frac{1}{Erro\_Total + N_{CI}}$

Dados													
S	NG	PI	TC	TM	T	α	β	NE	Nciv	CR	SR	$N°A$	PA (%)
4	5	8	0,6	0,01	0,01	0,1	0,05	250	4	8	7	5	83,3
6	5	8	0,6	0,01	0,01	0,4	0,2	250	5	11	6	5	83,3
9	5	10	0,6	0,01	0,01	0,1	0,05	1500	4	8	7	5	83,3
10	5	10	0,6	0,01	0,01	0,3	0,7	1500	4	10	7	4	66,7

Tabela 7.40: Simulação 24: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$

Dados													
S	NG	PI	TC	TM	T	α	β	NE	Nciv	CR	SR	$N°A$	PA (%)
2	5	8	0,6	0,1	0,01	0,3	0,7	250	10	23	12	5	83,3
4	5	8	0,6	0,01	0,01	0,1	0,05	250	10	23	7	3	50
7	5	8	0,8	0,01	0,01	0,3	0,7	250	10	30	13	5	83,3
9	5	10	0,6	0,01	0,01	0,1	0,05	1500	11	22	6	4	66,7
10	5	10	0,6	0,01	0,01	0,3	0,7	1500	10	22	11	5	83,3

- As Tabelas 7.37 e 7.38, referentes a classificação das vogais, permitem observar que ao se traduzir este conhecimento na saída do NNES em regras, dada a $FA = 1/Erro\_Total + N_{CI}$, não foi satisfatório. Isto se deve ao fato de que a rede, ao ser treinada, não aprendeu adequadamente. Todavia, com a $FA = N_{CI}/Erro\_Total$, os resultados melhoraram sensivelmente;

- Para o caso do exemplo de classificação de crises epilépticas idealizado pelo médico (Tabelas 7.39 e 7.40), independente do tipo de FA utilizada, foram obtidos resultados muitos bons de classificação e tradução do sistema conexionista em simbólico;

- Para os dois últimos exemplos, onde o médico se apoiou na extração dos dados clínicos via prontuários, aplicados para testar o RBES, o primeiro deles (Tabelas 7.41 e 7.42), demostra que para a $FA = 1/Erro\_Total + N_{CI}$ os resultados foram baixos comparados com o do NNES. Contudo, para uma das simulações, o resultado foi melhor, $66,7\%$. Entretanto, com a $FA = N_{CI}/Erro\_Total$, os

Tabela 7.41: Simulação 25: Exemplo 3 com $FA = \frac{1}{Erro\_Total + N_{CI}}$

Dados													
S	NG	PI	TC	TM	T	α	β	NE	Nciv	CR	SR	$N^{o}A$	PA (%)
3	10	8	0,6	0,01	0,01	0,1	0,05	500	8	38	23	6	66,7
6	5	10	0,6	0,01	0,01	0,1	0,05	1500	9	45	28	2	22,2
7	5	20	0,6	0,01	0,01	0,3	0,7	1500	8	49	26	2	22,2

Tabela 7.42: Simulação 26: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$

Dados													
S	NG	PI	TC	TM	T	α	β	NE	Nciv	CR	SR	$N^{o}A$	PA (%)
2	10	8	0,6	0,01	0,01	0,3	0,7	500	13	43	27	7	77,8
3	5	8	0,6	0,01	0,01	0,1	0,05	500	13	50	29	6	66,7
7	5	10	0,6	0,01	0,01	0,1	0,05	1500	13	51	32	5	55,6
8	5	10	0,6	0,01	0,01	0,3	0,7	1500	13	49	26	7	77,8

resultados obtidos foram próximos aos atingidos pelo NNES;

- Nas Tabelas 7.43 e 7.44, observa-se que a $FA = 1/Erro\_Total + N_{CI}$, não foi uma boa escolha, assim como no caso do NNES. Desta forma, os resultados foram prejudicados, ou seja, o número de acertos foram no máximo de $54,55\%$ para o NNES. Contudo, com a utilização da $FA = N_{CI}/Erro\_Total$, os resultados foram outros. Pode-se observar que estes resultados foram similares aos obtidos pelo NNES;

Tabela 7.43: Simulação 27: Exemplo 3 com $FA = \frac{1}{Erro\_Total + N_{CI}}$

Dados													
S	NG	PI	TC	TM	T	α	β	NE	Nciv	CR	SR	$N^{o}A$	PA (%)
4	5	8	0,6	0,1	0,01	0,3	0,7	1000	7	31	20	3	27,3
5	5	8	0,6	0,1	0,01	0,3	0,7	2000	7	27	16	2	18,2

Tabela 7.44: Simulação 28: Exemplo 3 com $FA = \frac{N_{CI}}{Erro\_Total}$

Dados													
S	NG	PI	TC	TM	T	α	β	NE	Nciv	CR	SR	$N°A$	PA (%)
1	5	8	0,6	0,1	0,01	0,1	0,05	250	15	70	40	7	63,6
3	5	8	0,6	0,1	0,01	0,1	0,05	1000	16	70	36	8	72,7
5	5	8	0,6	0,1	0,01	0,3	0,7	1250	16	78	41	5	45,5

- Todos os exemplos aplicados para testar o RBES demostraram que os procedimentos, para a redução do número de regras redundantes geradas após o aprendizado e refinamento do NNES, foram adequados.

Capítulo 8

Epílogo

Neste Capítulo serão apresentadas as contribuições e os resultados conclusivos alcançados no desenvolvimento da metodologia proposta para facilitar a tarefa de AC. Vários paradigmas da área de IA foram utilizados para se atingir os objetivos propostos. Além disso, serão ainda sugeridos alguns trabalhos futuros para dar continuidade à pesquisa em questão.

8.1 Contribuições deste Trabalho

Os sistemas complexos associados com as atividades humanas são freqüentemente *pobres*. A computação cognitiva[1] provê um modo efetivo e eficiente para executar uma análise de sistemas de processos no qual procedimentos tecnológicos e atividades humanas são interdependentes. Neste trabalho uma metodologia para o desenvolvimento de um HES é proposta sob os aspectos de AC, onde o tratamento de imprecisão é fundamental, de forma a se tentar *emular* o raciocínio humano no sentido de chegar a uma determinada conclusão. Conjuntos *fuzzy* e lógica *fuzzy* podem ser utilizados para expressar informações imprecisas. Empregando-se esta ferramenta, um computador pode compreender expressões vagas naturais para os humanos, por exemplo, *uma pressão muito alta ou pressão bastante alta.* Similarmente, regras abruptas podem ser traduzidas como regras *fuzzy.* Então, a fase da AC é baseada no aprendizado por paradigmas simbólico e conexionista, ou seja, esta fase é composta por um conjunto de regras *fuzzy* iniciais e por exemplos. Assim, é possível se montar o NNES inicial com as regras *fuzzy* iniciais. Após, com os exem-

[1]Neste trabalho a computação cognitiva é tratada como uma coleção de tecnologias de informações emergentes, a qual é encontrada no sistema nervoso, no raciocínio humano, na tomada de decisão e na seleção natural. Geralmente fazem parte da computação cognitiva as RNA, a lógica *fuzzy* e a computação evolucionária [102].

plos, o NNES é treinado e refinado de modo que as capacidades intrínsecas deste sistema sejam exploradas. Como uma das dificuldades do paradigma conexionista corresponde a explanação de como este chegou a uma dada solução, o conhecimento pode ser transferido para o RBES, o qual usa também a lógica *fuzzy* para lidar com a imprecisão.

Desta forma, com o auxílio desta pesquisa a metodologia desenvolvida aqui, no sentido de facilitar a tarefa de AC, tem várias contribuições para o aprendizado de máquinas. As seguintes contribuições podem ser citadas:

1. Ao invés de se utilizar uma das heurísticas encontradas na literatura com respeito ao número de neurônios que deve ser colocado na camada intermediária de uma RNA, como é o caso de R.C. Eberhart [54], optou-se inicialmente definir o número de neurônios na camada intermediária do NNES *fuzzy* em função do número de regras iniciais elicitadas durante o processo de EC. Porém, o número final de neurônios na camada intermediária do NNES *fuzzy* é encontrado pelo algoritmo de aprendizado proposto, ou seja, o GENBACK;

2. O algoritmo de aprendizado GENBACK para o treinamento do NNES *fuzzy* foi desenvolvido com inspiração no algoritmo de retropropagação clássico. Algumas alterações foram realizadas. Uma delas ocorreu na incorporação dos conectivos lógicos E/OU no local da operação somatório de pesos, isto é, foi substituída pelos operadores de agregação Min/Max e pelo produto algébrico da lógica *fuzzy*. Além disso, ele é capaz de otimizar o número de neurônios na camada intermediária do NNES *fuzzy*. Observou-se que apesar da operação matemática do ajuste dos pesos, no passo para trás, envolver as funções Min/Max, cuja problematica sobre a derivabilidade deste tipo de função já ter sido abordada no Capítulo 5, o NNES ainda alcançou resultados consideráveis no que diz respeito ao seu aprendizado.

