


O MODELO CONEXIONISTA EVOLUTIVO

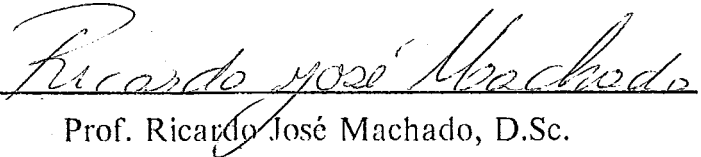
Fernando Antônio Rivas Maximus Denis

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

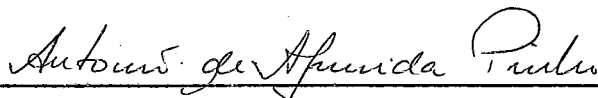
Aprovada por:



Profa. Suely Bandeira Teixeira Mendes, Ph.D.
(Presidente)



Prof. Ricardo José Machado, D.Sc.



Prof. Antonio de Almeida Pinho, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

JULHO DE 1991

DENIS, FERNANDO ANTONIO RIVAS MAXIMUS

O Modelo Conexcionista Evolutivo [Rio de Janeiro] 1991

xi, 138 p., 29,7cm. (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1991)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Sistemas Especialistas 2. Modelos Conexcionistas 3. Algoritmos Genéticos 4. Aprendizado por Máquina I. COPPE/UFRJ II. Título (série).

"Penso que só há um caminho para a ciência ou a filosofia: encontrar um problema, ver a sua beleza e apaixonar-se por ele, casar e viver feliz com ele até que a morte vos separe - a não ser que encontrem um outro problema ainda mais fascinante, ou, evidentemente, a não ser que obtenham uma solução. Mas, mesmo que obtenham uma solução, poderão então descobrir a existência de toda uma família de problemas-filhos, encantadores ainda que talvez difíceis, para cujo bem-estar poderão trabalhar, com um sentido até o fim dos vossos dias."

Karl Popper

Aos meus pais

Meus Agradecimentos:

- Ao Prof. Ricardo J. Machado, meu orientador, pelo apoio, dedicação e experiente orientação. A ele, um especial agradecimento por ter confiado em mim, apesar de todas as adversidades e dificuldades, e proporcionado a oportunidade de juntos desenvolvermos este trabalho. Inclusive, o fato de ter sido recebido em sua casa como um dos seus é algo que jamais esquecerei.
 - Ao Prof. Antônio de Almeida Pinho, pela orientação inicialmente prestada na pesquisa da tese. Pelo incentivo na elaboração do meu primeiro "paper", e no apoio constante ao longo da tese.
 - A Profa. Suely B. T. Mendes, por ter me apresentado o campo dos modelos conexionistas, e por ter aceito o convite para presidir a banca de defesa.
 - Ao Prof. Tarcísio Haroldo Cavalcante Pequeno, pelo apoio e incentivo para que eu viesse fazer mestrado.
-
- A minha mãe e às minhas irmãs pelo apoio nos momentos, não raros, de dificuldades e tensão.
 - Aos meus amigos da COPPE, André, Eliana, Edson, Célia, João, Adriano, Marcos (Babilônia), Gerardo, Einstein, Cláudia, Jussara, Márcia, Clevi, e outros mais, pela amizade, companheirismo e descontração
 - Aos meus amigos do Centro Científico Rio, que proporcionaram um ambiente de amizade emocionante e acolhedor. Tenho, por todos vocês que compartilharam estes dois anos, um carinho muito grande. Considero vocês a minha segunda família. Obrigado pela lição de vida.

- A Chulamis e a turma da psicologia, pelo suporte psicológico, amizade e confiança que muito me ajudaram.
- Aos meus companheiros de apartamento: Rui, Wamberto, Hércules, Samuel, Adelina, Almir, André e Carlos pela amizade e apoio.
- A CAPES e ao CNPq, pelo suporte financeiro concedido durante o curso de mestrado.
- A IBM-Brasil pelas excelentes condições de trabalho, dentro das quais esta tese foi desenvolvida.
- E a todos que, de alguma forma, contribuíram para a realização deste trabalho.