3. O uso do AG na otimização da topologia do NNES *fuzzy*. Neste aspecto, a contribuição dada por este trabalho, diz respeito a escolha e a definição da função de aptidão aplicada ao problema em estudo. Para a definição do mesmo foram necessárias duas funções de aptidão. Ambas foram definidas em função de duas variáveis: o número de neurônios na camada intermediária do NNES vencedor de cada indivíduo da população gerada, e o erro total obtido na saída do NNES, após o treinamento deste pelo GENBACK. Deste modo, elas foram especificadas como: $FA = N_{CI}/Erro\_Total$ e $FA = 1/Erro\_Total + N_{CI}$. Apesar de ser uma contribuição a escolha e a definição das funções de aptidão

mencionadas anteriormente, ainda resta o problema de achar uma função de custo melhor;

4. A capacidade de generalização de uma RNA, principalmente a do tipo abrupta, já foi bastante difundida. Quanto as RNA *fuzzy* pouco se descreve sobre este assunto. No Capítulo 1, na seção referente a descrição dos HES desenvolvidos até o presente momento, pode ser observado em que nenhum deles se menciona sobre este tema. Outras pesquisas foram feitas nesta linha, e pouca coisa foi encontrada descrevendo claramente sobre este assunto, apenas o que se menciona é que a RNA aprendeu, e algumas vezes, é citada que ela alcançou valores ou resultados satisfatórios com respeito a uma dada aplicação de classificação de padrões. Neste sentido, no Capítulo 7 foram realizadas várias simulações para testar esta característica. Lá são mostrados especificamente se para um dado exemplo a RNA *fuzzy* conseguiu ou não generalizar. Várias tabelas foram utilizadas para demonstrar esta característica;

5. No desenvolvimento desta pesquisa uma etapa interessante foi a criação e a implementação do algoritmo FUZZYRULEXT. Algumas particularidades deste algoritmo podem ser citadas: a facilidade de implementação e aplicação; os pesos acumulados wet_i tratados não somente como os pesos máximos, mas também como os pesos mínimos, de modo que se possa lidar também com entradas negadas; a simplicidade do processo de eliminação das regras geradas como redundantes.

8.2 Conclusões

A metodologia proposta nesta pesquisa foi desenvolvida no sentido de facilitar a tarefa de AC. Neste aspecto, foram utilizados dois dos principais paradigmas da área de IA para modelar o problema em estudo: o conexionista e o simbólico. Assim chegou-se a várias conclusões:

1. O desempenho dos estudos preliminares tem mostrado que, com a aplicação da metodologia proposta, se tem alcançado uma fase de AC mais fácil do que se utilizasse somente técnicas simbólicas. O hibridismo, por outro lado, permite complementar o NNES com explicações facilitadas do raciocínio que, na maioria dos casos, são difícieis de se obter com um sistema conexionista;

2. Como já mencionado, o algoritmo GENBACK possui algumas particularidades em relação ao do retropropagação clássico. Uma delas ocorreu na incorporação

dos conectivos lógicos E/OU no local da operação somatório de pesos, isto é, foi substituída pelos operadores de agregação Min/Max e pelo produto algébrico da lógica *fuzzy*. Observou-se que a operação matemática do ajuste dos pesos, no passo para trás, por envolver as funções Min/Max, fez surgir a problemática sobre a derivabilidade da mesma. Uma maneira de se contornar este problema e que foi estudada durante o desenvolvimento deste trabalho esta relacionada com a função $Lor(x)$. Contudo, foi visto que esta função não se aplica a um caso geral de estudo, mas sim a casos particulares, conforme mencionado nos Capítulos 5 e 7. Especificamente no Capítulo 5 são também justificados, com maiores detalhes, os problemas da derivabilidade das funções E/OU utilizando a matemática clássica. Assim, mesmo se utilizando o processo de têmpera simulada para contornar o referido problema, não foi possível *saltar* para um outro ponto (nível) da trajetória do deslocamento do gradiente descendente que fosse estável, conseqüentemente, derivável. Pode-se observar em algumas das simulações realizadas no Capítulo 7, que vários dos resultados não alcançaram um bom desempenho devido a esta dificuldade. Claro que, também, a escolha da função de aptidão influenciou consideravelmente as respostas para o sistema aplicado.

3. Outra meta alcançada diz respeito a otimização da topologia da RNA *fuzzy* adotada. Para este processo optou-se pelo AG para otimizar o tamanho da camada intermediária. Entretanto, quando se deseja aplicar o AG nesta direção, é necessário que se tenha o cuidado de respeitar alguns princípios. Eles são:

   - A escolha e a definição da função de aptidão aplicada ao problema aqui abordado. Partiu-se do pressuposto que esta seria deduzida somente pelo erro total obtido na saída do NNES após o treinamento deste pelo GENBACK. A primeira tentativa de definição para a função custo foi com respeito ao inverso deste erro. Contudo, os resultados não foram adequáveis devido a rede vencedora sempre possuir um número reduzido de neurônios na camada intermediária. Então, foram feitos vários outros experimentos e chegou-se a conclusão de que para o problema em estudo, a melhor solução seria utilizar uma outra função custo, que fosse plausível ao objetivo desejado, de forma que esta fosse dependente de uma outra variável, além do erro total obtido na saída do NNES. Assim, observou-se que a função de aptidão teria que ser composta também do número de neurônios da camada intermediária da rede vencedora em cada geração. Desta forma, as chances das redes geradas com um número de neurônios na camada intermediária maior que a da rede inicial, aumen-

taria. Então, para a definição do problema em questão foram necessárias duas funções de aptidão. Ambas foram definidas em função de duas variáveis: o número de neurônios na camada intermediária do NNES vencedor de cada indivíduo da população gerada, e o erro total obtido na saída do NNES após o treinamento deste pelo GENBACK. Desta forma, obteve-se não somente uma função custo, mas sim, duas para definir o problema em questão. Uma delas corresponde a Equação 5.38 e, a outra, a Equação 5.39. Com a Equação 5.38 a rede vencedora teve mais neurônios que a da rede inicial, enquanto que com a Equação 5.39 a rede vencedora passou a ter menos neurônios que a rede inicial. A explicação do ocorrido pode ser revisto no Capítulo 5. Contudo, acredita-se que se fosse possível encontrar uma outra função custo que melhor descrevesse o problema em questão, dependendo o caso, aumentasse ou diminuisse o número de neurônios da camada intermediária do NNES, poderia se ter um ganho melhor;

- Um número mínimo e máximo de neurônios a serem colocados na camada intermediária do NNES:
  - No caso da maximização, a solução adotada levou em consideração os seguintes pontos: os valores para as taxas de cruzamento e mutação foram escolhidas empiricamente, isto é, para a maioria dos casos em estudo, o valor de 0,6 para o operador crossover e 0,1 para o de mutação atingiram um bom desempenho, mesmo para um número reduzido de indivíduos criados na população a cada geração. Contudo, neste caso, a taxa de mutação foi alterada para 0,01, devido a população gerada ser menor. Além disso, durante o processo de mutação ocorreu a criação de uma população de indivíduos com um valor muito acima daquela solução aceitável. Assim, para eliminar este efeito foram considerados as pecularidades do exemplo tratado: diagnóstico médico.
  - No caso da minimização da camada intermediária foram considerados os seguintes aspectos: com relação ao conjunto de exemplos dados para os casos reais de crises epilépticas foram possíveis determinar quais valores eram prevalentes com respeito aos demais. Isto se deve ao fato de que um médico é capaz de articular, com um certo grau de confiança, que para uma dada doença, um determinado sintoma (ou mais sintomas) é prevalente em relação aos outros. Para um melhor entendimento deste processo, usou-se a seguinte idéia, a qual não é

válida para os casos dos números cardinais e nem das vogais: o valor mínimo de neurônios na camada intermediária da rede vencedora deve ser, no máximo, igual ao número de dados (sintomas) prevalentes relacionados no conjunto de exemplos, ou seja, o número de neurônios na camada intermediária não deve ser inferior aos dados prevalentes que constam do conjunto de exemplos. Exemplificando: um determinado especialista médico, após uma série de exames (laboratoriais, físicos, etc.) e consultas de um certo paciente, concluiu que este tinha uma dada doença. Dos sintomas apresentados por este paciente, isto é, S_1, S_2, S_3, S_4 e S_5, três destes eram prevalentes, ou seja, S_2, S_4 e S_5. Desta forma, a rede vencedora não deve possuir um número de neurônios na camada intermediária menor do que três, mas sim igual ou maior que este valor. Esta ocorrência pode ser observada no Capítulo 7. Entretanto, houve alguns casos no qual o número de neurônios na camada intermediária alcançou valores abaixo do esperado. Isto é, esperava-se que a rede vencedora não tivesse valores inferiores ao número de sintomas prevalentes discriminados pelo especialista médico. Supõe-se que este fato ocorreu devido ao processo de mutação. O operador de mutação deve ter gerado uma pertubação irregular na cadeia de cromossomos. Tem-se conhecimento que uma RNA pode executar uma determinada tarefa de classificação de padrões para no mínimo 2 (dois) neurônios na camada intermediária. Para este caso, se há uma pertubação na cadeia de cromossomos o valor gerado deve ser escolhido aleatoriamente, de forma que não seja inferior a este, enquanto que o superior deve obedecer ao teorema de Komolgorov. Isto é, este valor é suposto ser igual, por exemplo, duas vezes o número de neurônios na camada de entrada e somado a 1 (um). Acredita-se que com esta idéia é possível se manter o conhecimento elicitado de um dado especialista sem se perdê-lo após o NNES ser treinado e refinado. Esta idéia foi válida para a maioria dos casos de aplicação de classificação de crises epilépticas apresentados neste trabalho;

4. Com relação, ainda, a etapa de otimização do NNES, durante a aplicação do GENBACK, foi implementada com algumas considerações sendo que algumas delas já foram mencionadas anteriormente. Mas uma delas corresponde ao uso do AG no algoritmo de treinamento para o NNES. A cada geração são criadas uma população com um número x de indivíduos (redes) em que, muitas ve-

zes, como foi observado durante as simulações, devido ao processo de seleção e a aplicação dos OG a cada geração, vários indivíduos mortos possuíam uma aptidão alta, isto é, nas gerações seguintes muitos indivíduos mais aptos deixaram de ser escolhidos. Então para superar esta dificuldade foi aplicada a seguinte condição em cada geração: além da determinação da rede vencedora com maior aptidão, a partir das funções de custo sugeridas neste trabalho na geração atual pelo processo corrente de determinação da nova geração de indivíduos, guardou-se também a rede que teve o maior valor de aptidão durante a geração anterior, independente desta ter sido eliminada ou não para a próxima geração. Assim, comparou-se o valor da aptidão da rede vencedora obtida na geração atual com a da rede ganhadora da geração anterior. Assim, observou-se que os resultados foram melhores do que os anteriores;

5. Após a determinação da rede otimizada foi realizado o refinamento da mesma. Quando a rede vencedora foi alcançada, obteve-se tanto o número de neurônios na camada intermediária otimizado, quanto a rede treinada. Desta forma, foi apresentada a mesma, outros conjuntos de teste com a finalidade de analisar o refinamento do NNES. Observou-se que com os conjuntos de padrões de testes apresentados ao NNES a maioria deles foi reconhecido;

6. O conhecimento que era antes localizado no NNES passou, após o aprendizado e o refinamento, a ser distribuído nas conexões da rede vencedora. Assim, após o refinamento do NNES, com a obtenção da rede vencedora, foi possível testar a seguinte condição: o uso da função de aptidão dada pela Equação 5.39, ao se reduzir mais uma vez o número de neurônios na camada intermediária da mesma, ou seja, a um valor igual a diferença entre o número de neurônios na camada intermediária da rede vencedora e a da rede inicial, é possível que o sistema conexionista continue a fazer a classificação dos padrões. Por exemplo, se a rede inicial tivesse 16 neurônios na camada intermediária e após a utilização do GENBACK (ao final de um número x de iterações e gerações), chegou-se a uma rede vencedora com 19 neurônios, era interessante saber se utilizando a rede vencedora apenas, como no exemplo ilustrado, com 3 neurônios na camada intermediária, o sistema conexionista continuaria a fazer a classificação dos padrões. Pode ser observado nas Tabelas 7.3 e 7.4 que o sistema foi capaz de realizar esta condição para vários casos com bons resultados, chegando a 87,5% de acertos. O mesmo pode ser visualizado nas Tabelas 7.17, 7.18, 7.25, 7.26, 7.31, 7.32, 7.33 e 7.34;

7. A capacidade de generalização do paradigma conexionista para a RNA E/OU foi também observada. Para testar esta característica marcante, já provada para as redes abruptas, foi aplicada à RNA em estudo dois exemplos: o da classificação das vogais e os dos casos de classificação de crises epilépticas. Então, após análise, concluir que:

   - Vogais: Usando-se a Equação 5.39, o NNES vencedor perdeu uma boa parte desta característica. Veja as Tabelas 7.11 e 7.12. Contudo, em relação a Equação 5.38, o paradigma conexionista conseguiu obter muito bons resultados. Veja as Tabelas 7.17 e 7.18;
   - Classificação de crises epilépticas: Visualizando as Tabelas 7.27 à 7.30, nota-se que muitos casos, utilizando-se a Equação 5.39, perderam um pouco desta característica. Entretanto, ao se utilizar a Equação 5.38, a capacidade de generalização do NNES aumentou. Veja as Tabelas 7.31 à 7.34;

8. Após o aprendizado do sistema conexionista, foi desenvolvido o sistema simbólico, ou seja, o RBES. Como já questionado no Capítulo 6, a necessidade de se dar uma explicação na saída do NNES, implementou-se o algoritmo de extração de regras *fuzzy*, ou seja, o FUZZYRULEXT para auxiliar nesta etapa. Este algoritmo é simples e eficaz, como pode ser observado nos resultados obtidos em função dos exemplos dos números cardinais, das vogais e dos casos reais para as crises epilépticas sugeridos e analisados neste trabalho (Veja as Tabelas de 7.35 à 7.44). Uma das particularidades do FUZZYRULEXT diz respeito a etapa de eliminação da geração de regras redundantes a partir dos dados obtidos na saída do NNES. Ele conseguiu reduzir o número total de regras geradas[2], consideravelmente, sem que se perdesse a maioria das informações contidadas nos dados fornecidos pelo NNES;

9. Um dos objetivos desta pesquisa era de implementar um sistema explicativo que conseguisse traduzir um conjunto de regras na forma *Se/Então* na saída do NNES. Para este sistema, a implementação foi realizada no *Shell* SINTA. Foi possível desenvolvê-lo para testar a eficiência dos resultados alcançados na saída do NNES, bem como para dar uma explicação das respostas obtidas nas saídas deste paradigma conexionista. Este sistema simbólico explicativo é o próprio RBES;

[2]Um dos grandes problemas que existe neste tipo de algoritmo é o grande número de regras geradas, bem como o número de regras redundantes. Ambas as razões diminuem a aplicabilidade deste algoritmo.

10. Validação do sistema conexionista e simbólico

    - Uma das condições de se validar um sistema em IA é, primeiramente, através dos *Toy Problems*, como foi o caso da classificação dos números cardinais de 0 à 15 em par, ímpar e primo e a da classificação das vogais. Depois aplica-se este tipo de sistema (conexionista, simbólico ou híbrido) para os casos reais, como foi o apresentado aqui para a classificação das crises epilépticas;
    - No caso dos exemplos de crises epilépticas apresentados neste trabalho, no mundo real é possível se ter três classes de percentuais de acertos clínicos. O primeiro deles corresponde ao atendimento de um médico que seja um clínico geral. Normalmente o número de acertos para uma gama de pacientes diagnosticados para a doença em questão, não passa de 1/3, pelo motivo deste não ser especialista da área e poder apenas diagnosticar que determinado paciente tem ou não epilepsia, não conseguindo classificá-la nos tipos que existem. A segunda classe trata com um neurologista geral. Neste caso, o número de acertos esperados seria em torno de 50%. Como pode-se observar, melhorou. Contudo, não seria o diagnóstico final adequado, devido este não ser, também, o especialista da área. O último e terceiro caso, que é o mais interessante para o engenheiro de conhecimento saber, corresponde ao especialista de domínio, cuja porcentagem de acerto está em torno de 80% à 90% dos casos atendidos. Desta forma, é possível se ter uma idéia sobre um dado SE apresenta ou não um bom desempenho, dentro de uma dada área de domínio médico[3];
    - Dentro da faixa estipulada de acertos de um diagnóstico de classificação de crises epilépticas, referenciadas no item anterior, o protótipo do NNES e do RBES desenvolvido para a aplicação médica, exemplificada neste trabalho, pode ser classificado da seguinte forma:
        - Na maioria dos casos, ou seja, os melhores resultados obtidos foram quando o sistema conseguiu diagnosticar e classificar os pacientes analisados entre 68% à 78%. Com esta porcentagem foi possível classificar a metodologia proposta para o problema de facilitar a tarefa de AC, como sendo entre um diagnóstico realizado por um neurologista geral e o especialista da área;

[3]As classes de percentuais de acertos clínicos aqui apresentados foram concedidos da experiência da área de domínio do Professor Paulo C.T. Bittencourt (M.D., M.Sc.), da Universidade Federal de Santa Catarina. Sua especialidade (domínio) é na área de neurologia, mais especificamente, classificação de crises e síndromes epilépticas.