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

O MODELO CONEXIONISTA EVOLUTIVO

Fernando Antônio Rivas Maximus Denis

Julho de **1991**

Orientador: Ricardo José Machado

Co-orientadora: Suely Bandeira Teixeira Mendes

Programa: Engenharia de Sistemas e Computação

Visando amenizar os problemas de fragilidade e de dificuldade de aquisição de conhecimento, existentes nos sistemas especialistas, adotamos uma arquitetura híbrida, combinando processamento simbólico com redes neurais, lógica nebulosa e algoritmos genéticos, capaz de propiciar capacidade de aprendizado a esses sistemas.

Neste trabalho, propomos o Modelo Conexionista Evolutivo (MCE) que utiliza, como base para representação do conhecimento e aprendizado, o Modelo Neural Combinatório (MNC). Para descobrir a topologia adequada da rede neuronal, empregamos um algoritmo genético que visualiza as vias de raciocínio dessa rede como elementos de uma população em evolução.

Dentre as características principais do MCE destacamos: capacidade de refinar um conhecimento anteriormente armazenado na rede neuronal artificial (RNA), através da busca de uma topologia adequada para a RNA; o uso de múltiplos operadores genéticos; incorporação de heurísticas gerais visando realizar operações genéticas mais efetivas; possibilidade de evolução dos operadores genéticos através do algoritmo meta-genético, que determina a população ideal de operadores e a frequência com que devem ser usados.

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

AN EVOLUTIVE CONEXIONIST MODEL

Fernando Antônio Rivas Maximus Denis

July, 1990

Thesis Supervisor: Ricardo José Machado

Thesis Co-supervisor: Suely Bandeira Teixeira Mendes

Department: Systems Engineering and Computing

Aiming to solve the problems presented by expert systems, such as brittleness concerning the problem domain boundaries and the knowledge acquisition bottleneck, we adopted a hybrid architecture combining symbolic processing with neural networks, fuzzy logic and genetic algorithms, able to provide a learning ability to these systems.

In this thesis, we propose the Evolutive Connectionist Model (ECM), using as basis for knowledge representation and learning, the Combinatorial Neural Model. For discovering the adequate topology for the neural network, we used a genetic algorithm that considers the reasoning pathways of the net as elements of an evolving population.

The main characteristics of the ECM are: ability to refine a previously knowledge stored in the neural network by searching the adequate topology for it; use of multiple genetic operators; incorporation of general heuristics aiming to perform the most effective genetic operations; evolution of genetic operations through the use of a metagenetic algorithm, that finds the ideal population of operators and the frequency they should be used.

ÍNDICE

Capítulo I - Introdução	1
Capítulo II - Modelos Conexionistas	10
11.1 - Componentes dos Modelos Conexionistas	12
11.2 - Aprendizado em Modelos Conexionistas	19
11.3 - Dinâmica: Convergência e Estabilidade	23
11.4 - Aplicações e Principais Modelos Conexionistas	26
Capítulo III - Sistemas Especialistas Conexionistas	33
III.1 - Aplicações na Área de Sistemas Especialistas	34
III.2 - O Modelo Neural Combinatório	39
III.2.1 - O Algoritmo de Aprendizado	42
III.2.2 - O Problema do João, Maria, Pedro e Diana	44
Capítulo IV - Algoritmos Genéticos	46
IV.1 - Componentes dos Algoritmos Genéticos	49
IV.2 - Teorema Fundamental dos Algoritmos Genéticos	60
IV.3 - Aplicações dos Algoritmos Genéticos	61
IV.3.1 - Visão Geral do Sistema Classificador	63
IV.3.1.1 - Sistemas Classificadores Propriamente Ditos (SCPDs)	65
IV.3.1.2 - Sistema de Atribuição de Crédito (SAC)	67
IV.3.2 - Integrando AGs e Modelos Conexionistas	69
IV.3.2.1 - Elementos dos AGs Propostos	70
IV.3.2.2 - Principais Características dos Trabalhos	72
Capítulo V - O Modelo Conexionista Evolutivo	76
V.1 - Arquitetura do MCE	77
V.2 - Atuação do Algoritmo Genético no MCE	83
V.2.1 - Operadores Genéticos	87
V.2.1.1 - Exemplos de Operadores Genéticos	88
V.2.2 - Comparação entre as duas versões do MCE	92