  - O restante dos casos podem ser classificados dentro da faixa de acertos estipulados como sendo o do clínico geral;

- Acredita-se que o sistema não tenha tido um bom desempenho em vários casos devido:
  - A dificuldade de acesso a um banco de dados sobre classificação de crises epilépticas (quantidade e qualidade de dados de informações) mais próximo ao ideal;
  - Aos problemas apresentados pelas RNA baseadas em neurônios E/OU, cujas limitações (neste caso incluindo as suas derivabilidades) ainda são um campo aberto para muito trabalho.

8.3 Trabalhos Futuros

Algumas metas futuras serão citadas a seguir para dar continuidade na linha de pesquisa abordada neste trabalho.

1. Para o desenvolvimento do GENBACK foram estudadas várias metodologias empregadas para superar a problemática das funções E/OU não serem deriváveis (Ver Capítulo 1). Porém, somente duas delas foram utilizadas aqui, por serem, na época, consideradas adequadas ao objetivo proposto. Uma delas foi o trabalho desenvolvido por X. Zhang et al. [194]. A outra foi de S. Mitra et al. [134][135][133]. Em ambos os casos, as ferramentas matemáticas sugeridas por estes, pelo que pode-se observar, pareciam ser um bom caminho de partida para o problema abordado nesta pesquisa científica. Contudo, ao longo do tempo, descobriu-se que ambas as metodologias usadas para contornar o problema da descontinuidade das funções Min/Max não se mostraram geral o suficiente para todos os problemas. Elas podem apenas encontrar uma solução. Para contornar este problema, inseriu-se alguns mecanismos, como foi o caso da têmpera simulada e os operadores alternativos para substituir a função $Lor(x)$, já que as operações T-Norma e S-Conorma são determinadas pelos operadores *fuzzy* do tipo Min/Max, que são considerados não-iterativos. Desta forma, usou-se vários operadores de implicação, como foi o caso do de Lukasiewicz. Isso, obviamente, trouxe um pequeno prejuízo ao trabalho, mas as simulações têm mostrado que, muitas vezes, o sistema consegue convergir para uma solução aceitável. Na realidade, baseou-se em teorias que já foram desenvolvidas por outros, mas que, lamentavelmente, não são gerais o suficiente para determinados problemas. Isto significa que tem ainda muito campo

de pesquisa aberta e que, talvés, o caminho não seja este mas poderia, por exemplo, ser utilizado um outro método para o treinamento desta RNA *fuzzy* direta, como os exemplificados e sugeridos no Capítulo 5 deste trabalho;

2. Criar um outro sistema, o Sistema Especialista Explicativo (*Explanation Expert System* - EES), que traduza, a partir dos dados obtidos na saída do RBES, em comparação com o NNES, na forma de regras *fuzzy Se/Então*. Por enquanto, o módulo RBES está fazendo o papel do EES, mas não está comparando os resultados obtidos para o paradigma conexionista com o do simbólico;

3. Integrar os três módulos (NNES, RBES e EES), em um único sistema;

4. Testar o sistema integrado para um caso geral, independente do domínio;

5. Comparar a saída do RBES, após o refinamento e extração de regras do NNES, com outro sistema simbólico usando também regras. Contudo, neste segundo sistema, os dados de entrada serão baseados em uma base de conhecimento já existente, de modo que estes dados não passarão pelo processo de tradução de regras, como é o caso das entradas do NNES, no sentido de extrair algumas regras iniciais de um conjunto de exemplos cedidos por um especialista de um domínio para o desenvolvimento de um sistema conexionista inicial. Conjectura-se também, como na montagem de um NNES e após o seu refinamento, que é mais provável que a sua saída apresente melhores resultados do que no sistema simbólico que é elicitado de uma base de conhecimento já existente. Neste último caso, a base de conhecimento tem que ser suficientemente extensa para representar o conhecimento de um determinado domínio. Assim, existe o problema de inconsistência de regras, bem como o tempo de aprendizado e o método de busca para a obtenção de um resultado plausível;

6. Criar uma base de conhecimento automatizado para a área médica de diagnóstico de crises e síndromes epilépticas. Motivos estes já abordados na subseção anterior;

7. Usar uma das mais recentes técnicas de extração de conhecimento de grande massa de dados de forma inteligente e automatizada, ou seja, *Data Mining* (mineração de dados) em lugar da metodologia proposta neste trabalho. Claro que os exemplos aplicados para testar a metodologia desenvolvida nesta pesquisa foi para aplicações pequenas, mas que poderiam ser expandidas para maiores. Além do que, poderia-se fazer uma análise comparativa entre estas duas técnicas;

8. Escolher um outro algoritmo da computação evolucionária, por exemplo, a Programação Evolucionária, que não se utiliza de um dos OG, ou seja, o *crossover*, e aplicá-la a metodologia aqui sugerida. Analisar a aplicação deste algoritmo de forma a se saber se os novos resultados serão melhores do que com a utilização dos AG.

Referências Bibliográficas

[1] K.-P. Adlassnig. Fuzzy set theory in medical diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 16, n. 2, p. 260–265, 1986.

[2] A.S. Algarve. *Simulação do Sistema Circulatório com Controle Neuronal Central.* Exame de Qualificação para Doutorado, UFSC, Florianópolis, Brasil, 1997.

[3] J.A. Anderson. A simple neural network generating an interactive memory. *Mathematical Biosciences*, v. 14, n. 1, p. 197–220, 1972.

[4] R. Andrews and J. Diederich. *Rules and Networks.* QUT - Queensland University of Technology, Brisbane, Australia, 1996.

[5] R. Andrews, J. Diederich, and A.B Tickle. A survey and critique of techniques for extracting rules from trained artificial neural newtorks. Site: *http//157.228.15.98/Ken_pubs.html*, Brisbane, Australia, 1998.

[6] R. Andrews and S. Geva. Rule extraction from a constrained error back propagation MLP. In: *Proceedings of the 6th Australian conference on Neural Networks*, p. 9–12, Brisbane Queensland, 1994.

[7] G. Araribóia. *Inteligência Artificial: Um Curso Prático.* Livros Técnicos e Científicos Editora Ltda, Rio de Janeiro, R.J., 1989.

[8] S. Austin. An introduction to genetic algorithms. *AI Expert*, v. 1, n. 1, p. 48–53, 1990.

[9] J.M. Barreto. *Contribution a L'etude du Reglage Optimal Avec Fonction de Cout Quadratique.* Ph.D. Thesis, Laboratoire d'Automatique, Faculté des Sciences Appliquées, Bruxelles, Belgium, 1970.

[10] J.M. Barreto. *Conexionismo e a resolução de problemas.* Tese de Concurso para Professor Titular, UFSC, Florianópolis, Brasil, 1996.

[11] J.M. Barreto. *Inteligência Artificial No Limiar do Século XXI.* DUPLIC - Prestações de Serviços, Florianópolis, S.C., 1997.

[12] J.M. Barreto and F.M. de Azevedo. Connectionist expert systems as medical decision aid. *Artificial Intelligence in Medicine*, v. 5, n. 1, p. 515–523, 1993.

[13] J.M. Barreto, F.M. de Azevedo, W.C. Lima, and L.R. Epprecht. A neural network with fuzzy elements for a medical diagnosis. In: *IEEE Latinoamerican Conference, LATINCON'92*, p. 335–339, Santiago, Chile, 1992.

[14] J.M. Barreto, F.M. de Azevedo, C.I. Zanchin, and L.R. Epprecht. Associative memories in medical diagnosis. In: *10th International Congress on Medical Informatics*, p. 348–352. Springer-Verlag, Vienna, Austria, 1991. K.P. Adlassing and G. Grabner and S. Bengtsson and R.Hansen (eds.).

[15] H.R. Berenji. Refinement of approximate reasoning-based controllers by reinforcement learning. In: *Proceedings of the Eighth International Machine Learning Workshop*, p. 475–479, Evanston, IL, 1991.

[16] B.I. Blum. Clinical information systems - a review. *West Journal of Medicine*, v. 145, n. 1, p. 791–797, 1986.

[17] M.J.F. Braga, J.M. Barreto, and M.A. S. Machado. *Conceitos da Matemática Nebulosa na Análise de Risco.* Artes & Rabisjus Comunicação Empresarial Ltda., Rio de Janeiro, R.J., 1995.

[18] L.M. Brasil. Aquisição de conhecimento aplicada ao diagnóstico de epilepsia. Tese de Mestrado, Florianópolis, Brasil, UFSC, Engenharia Biomédica, 1994.

[19] L.M. Brasil. *Uma Proposta de Arquitetura para Sistemas Especialista Híbrido e a Correspondente Metodologia de Elicitação/Representação do Conhecimento.* Exame de Qualificação para Doutorado, UFSC, Florianópolis, Brasil, 1996.

[20] L.M. Brasil, F.M. de Azevedo, and J.M. Barreto. Algoritmo de aprendizado para redes neurais fuzzy and/or. *Pesquisa Naval - Suplemento Especial da Revista Marítima Brasileira, ISSN 1414-8595*, v. 1, n. 10, p. 111–132, 1997.

[21] L.M. Brasil, F.M. de Azevedo, and J.M. Barreto. An hybrid expert architecture for medical diagnosis. In: *Proceedings of The 3th International Conference on Artificial Neural Networks and Genetic Algorithms - ICANNGA'97, ISBN 3-211-83087-1*, p. 176–180, Norwich, England, 1997.

[22] L.M. Brasil, F.M. de Azevedo, and J.M. Barreto. Learning algorithm for connectionist systems. In: *Anales del XII Congreso Chileno de Ingeniería Eléctrica*, v. II, p. 697–702, Temuco, Chile, 1997.

[23] L.M. Brasil, F.M. de Azevedo, and J.M. Barreto. Uma arquitetura híbrida para sistemas especialistas. In: *III Congresso Brasileiro de Redes Neurais (CBRN'97), ISBN 85-900382-1-1*, p. 167–172, Florianópolis, Brasil, 1997.

[24] L.M. Brasil, F.M. de Azevedo, and J.M. Barreto. Uma arquitetura para sistema neuro-fuzzy-ga. In: *Anales del XII Congreso Chileno de Ingeniería Eléctrica*, v. II, p. 712–717, Temuco, Chile, 1997.

[25] L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and M. Noirhomme-Fraiture. Complexity and cognitive computing. In: *Proceedings of The 11th International Conference on Industial & Engineering Applications of Artificial Intelligence & Expert Systems - IEA-98-AIE (In Press)*, Castellon, Spain, 1998.

[26] L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and M. Noirhomme-Fraiture. Knowledge acquisition using a hybrid expert systems. In: *Proceedings of The World Multiconference on Systemics, Cybernetics and Informatics and, The 4th International Conference on Information Systems, Analysis and Synthesis - SCI'98-ISAS'98, ISBN 980-07-5079-7*, v. 2, p. 84–91, Florida, USA, 1998.

[27] L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and M. Noirhomme-Fraiture. A neuro-fuzzy-ga system architecture for helping the knowledge acquisition processo. In: *Proceedings of The International IEEE Joint Symposia on Intelligence and Systems (In Press)*, Washington-DC, USA, 1998.

[28] L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and M. Noirhomme-Fraiture. A new approach to classical backpropagation algorithm for neuro-fuzzy-ga systems learning. In: *Proceedings of The IASTED International Conference on Artificial Intelligence and Soft Computing - ASC'98, ISSN 1482-7913, ISBN 0-88986-256-7*, p. 489–492, Cancún, Mexico, 1998.

[29] L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and M. Noirhomme-Fraiture. A new hybrid architecture for helping the knowledge acquisition processo. In: *Proceedings of The 15th International Congress on Cybernetics - Interdisciplinary Symposium in Artificial Intelligence, Cognitive Science, and Philosophy for Social Progress (In Press)*, Namur, Belgium, 1998.

[30] L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and M. Noirhomme-Fraiture. Training algorithm for neuro-fuzzy-ga systems. In: *Proceedings of The 16th IASTED International Conference on Applied Informatics - AI'98, ISSN 1027-2666, ISBN 0-88986-250-8*, p. 45–47, Garmisch-Partenkirchen, Germany, 1998.

[31] L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and R. Garcia Ojeda. Extração de regras básicas para sistemas especialistas conexionistas. In: *Anales del IV Coloquio de Bioingenieria, COLOQUIO'97, ISBN 980-00-1112-9*, p. TS–40–TS–46, Valência, Venezuela, 1997.

[32] L.M. Brasil, F.M. de Azevedo, J.M. Barreto, and R.Garcia Ojeda. A hybrid architecture for expert systems. In: *Proceedings of The World Congress On Medical Physics and Biomedical Engineering, ISSN 0140-018/97*, v. 55, p. 517, Nice, France, 1997.

[33] L.M. Brasil, F.M. de Azevedo, R. Garcia Ojeda, and J.M. Barreto. Cooperation of symbolic and connectionist expert system techniques to overcome difficulties. In: *Proceedings do II Congresso Brasileiro de Redes Neurais - CBRN'95*, p. 177–182, Curitiba, Brazil, 1995.

[34] L.M. Brasil, F.M. de Azevedo, R. Garcia Ojeda, and J.M. Barreto. A methodology for implementing hybrid expert systems. In: *Proceedings of The IEEE Mediteranean Electrotechnical Conference, MELECON'96, ISBN 0-7803-3109-5*, p. 661–664, Bari, Italy, 1996.

[35] T.A. Byrd, K.L. Cossik, and R.W. Zmud. A synthesis of research on requirements analysis and knowledge acquisition techniques. *Mis Quarterly*, v. 16, n. 1, p. 117–138, 1992.

[36] W.M. Caminhas, H.M.F. Tavares, and F. Gomides. Sistema de classificação de padrões usando redes neuro-fuzzy e redes neurais. In: *Anais do II Congresso Brasileiro de Redes Neurais*, Curitiba, Brasil, 1995.

[37] E. Charniac and D. McDermott. *Introduction to Artificial Intelligence.* Addison-Wesley Publishing Company, Inc., Reading, Massachussets, 1985.

[38] J.M. Chetupuzha and A.B. Badiru. Design considerations for knowledge acquisition. *Computers Industrial Engineering*, v. 21, n. 1-4, p. 257–261, 1991.

[39] M.W. Craven and J.W. Shavlik. Extracting tree-structured representations of trained networks. *Advances in Neural Information Processing Systems*, v. 8, n. 1, 1996.

[40] Y. Davidor. Genetic algorithms: a survey. In: *Dynamic, Genetic, and Chaotic Programming*, p. 323–338. John Wiley & Sons, Inc., New York, USA, 1992. B. Soucek and The IRIS Group (eds.).

[41] R. Davis, B.G. Buchanan, and E. Shortliffe. Production rules as a representation for a knowledge-based consultation program. *Artificial Intelligence*, v. 8, n. 1, p. 15–45, 1977.

[42] F.M. de Azevedo. *Contribution to the Study of Neural Networks in Dynamical Expert Systems.* Ph.D. Thesis, Institut d'Informatique, FUNDP, Namur, Belgium, 1993.

[43] F.M. de Azevedo. Uma proposta de modelos formais de neurônios e redes neurais artificiais. In: *III Congresso Brasileiro de Redes Neurais (CBRN'97), ISBN 85-900382-1-1*, p. 503–514, Florianópolis, Brasil, 1997.

[44] F.M. de Azevedo, J.M. Barreto, E.K. Epprecht, L.R. Epprecht, and W.C. Lima. Two approaches in case-based connectionist expert systems. In: *Artificial Intelligence and Neural Networks (AINN'91)*, p. 78–81. Acta Press, Anahaim, Calgary, 1991. M.H. Hanza (ed.).

[45] F.M. de Azevedo, J.M. Barreto, L.R. Epprecht, W.C. Lima, and C.I. Zanchin. A neural network approach for medical diagnosis. In: *Mini and Microcomputers in Medicine and Health Care 91*, p. 78–81. Acta Press, Anahaim, Calgary, 1991. D. Hudson (ed.).

[46] F.M. de Azevedo, J.M. Barreto, and W.C. Lima. A neural network approach for fuzzy knowledge bases. In: *Anales del IX Congreso de Ingeniería Electrica*, p. 9.1.1–9.1.6, Universidad de Tarapaca, Chile, 1991.

[47] F.M. de Azevedo, J.M. Barreto, W.C. Lima, and C.I. Zanchin. Teaching medical diagnosis with associative memories. In: *Abstracts of XI Systems Science International Conference*, p. 28–29, Wroclaw, Poland, 1992.

[48] F.M. de Azevedo, J.M. Barreto, W.C. Lima, C.I. Zanchin, and L.R. Epprecht. A neural network implementation of case base reasoning. In: *Proceeding of the 43th Meeting of the SBPC*, p. 132–133, Rio de Janeiro, Brasil, 1991.