V.3 - Atuação do Algoritmo Meta-Genético no MCE	93
V.4 - Diferenças entre o MCE e o AG Clássico	95
V.4.1 - Simulando as Operações Genéticas do MCE no AG Clássico	98
Capítulo VI - Descrição do Ambiente de Experimentação	100
VI.1 - Descrição do NEXT	100
VI.1.1 - A Arquitetura do NEXT	101
VI.1.2 - Método de Construção de SEs no NEXT	104
VI.2 - Medição do Erro em Sistemas Classificatórios	107
Capítulo VII - Avaliação do Modelo Conexionista Evolutivo	110
VII.1 - O Problema do OU-Exclusivo	112
VII.2 - O-Problema do João e Maria	113
VII.2.1 - Rede Treinada com ordem 1	114
VII.2.2 - Rede Treinada com ordem 3	116
VII.2.3 - Rede Treinada ordem 1 com ligações invertidas	117
VII.3 - O Problema do 2-3 Aglomerados	119
Capítulo VIII - Conclusões	122
Referências Bibliográficas	126
Apêndice I	136

FIGURAS

1.	Próxima geração de SEs (SIHs)	6
2.	Neurônio como unidade limiar	13
3.	Exemplo de RNA	15
4.	Exemplos de funções de ativação	17
5.	Representação de um mínimo local	24
6.	Versão completa do MNC para 3 evidências e 2 hipóteses	40
7.	Rede podada usando-se o limiar 0.4 no problema do	45
8.	Padrão de um Algoritmo Genético	48
9.	Exemplo de operação de recombinação	56
10.	Operação dos operadores genéticos	57
11.	Organização Geral de um Sistema Classificador	64
12.	Elementos Básicos do SCPDs	66
13.	Codificando uma rede em um cromossomo	72
14.	Comparação do AG de Montana com o "backpropagation"	73
15.	Rede obtida pelo GENITOR para o somador de bits de 2 posições	75
16.	Arquitetura do MCE	79
17.	Exemplo de reprodução de vias	80
18.	Operação de Mutação por Eliminação	89
19.	Operação de Mutação por Substituição	90
20.	Operação de Mutação por Adição	91
21.	Operação de Recombinação	91
22.	Arquitetura do NEXT	102
23.	Gráfico de evolução das taxas de erro para o OU-exclusivo	113
24.	Rede treinada de ordem 1 para o problema do João e Maria	114
25.	Gráfico da evolução das taxas de erro para o problema João e Maria ordem 1	115
26.	Gráfico da evolução das taxas de erro para o problema João e Maria	117
27.	Rede inicial com ligações invertidas	117
28.	Gráfico da evolução das taxas de erro para o João e Maria	119

29. Gráfico da evolução das taxas de erro para o problema 2-3
aglomerados. 121

TABELAS

1.	Descrição Geral dos Paradigmas de AM	5
2.	Diferenças entre o computador tradicional e o cérebro humano	11
3.	Funcionalidade do aprendizado em RNAs	20
4.	Principais técnicas de aprendizado em RNAs	21
5.	Apresentação dos principais modelos conexionistas	28
6.	Características dos principais modelos conexionistas	29
7.	Apresentação dos SEs Conexionistas	34
8.	Características das redes dos SEs Conexionistas.	35
9.	Comparação entre os elementos que compõem os AGs de Montana e Whitley.	71
10.	Comparação entre o Sistema Classificador de Holland e o MCE	82
11.	Diferenças entre o MCE e o AG Clássico	96
12.	Resumo dos resultados para o problema do OU-exclusivo	112
13.	Resumo dos resultados para o João e Maria ordem 1	115
14.	Resumo dos resultados para o problema João e Maria ordem 3	116
15.	Resumo dos resultados para o problema João e Maria ordem 1	118
16.	Resumo dos resultados para o problema 2-3 aglomerados ordem 2.	121