[49] A. de Baenst-Vandenbrouck, F.M. de Azevedo, and J.M. Barreto. Is a neural network object-oriented? a look at an application in medical decision support. In: *13th International Congress on Cybernetics*, p. 290–294, Namur, Belgium, 1992.

[50] J. P. de Oliveira. Redes neurais artificiais para representação de conhecimento médico em anestesiologia. Tese de Mestrado, Florianópolis, Brasil, UFSC, 1997.

[51] D. Diaper. *Knowledge Elicitatio: Principles, Techniques and Applications.* Hohb Wiley & Sons, New York, USA, 1989.

[52] J. Dias. *Treinamento Híbrido de Redes Neurais para Processamento de Informações Biomédicas.* Exame de Qualificação para Doutorado, UFSC, Florianópolis, Brasil, 1996.

[53] J. Diederich and A.B. Tickle. Explanation and collective computation. *Complexity International*, v. 2, n. 1, 1995.

[54] R.C. Eberhart and R.W. Dobbins. *Neural Network PC Tools - A Pratical Guide.* Academic Press, Inc., Laurel, Mayland, 1990.

[55] M. Figueiredo, F. Gomide, and W. Pedrycz. Arquitetura e aprendizagem de uma rede nebulosa. In: *Proceedings do X Congresso Brasileiro de Automática, CBA*, v. 2, Rio de Janeiro, R.J., 1994.

[56] D.B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press, New York, USA, 1995.

[57] H.C. Fu and J.J. Shann. A fuzzy neural network for knowledge learning. *International Journal of Neural Systems*, v. 5, n. 1, p. 13–22, 1994.

[58] L.M. Fu. Integration of neural heuristics into knowledge-based inference. *Connection Science*, v. 1, n. 3, p. 325–340, 1989.

[59] L.M. Fu. Rule learning by searching on adapted nets. In: *Proceedings of The International Conference on Artificial Intelligence, AAAI-91*, p. 325–340, Anaheim, CA, 1991.

[60] L.M. Fu. Connectionism for fuzzy learning in rule-based expert systems. In: *Proceedings of The 5th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE'92*, p. 337–340, Paderborn, Germany, 1992.

[61] L.M. Fu. A connectionist approach to rule refinement. *Applied Intelligence, International Journal Artificial Intelligence Neural Networks Complex Problem-Solving Technologic*, v. 2, n. 1, p. 93–103, 1992.

[62] L.M. Fu. Detection of semantically incorrect rules knowledge-based systems. *Knowledge-Based Systems*, v. 5, n. 2, p. 117–124, 1992.

[63] L.M. Fu. Knowledge-based refinement by backpropagation. *Data & Knowledge Engineering*, v. 7, n. 1, p. 35–46, 1992.

[64] L.M. Fu. Knowledge-based connectionism for revising domain theories. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 23, n. 1, p. 173–182, 1993.

[65] L.M. Fu. Rule generation from neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 28, n. 8, p. 1114–1124, 1994.

[66] L.M. Fu and L.C. Fu. Mapping rule-based systems into neural architecture. *Knowledge-Based Systems*, v. 3, n. 1, p. 48–56, 1990.

[67] S.I. Gallant. Connectionist expert systems. *Communications of the ACM*, v. 31, n. 2, p. 152–169, 1988.

[68] R. Garcia. *Técnicas de Inteligência Artificial Aplicadas ao Apoio à Decisão Médica na Especialidade de Anestesiologia.* Tese de Doutorado, UFSC, Engenharia Biomédica, Florianópolis, Brasil, 1992.

[69] R. Garcia, F.M. Azevedo, and J.M. Barreto. Genetic algorithms in the optimal choice of neural networks for signal processing. In: *Proceedings of The 38th. Midwest Symposium on Circuits and Systems*, v. 2, p. 1361–1364, Rio de Janeiro, Brazil, 1995.

[70] H. Garis. Artificial embryology: the genetic programming of an artificial embryo. In: *Dynamic, Genetic, and Chaotic Programming*, p. 373–393. John Wiley & Sons, Inc., New York, USA, 1992. B. Soucek and The IRIS Group (eds.).

[71] S. Genaro. *Sistemas Especialistas - O Conhecimento Artificial.* Livros Técnicos e Científicos Editora S. A., Rio de Janeiro, Brasil, 1986.

[72] N. Gilbert. Explantion an dialogue. *The Knowledge Engineering Review*, v. 4, n. 3, p. 235–247, 1989.

[73] D.E. Goldberg. *Genetic Algorithms in Search, Optmization, and Machine Learning.* Addison-Wesley Publishing Company, Inc., Reading, Massachussets, 1989.

[74] J.J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, v. SMC-16, n. 1, p. 122–128, 1986.

[75] M.M. Gupta and J. Qi. Connectives (and, or, not) and t-operators in fuzzy reasoning. In: *Conditional Logic in Expert Systems*, p. 211–233. Elsevier Science Publishers B.V., North-Holland, Amsterdam, 1991. I.R. Goodman, M.M. Gupta, H.T. Nguyen, and G.S. Rogers (eds.).

[76] M.M. Gupta and D.H. Rao. *Neuro-Control Systems: A Tutorial.* IEEE Press, Saskatoon, Canada, 1994.

[77] M.M. Gupta and D.H. Rao. On the principles of fuzzy neural networks. *Fuzzy Set and Systems*, v. 61, n. 1, p. 1–18, 1994.

[78] A.C. Guyton and J.E. Hall. *Textbook of medical physiology.* W.E. Saunders Company, Philadelphia, London, tenth edition, 1996.

[79] S.K. Halgamuge and M. Glesner. A fuzzy-neural approach of pattern classification with the generation of rules based on supervised learning. In: *Proceedings of The Neuro-Nimes 92*, p. 167–173, Nanterre, 1992.

[80] S.K. Halgamuge and M. Glesner. Neural networks in designing fuzzy systems for real world applicatins. *Fuzzy Sets and Systems*, v. 65, n. 1, p. 1–12, 1994.

[81] A. Hart and J. Whatt. Evaluating black-boxes as medical decision aids: issues arising from a study of neural networks. *Medical Information*, v. 15, n. 3, p. 229–236, 1990.

[82] Y. Hayashi. A neural expert systems using fuzzy teaching input. In: *Proceedings of The IEEE International Conference on Fuzzy Systems*, p. 485–491, San Diego, CA, 1989.

[83] Y. Hayashi. A neural expert system with automated extraction fuzzy. *Advances in Neural Information Processing Systems*, v. 3, n. 1, p. 578–584, 1991.

[84] Y. Hayashi. Fuzzy neural expert system with automated extraction of fuzzy if-then rules from a trained neural network. In: *Analysis and Management of Uncertainty: Theory and Applications*, p. 171–181. North-Holland Publishing Co., North-Holland, Amsterdam, 1992. B.M. Ayyub and M.M. Gupta and L.N. Kanal (eds.).

[85] Y. Hayashi and J.J. Buckley. Approximations between fuzzy expert systems and neural networks. *International Journal of Approximate Reasoning*, v. 10, n. 1, p. 63–73, 1994.

[86] S. Haykin. *Neural Networks: A comprehensive Foundation.* Macmillian College Publishing Company, Inc., New York, USA, 1994.

[87] D. Hebb. *Organization of Behavior.* John Wiley & Sons, New York, USA, 1949.

[88] D. Hebb. *Psicologia.* W.B.Saunders Company, Philadelphia, USA, 2 edition, 1979.

[89] D.M. Himmelblau. *Applied Nonlinear Programming.* McGraw-Hill Book Company, Texas, USA, 1987.

[90] K. Hirota and W. Pedrycz. Fuzzy logic neural networks: Design and computations. In: *Proceedings of The IEEE International Joint Conference on Neural Networks*, v. 1, p. 152–157, New York, USA, 1991.

[91] K. Hirota and W. Pedrycz. Fuzzy logic neural networks: Design and computations. In: *Proceedings of The IEEE International Joint Conference on Neural Networks*, v. 1, p. 152–157, New York, USA, 1991.

[92] J.H. Holland. *Adaption in Natural and Artificial Systems.* MIT Press, Cambridge, Massachusetts, 1975.

[93] S. Horikawa, T. Foruhashi, and Y. Uchikawa. On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. *IEEE Transactions on Neural Networks*, v. 3, n. 5, p. 801–806, 1992.

[94] H. Ishibuchi, R. Fujioka, and H. Tanaka. Neural networks that learn from fuzzy if-then rules. *IEEE Transactions on Fuzzy Systems*, v. 1, n. 2, p. 85–97, 1993.

[95] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, v. 3, n. 1, p. 79–87, 1991.

[96] J.-S.R. Jang. Fuzzy modeling using generalized neural networks and kalman filter algorithm. In: *Proceedings of The Ninth National Conference on Artificial Intelligence (AAAI-91)*, v. 1, p. 762–767, 1991.

[97] J.-S.R. Jang. *Neuro-Fuzzy Modeling: Architecture, Analyses and Applications.* Ph.D. Thesis, University of California, Berkeley, CA, 1992.

[98] J.-S.R. Jang. Anfis: Adaptive-network-based fuzzy inference systems. *IEEE Transactions Systems, Man & Cybernetics*, v. 23, n. 1, p. 665–685, 1993.

[99] J.-S.R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing: A computational Approach to Learning and Machine Intelligence.* Prentice-Hall, Inc., NJ, USA, 1997.

[100] D.J. Janson and J.F. Frenzel. Training product unit neural networks with genetic algorithms. *IEEE Expert*, v. 1, n. 1, p. 26–33, 1993.

[101] J.Hopfield. Neural networks and physical systems with emergent collectives computational abilities. In: *Proceedings of The National Academy of Sciences*, v. 79, p. 2554–2558, 1982.

[102] R.C. Johnson. What is cognitive computing? *Dr. Dobb's Journal*, v. 1, n. 1, 1993.

[103] A. Kandel. *Fuzzy Mathematical Techniques with Applications.* Addison-Wesley Publishing Company, Inc., Florida, USA, 1986.

[104] A. Kandel and G. Langholz. *Hybrid Architectures for Intelligent Systems.* CRC Press, Inc., Boca Raton, Florida, 1992.

[105] S.V. Kartalopoulos. *Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications.* IEEE Press, Inc., New York, USA, 1996.

[106] J.P. Kassirer. Diagnostic reasoning. *Annals of Internal Medicine*, v. 110, n. 11, p. 893–900, 1989.

[107] J.P. Kassirer and G.A. Gorry. Clinical problem solving: a behavioral analysis. *Annals of Internal Medicine*, v. 89, n. 1, p. 245–255, 1978.

[108] T. Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, v. 21, n. 1, p. 353–359, 1972.

[109] B. Kosko. Bidirectional associative memories. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 18, n. 1, p. 49–60, 1988.

[110] R. Kowalski. *Logic for Problem Solving.* Computer Science Library, New York, USA, 1979.

[111] T. Kozek, T. Roska, and L.O. Chua. Genetic algorithm for cnn template learning. *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, v. 40, n. 6, p. 392–402, 1993.

[112] L.I. Kuncheva. Evaluation of computerized medical diagnosis decisions via fuzzy sets. *International Joint Biomedical Computation*, v. 28, n. 1, p. 91–100, 1991.

[113] J. Lawrence. *Introduction to Neural Networks and Expert Systems.* California Scientific Software, Nevada City, CA, 1992.

[114] L. Leitthold. *O Cálculo com Geometria Analítica*, v. 1. Editora Harper & Row do Brasil Ltda., São Paulo, SP, 2ª edition, 1982.

[115] C.F. Liaw, B.S. Stewart, and C.C. White. Multiobjective heuristic search in and/or graphs. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 25, n. 11, p. 1513–1521, 1995.

[116] K.S.C. Linares. Sistema especialista nebuloso para diagnóstico médico. Tese de Mestrado, Florianópolis, Brasil, UFSC, 1997.

[117] Y.I. Liou. Knowledge acquisition: Issues, techniques and methodology. *Data Base*, v. 1, n. 1, p. 59–64, 1992.

[118] R.P. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, v. 1, n. 1, p. 4–42, 1987.

[119] A. Machado. *Neuroanatomia Funcional.* Livraria Atheneu Editora, Rio de Janeiro, R.J., 1991.

[120] R.J. Machado, C. Ferlin, and A.F. Rocha. Combining semantic and neural networks in expert systems. Relatório técnico CCR-140, IBM Rio Scientific Center, p. 21, Rio de Janeiro, Brazil, 1992.

[121] R.J. Machado and A.F. Rocha. The combinatorial network: A connectionist model for knowledge based systems. In: *Proceedings of The 3rd International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, p. 578–587, Paris, France, 1990.

[122] R.J. Machado and A.F. Rocha. A hybrid architecture for fuzzy connectionist expert systems, 1992. A. Kandel and G. Langholz (eds.).

[123] R.J. Machado, A.F. Rocha, and B.F. Leao. Calculating the mean knowledge representation from multiple experts. In: *Multiperson Decision Making Using Fuzzy Sets and Possibility Theory*, p. 113–127. Kluwer Academic Publishers, Netherlands, 1990. B. Soucek and The IRIS Group (eds.).

[124] P. Mahey. *Programação Não-Linear: Introdução à Teoria e aos Métodos.* Editora Campus Ltda., Rio de Janeiro, R.J., 1987.

[125] R. Masuoka, N. Watanabe, A. Kawamura, Y. Owada, and K. Asakawa. Neurofuzzy systems - fuzzy inference using a structured neural network. In: *Proceedings of the International Conference on Fuzzy Logic and Neural Networks*, p. 173–177, Lizuka, Japan, 1991.

[126] G.R. Mateus and H.P.L. Luna. *Programação Não-Linear.* Editora Gráfica Formato Ltda., Belo Horizonte, M.G., 1986.

[127] W.A. Maurer. *Curso de Cálculo Diferencial e Integral*, v. 4. Editora Edgard Blucher Ltda., São Paulo, SP, 1977.

[128] J.L. McClelland, D.E. Rumelhart, and PDP Group. *Parallel Distributed Processing*, v. 1, 2. MIT Press, Cambridge, Massachusetts, 1986.

[129] W.S. McCulloch and W.H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, n. 1, p. 115–133, 1943.

[130] P. McGeorge and G. Rugg. The uses of contrived knowledge elicitation techniques. *Expert Systems*, v. 9, n. 3, p. 149–154, 1992.

[131] C. McMillan, M.C. Mozer, and P. Smolensky. The connectionist scientist game: rule extraction and refinament in a neural network. In: *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, 1991.

[132] L.S. Min. Sistemas baseados em conhecimentos para detecção e classificação de crises epilépticas. Tese de Mestrado, Florianópolis, Brasil, UFSC, Medicina Interna, 1994.

[133] S. Mitra, R.K. De, and S.K. Pal. Fuzzy multi-layer perceptron, inferencing, and rule generation. *IEEE Transactions on Neural Networks*, v. 8, n. 6, p. 1338–1350, 1997.

[134] S. Mitra and S.K. Pal. Logical operation based fuzzy MLP for classification and rule generation. *Neural Networks*, v. 7, n. 2, p. 353–373, 1994.

[135] S. Mitra and S.K. Pal. Fuzzy multi-layer perceptron, inferencing, and rule generation. *IEEE Transactions on Neural Networks*, v. 6, n. 1, p. 51–63, 1995.

[136] J.D. Moore and W.R. Swartout. A reactive approach to explanation. In: *Proceedings of The International Joint Conference on Artificial Intelligence, IJCAI'89*, v. 2, p. 1504–1510, Los Altos, CA, 1989.

[137] S.M. Nassar, H.S. Lopes, and A.R. Pozo. Elicitação de conhecimento. Relatório Interno - GPEB, 1992.

[138] D. Nauck, F. Klawonn, and R. Kruse. *Foundations of Neuro-Fuzzy Systems.* John Wiley & Sons, New York, USA, 1997.

[139] T. Nedjari. Miter: mutual information and template for extracting rules. In: *Proceeding of the NIPS'96 Rule Extraction from Trained Artificial Neural Network Workshop*, p. 54–61, Queensland University of Technology, 1996.

[140] E. Nick and S.R.O. Kellner. *Fundamentos de Estátistica para as Ciências do Comportamento.* Editora Renes Ltda., Rio de Janeiro, R.J., 1971.

[141] N.J. Nilsson. *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill Book Company, Menlo Park, CA, 1971.

[142] H. Okada, R. Masuoka, and A. Kawamura. Knowledge based neural network - using fuzzy logic to initialise a multilayered neural network and interpret postlearning results. *IEEE Transactions on Knowledge and Data Engineering*, v. 29, n. 3, p. 217–226, 1993.

[143] D.W. Opitz and J.W. Shavlik. Heuristically expanding knowledge-based neural networks. In: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, v. 1, p. 1350–1365, Chambery, France, 1993.

[144] F.S. Osório and B. Amy. Inss: Um sistema híbrido simboli-conexionista com aprendizado à partir de regras e de exemplos. Site: *http//lifia.imag.fr/RESEAUX/public.html*, Paris, France, 1997.

[145] S.K. Pal and S. Mitra. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on Neural Networks*, v. 3, n. 5, p. 683–697, 1992.

[146] D. Park and A. Kandel. Genetic-based new fuzzy reasoning models with application to fuzzy control. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 24, n. 1, p. 39–47, 1994.