CAPÍTULO I

INTRODUÇÃO

A inteligência humana sempre fascinou o homem ao longo do tempo. Questões sobre como pensamos, como aprendemos, como raciocinamos e como armazenamos conhecimento, têm recebido a atenção de várias ciências, como a Filosofia, a Psicologia e a Neurologia, há muitos anos.

Na década de 40, com o advento dos computadores, a idéia de se criar uma *máquina inteligente* tornou-se plausível. Em 1956, McCarthy cunhou o termo Inteligência Artificial para este ramo de pesquisa dentro da ciência da computação [PINH88]. A definição que apresentamos a seguir, extraída de Barr e Feigenbaum [BARR81], é bem representativa sobre esta área:

Inteligência Artificial (IA) é a parte da ciência da computação que visa projetar sistemas de computação inteligentes, isto é, sistemas que exibam características que associamos com inteligência no comportamento humano (compreensão de linguagem, aprendizado, raciocínio, resolução de problemas, etc).

Jackson [JACK86] apresenta a evolução da IA ao longo do tempo. Ele destaca três grandes períodos a saber:

1. Clássico - jogos e prova de teoremas (década de 50 até meados de 60);
2. Romântico - compreensão por computador (meados de 60 até meados de 70);
3. Moderno - técnicas e aplicações (meados de 70 até hoje).

No período clássico, grandes progressos foram obtidos em relação a programas que jogassem xadrez e outros que provassem teoremas. Nesta época surgiram os primeiros algoritmos de busca (busca em largura e em profundidade dentre outros). É também desta época o "General Problem Solver" (GPS), projetado por Newell e Simon [NEWE63], com o objetivo de ser um solucionador de problemas independente do domínio de aplicação.

No período romântico, as pesquisas se direcionaram para a tentativa de se produzir sistemas capazes de compreender a linguagem natural no chamado processamento de linguagem natural. Os primeiros sistemas capazes de compreender histórias e diálogos foram projetados neste período. Um exemplo de sistema desta época é o SHRDLU de Winograd [WINO72]. Neste período surgem as pesquisas em *representação do conhecimento* e diversas propostas são apresentadas ([QUIL68], [MINS75]).

No período moderno, a orientação tem sido no sentido de se desenvolver técnicas e aplicações. A desilusão com métodos gerais para solução de problemas acarretou uma mudança conceitual no tratamento dos problemas. As pesquisas mostraram que tais métodos subestimam o senso comum, incluindo a habilidade que os seres humanos possuem de evitar, identificar e corrigir erros [JACK86]. A idéia de que o poder heurístico de um solucionador de problemas reside na representação do conhecimento especializado que o programa pode acessar, e não em algum mecanismo de inferência foi ganhando mais e mais adeptos.

As pesquisas se direcionaram no sentido de entender o funcionamento da mente humana, e de procurar métodos dependentes do domínio de conhecimento, na busca de soluções para os problemas [GOTT90]. Tais métodos têm sido aplicados a domínios de conhecimento específicos (Medicina, Agronomia, Geologia, etc) nos chamados Sistemas Especialistas (SEs) ([HAYE83], [WATE86], [JACK86]).

SEs são programas de computador que simulam as capacidades que especialistas humanos têm de resolver problemas de um determinado domínio de conhecimento. É desejável que atuem de uma forma cuja competência se

aproxime a destes especialistas. Eles retiram o seu poder da grande quantidade de conhecimento que armazenam em suas bases de conhecimento. Eles são capazes de executar inferências sobre suas bases de conhecimento com o objetivo de fornecer soluções para os problemas propostos pelos usuários [MICH87].