[147] D. Parker. Learning logic. Invention Report, Stanford University, File 1, Office of Technology Licensing, p. S81–64, Stanford, Californ, 1982.

[148] E.L. Passos. *Inteligência Artificial e Sistemas Especialistas ao Alcance de Todos.* Livros Técnicos e Científicos Editora, Rio de Janeiro, R.J., 1989.

[149] W. Pedrycz. A referencial scheme of fuzzy decision making and its neural network structure. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 21, n. 6, p. 1593–1604, 1991.

[150] W. Pedrycz. Neural networks: Concepts and architectures. *SBA Controle & Automação*, v. 4, n. 3, p. 126–140, 1994.

[151] W. Pedrycz and A.F. Rocha. Fuzzy-set based models of neurons and knowledge-based networks. *IEEE Transactions on Fuzzy Systems*, v. 1, n. 4, p. 254–266, 1993.

[152] E. Pop, R. Hayward, and J. Diederich. RULENEG: extracting rules from a trained ann by stepwise negation. *QUT NRC*, 1994.

[153] M.A. Rabuske. *Introdução à Teoria dos Grafos.* Editora da UFSC, Florianópolis, Brasil, 1992.

[154] R.A. Rabuske. *Inteligência Artificial.* Editora da UFSC, Florianópolis, Brasil, 1995.

[155] J.A. Reggia. Artificial neural systems in medical science and practice. *MD Computing*, v. 5, n. 3, p. 4–6, 1988.

[156] H. Reichgelt. *Knowledge Representation: An AI Perspective.* Ablex Publishing Corporation, Norwood, New Jersey, 1991.

[157] E. Rich and K. Knight. *Inteligência Artificial.* MAKRON Books do Brasil Editora Ltda., Rio de Janeiro, R.J., 1994.

[158] M. Roisenberg. *Emergência de Inteligência em Agentes Autônomos Através de Modelos Inspirados na Natureza.* Tese de Doutorado, UFSC, Florianópolis, Brasil, 1998.

[159] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In: *Parallel Distributed Processing*, v. 1: Foundations, p. 319–362. MIT Press, Cambridge, Massachusetts, 1986. D.E. Rumelhart and J.L. McClelland and The PDP Group (eds.).

[160] K. Saito and R. Nakano. Medical diagnostic expert system based on PDP model. In: *Proceeding of the International Conference on Neural Networks*, v. 1, p. 255–262, Sao Diego, CA, 1991.

[161] A.C. Scott, J.E. Clayton, and E.L. Gibson. *A Practical Guide to Knowledge Acquisition.* Addison-Wesley Publishig Co., Reading, MA, 1991.

[162] K. Sestito and T. Dillon. The use of sub-symbolic methods for the automation of knowledge acquisition for expert systems. In: *Proceeding of the 11th International Conference on Expert Systems and Their Applications - AVIGNON'98*, p. 317–328, Avignon, France, 1991.

[163] E.H. Shortliffe. *MYCIN: A Rule-Based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection.* Ph.D. Thesis, Stanford University, California, USA, 1974.

[164] E.H. Shortliffe. *Computer-based medical consultations: MYCIN.* Elsevier Science Publishers B.V., New York, USA, 1976.

[165] N.A. Sigaki. IAC models. Tese de Mestrado, Florianópolis, Brasil, UFSC, 1997.

[166] P.R. Simpson. *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementation.* Pergamon Press, Inc., New York, USA, 1990.

[167] P.R. Simpson. Fuzzy min-max neural networks - part 1: Classification. *IEEE Transactions on Neural Networks*, v. 3, n. 5, p. 776–800, 1992.

[168] M. Sirivas and L.M. Patnaik. Genetic algorithms: A survey. *IEEE Computer*, v. 27, n. 6, 1994.

[169] B. Soucek and The IRIS Group. *Neural and Intelligent Systems Integration: Fifth and Sixth Generation Integrated Reasoning Information Systems.* John Wiley & Sons, Inc., New York, USA, 1991.

[170] M.C. South, G.B. Wetherill, and M.T. Tham. Hitch-hiker's guide to genetic algorithms. *Journal of Applied Statistics*, v. 20, n. 4, p. 153–175, 1993.

[171] M. Srinivas and L.M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 24, n. 4, p. 656–667, 1994.

[172] C.T. Sun. Rule-based structure identification in an adaptive - network-based fuzzy inference system. *IEEE Transactions on Fuzzy Systems*, v. 2, n. 1, p. 64–73, 1994.

[173] R. Sun. The discrete neuronal model and the probabilistic discrete neuronal models. In: *Neural and Intelligent Systems Integration*, p. 161–179. John Wiley & Sons, Inc., New York, USA, 1991. B. Soucek and The IRIS Group (eds.).

[174] R. Sun. Neural network models for rule-based reasoning. In: *Proceedings of the IEEE International Joint Conference on Neural Networks*, v. 1, p. 503–508, New York, USA, 1991.

[175] R. Sun. Connectionist models of rule-based reasoning. *AISBQ*, v. 1, n. 79, p. 21–24, 1992.

[176] R. Sun and D. Waltz. A neurally inspired massively parallel model of rule-based reasoning. In: *Neural and Intelligent Systems Integration*, p. 341–381. John Wiley & Sons, Inc., New York, USA, 1991. B. Soucek and The IRIS Group (eds.).

[177] J. Tanomaru. Motivação, fundamentos e aplicações de algoritmos genéticos. In: *Proceedings do II Congresso Brasileiro de Redes Neurais*, v. 1, p. 331–411, Curitiba, Brasil, 1995.

[178] S.B. Thrun. Extracting provably correct rules from artificial neural networks. Relatório técnico IAI-TR-93-5, Institut fur Informatik III Universitat, Bonn, Germany, 1994.

[179] A.B. Tickle, M. Orlowski, and J. Diederich. DEDEC: decision detection by rule extraction from neural networks. *QUT NRC*, 1994.

[180] H. Tirri. Implementing expert system rule conditions by neural networks. *New Generation Computing*, v. 10, n. 1, p. 55–71, 1991.

[181] G.G. Towell. *Symbolic Knowledge and Neural Networks: Insertion, Refinement and Extraction.* Ph.D. Thesis, University of Wisconsin, Wisconsin, Madison, 1991.

[182] G.G. Towell and J.W. Shavlik. The extraction of refined rules from knowledge-based neural networks. *Machine Learning*, v. 131, n. 1, p. 71–101, 1993.

[183] G.G. Towell and J.W. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, v. 69, n. 1, 1994.

[184] C. Townsend. *Introduction to turbo prolog.* SYBEX, Inc., Berkeley, CA, 1987.

[185] V. Tresp, J. Hollatz, and S. Ahmad. Network structuring and training using rule-based knowledge. In: *Advances in Neural Information Processing Sistems*, p. 1–10. Kluwer Academic Publishers, San Mateo, CA, 1993. C.L. Giles and S.I. Hanson and J.D. Cowan (eds.).

[186] E. Turban. *Expert Systems and Applied Artificial Intelligence.* Macmillian Publishing Company, Inc., New York, USA, 1992.

[187] G.S. Tuthill. *Knowledge Engineeering - Concepts and Practices for Knowledge-Based Systems.* TAB Books, Inc., Blue Ridge Summit, PA, 1990.

[188] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Ph.D. Thesis, Harvard University, Cambridge, Massachusetts, 1974.

[189] B. Widrow. Generation and information storage in networks of adaline neurons. In: *Self-Organizing Systems*, p. 435–461. Spartan Books, Washington, D.C., 1962. M. Yovits and G. Gacobi and G.Goldstein (eds.).

[190] L.A. Zadeh. Fuzzy sets. *Information and Control*, v. 8, n. 1, p. 338–353, 1965.

[191] L.A. Zadeh. Commonsense knowledge representation based on fuzzy logic. *IEEE Computer*, v. 16, n. 10, p. 61–65, 1983.

[192] L.A. Zadeh. Fuzzy logic. *IEEE Computer*, v. 21, n. 4, 1988.

[193] M.B. Zanusso. *Famílias de T-Normas Diferenciáveis, Funções de Pertinência Relacionadas e Aplicações.* Tese de Doutorado, UFSC, Florianópolis, Brasil, 1997.

[194] X. Zhang, C. Hang, S. Tan, and P.Z. Wang. The min-max function differentiation and training of fuzzy neural networks. *IEEE Transactions on Neural Networks*, v. 7, n. 5, p. 1139–1150, 1996.

[195] H.J. Zimmermann. *Fuzzy Set Theory - and Its Applications.* Kluwer Academic Publishes, Norwell, Massachusetts, 1991.

[196] J.M. Zurada. *Introduction to Artificial Neural Systems.* West Publishing Company, Inc., St. Paul, MN, 1992.