Portanto, extrair o conhecimento dos especialistas passou a ser um dos principais alvos das pesquisas em IA. A transferência do conhecimento especializado para um formalismo de representação do conhecimento adequado, compreensível pela máquina, recebeu o nome de aquisição de conhecimento [GOTT90], surgindo a profissão de engenheiro de conhecimento. Este processo se revelou difícil e dispendioso, sendo que a qualidade final dos SEs está diretamente relacionada com ele. Uma alternativa promissora para automatizar este processo, como forma de torná-lo menos oneroso, é o desenvolvimento de um processo de *aprendizado por máquina* (AM) [MICH87]. A importância de AM para o desenvolvimento de SEs já foi citada por vários autores (ver Waltz et alli [WALT83]).

AM se constitui hoje em um importante ramo de pesquisa dentro da IA. AM, bem como representação de conhecimento, cobre todas as áreas de problemas da IA (prova de teoremas, raciocínio não-monotônico, processamento de linguagem natural, reconhecimento de padrões, visão, robótica, jogos, etc). Conseqüentemente, o progresso em AM pode influenciar todas estas áreas.

Acrescente-se, ainda, a motivação filosófica de se estudar o aprendizado. O interesse de se pesquisar o aprendizado existe desde os tempos de Platão. O desenvolvimento de modelos computacionais de aprendizado pode ajudar filósofos e psicólogos a compreender como os seres humanos aprendem, e, conseqüentemente, a compreender o conhecimento e como ele se desenvolve [FEIG82].

Uma questão importante a ser considerada dentro do estudo do aprendizado humano diz respeito ao refinamento do conhecimento adquirido, chamado de refinamento de perícia por alguns autores (ver Carbonell et alli [CARB83] e Feigenbaum [FEIG82]). Enquanto a essência da aquisição do conhecimento

pode ser considerada como um processo consciente cujo resultado é a criação de novas estruturas de conhecimento simbólicas, o processo de refinamento de perícia ocorre em função da prática repetitiva, sem um esforço consciente. O aprendizado humano parece ser uma mistura de ambos os processos.

O principal objetivo científico das pesquisas em AM é o estudo dos possíveis mecanismos de aprendizado. Deste estudo fazem parte a descoberta de diferentes algoritmos de indução, a definição das limitações teóricas das técnicas de aprendizado propostas, os problemas e as soluções encontrados em lidar com dados de treinamento imperfeitos, e a criação de técnicas de aprendizado aplicáveis a qualquer tipo de domínio [CARB83].

O desenvolvimento de sistemas de aprendizado para propósito geral é um objetivo a longo prazo devido à grande complexidade dos processos envolvidos. Entretanto, com o desenvolvimento dos SEs, a implementação de algumas formas de AM passou a ser, do ponto de vista econômico, uma tarefa extremamente necessária [MICH87].

Existem diversas taxonomias para as pesquisas em AM. Em geral, os critérios de classificação consideram a estratégia de aprendizado utilizada, o tipo de conhecimento adquirido e o domínio de aplicação ([FEIG82], [CARB83], [MICH87]). Carbonell [CARB89] considera a existência de quatro grandes paradigmas de AM: Indutivo, Analítico, Genético e Conexionista. Uma descrição dos quatro é apresentada na tabela 1. Uma sub-classificação destes paradigmas refere-se ao tipo de controle realizado durante o aprendizado (supervisionado ou não-supervisionado). No supervisionado existem, a priori, rótulos que especificam a classe correta para os padrões de entrada (instâncias), e no não-supervisionado não existem tais rótulos [LIPP87].

Como resultado destas pesquisas, diversos sistemas que utilizam AM começaram a surgir ([WINS75], [LAIR86], [DAVI87], [RUME86A]). A perspectiva é que ocorra uma integração cada vez maior entre as diversas técnicas de AM, com o objetivo de se desenvolver sistemas cada vez mais robustos ([MONT89], [MUHL90], [WHIT90]).

Tabela 1. Descrição Geral dos Paradigmas de AM

PARADIGMA	DESCRIÇÃO GERAL
Indutivo	Induz a descrição de um conceito geral a partir de exemplos e contra-exemplos deste conceito. A tarefa é construir uma descrição do conceito na qual os exemplos possam ser inferidos por uma instância universal, de tal forma que nenhum dos contra-exemplos possa ser deduzido pelo mesmo processo.
Analítico	Utiliza poucos exemplos e um domínio de base teórica. Os métodos são dedutivos. O conhecimento existente é utilizado para resolver problemas servindo de guia nas deduções dos novos problemas, e para formular regras de controle de busca que tornem mais eficiente a aplicação do domínio de conhecimento.
Genético	É inspirado na analogia com o processo de evolução biológica das espécies, e na seleção natural de Darwin. Variações das descrições dos conceitos correspondem a indivíduos de uma espécie, e combinações destes indivíduos são testados contra uma função objetivo (critério de seleção). Os algoritmos genéticos codificam uma busca paralela através do espaço de conceitos, onde cada processo tenta maximizar a função objetivo.
Conexionista	Realiza o aprendizado mediante o ajuste de pesos em uma rede neuronal utilizando um algoritmo de aprendizado. Alguns modelos deste paradigma aprendem a reconhecer instâncias de uma maneira supervisionada e outros de uma maneira não-supervisionada.

Corroborando a idéia de integração dos diversos paradigmas de AM, Kandel [KAND91] apresenta a proposta dos *Sistemas Inteligentes Híbridos* (SIHs) que combinariam os diversos modelos, conforme apresentado na figura 1.1 (baseada em Schwartz [SCHW90]), sendo considerados a próxima geração de SEs.

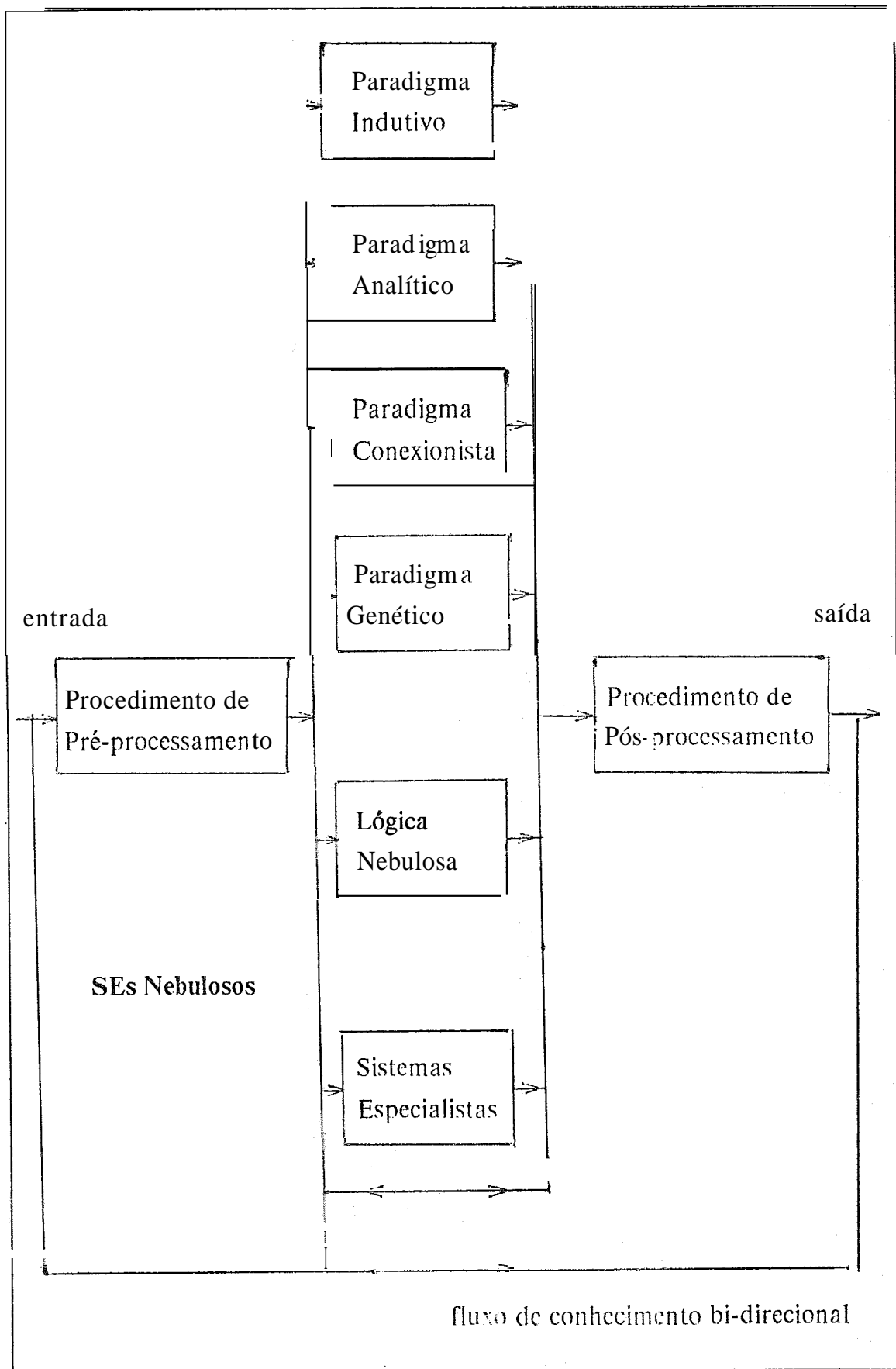


Figura 1. Próxima geração de SEs (SIIs)

A lógica nebulosa, ou teoria dos conjuntos difusos, foi proposta por Zadeh [ZADE65]. Esta técnica de raciocínio aproximado tem sido utilizada nos SEs baseados em regras de produção, os chamados SEs Nebulosos [MONA88], e em SEs que a combinam com o paradigma conexionista para o tratamento de informações vagas, incertas e incompletas ([MACH89], [ROMA89]), frequentemente associadas ao raciocínio humano.

Em muitos casos, a incerteza pode ser associada ao fato de que os conceitos básicos envolvidos no raciocínio humano não podem ser modelados por valores do tipo falso e verdadeiro, mas por rótulos de conceitos nebulosos, sem um limite preciso a separá-los (por exemplo pequeno, médio e grande). Estes conceitos nebulosos podem ser considerados como classes de objetos, onde a transição de uma determinada instância x pertencente a uma classe (por exemplo *pequeno*), para uma outra classe (por exemplo médio) se processa de maneira gradual e não determinística.

A Teoria dos Conjuntos Difusos utiliza uma função de pertinência como uma medida variável do *grau* de possibilidade que uma instância x qualquer possui em relação ao conceito de uma determinada classe de objetos (característica) [PAO89]. Esta distribuição de possibilidade não deve ser confundida com distribuição de probabilidade, ou seja, o que é impossível também é improvável, mas o que é possível nem sempre é provável [ZADE78].

Uma característica importante dos SIHs é a premissa de transferência de conhecimento bi-direcional [KAND91]. O aprendizado no paradigma conexionista é implementado de uma maneira mais eficiente porque os SEs fornecem o conhecimento inicial para começar este processo. Os SEs podem inferir novas regras, ou modificar as existentes, baseados na informação adquirida pelo paradigma'conexionista. Acrescente-se o fato de que o uso de lógica nebulosa permite aos SIHs considerar o grau que cada regra agrega de acordo com a compreensão corrente da realidade, e não na probabilidade de que esta regra seja a descrição verdadeira desta realidade.

Dentro desta linha de raciocínio foi desenvolvida esta pesquisa. Foi feita a combinação de um exemplo do paradigma genético com um exemplo do

conexionista. É proposto o *Modelo Conexionista Evolutivo* que é um tipo de modelo que permite refinar o conhecimento inicial, através da criação de novos caminhos dentro da rede, sem a perda das capacidades do conhecimento já adquirido.

A tese está estruturada da seguinte forma:

No capítulo II é descrito o paradigma conexionista. São apresentados os conceitos básicos dos modelos conexionistas e as propriedades de alguns dos principais modelos existentes na literatura.

Em seguida são tecidas considerações acerca do uso dos modelos conexionistas em SEs. São apresentadas algumas aplicações encontradas na literatura, enfatizando o *Modelo Neural Combinatório* [MACH89], que foi utilizado nesta pesquisa.

No capítulo IV são apresentados os algoritmos genéticos, sendo feita uma revisão sobre sistemas classificadores e sistemas de atribuição de crédito. São apresentados o teorema fundamental dos algoritmos genéticos e o teorema de propagação (paralelismo implícito) propostos por Holland [HOLL75]. Concluindo este capítulo, são mostrados exemplos de integração dos algoritmos genéticos com modelos conexionistas.

Dando prosseguimento é descrita a proposta da construção de um modelo conexionista evolutivo no capítulo V. Este sistema é resultado da aplicação de algoritmos genéticos em um SE construído segundo o modelo neural combinatório [MACH89]. São apresentadas as estruturas dos algoritmos genético e meta-genético implementados, destacando a atuação de cada algoritmo no sistema.

No capítulo VI é apresentado o ambiente de experimentação no qual deverá ser aplicado o sistema descrito no capítulo IV. É apresentada uma descrição do NEXT, que é a ferramenta onde foi implementada o algoritmo genético, enfatizando sua arquitetura e seu método de construção de SEs. Por último

são descritos os índices utilizados na medição do erro em sistemas classificatórios, categoria onde está inserida a experimentação.

No capítulo seguinte é feita a avaliação do modelo conexionista evolutivo. São reportados os resultados dos experimentos realizados. São descritas as redes, construídas segundo o modelo neural combinatório e combinadas com algoritmos genéticos, que foram aplicadas em três problemas específicos constantes na literatura (OU-exclusivo, João e Maria, e 2-3 aglomerados).

Encerrando a tese, são apresentadas as conclusões do trabalho e direções que orientem pesquisas futuras.

CAPÍTULO II

MODELOS CONEXIONISTAS

Os modelos conexionistas se fundamentam nos estudos sobre a estrutura do cérebro humano para tentar emular sua forma inteligente de processar informação. Alguns estudos da neurofisiologia consideram que a riqueza computacional do cérebro humano vem do grande número de neurônios que estão interconectados por uma rede complexa de sinapses [CARV88].

Observando a *máquina inteligente* que é o cérebro humano, estima-se que a quantidade de neurônios existentes no mesmo está na casa dos bilhões. A velocidade de processamento destes componentes é baixa se comparada com a velocidade dos computadores tradicionais. Esta deficiência é superada pela imensa quantidade de neurônios existentes que operam em paralelo [SIMP90]. Estima-se que existam cerca de 10^{11} a 10^{14} neurônios operando em paralelo no cérebro humano. Cada um destes está conectado através de 10^3 a 10^4 sinapses em média [COTT85].

Tais características permitem ao cérebro humano executar rapidamente certas funções (por exemplo: reconhecer fisionomias). que os computadores convencionais não conseguem realizar com o mesmo desempenho. Na tabela 2, baseada em Cottrell [COTT85] e Simpson [SIMP90], é apresentada uma comparação das principais diferenças existentes entre os computadores tradicionais e o cérebro humano. Esta comparação permite ter uma idéia mais clara sobre a capacidade adaptativa do cérebro humano, em contraste com a rigidez e precisão dos computadores convencionais.

Tabela 2. Diferenças entre o computador tradicional e o cérebro humano

	COMPUTADOR TRADICIONAL	CÉREBRO HUMANO
Elementos Computacionais	processadores poderosos	neurônios simples
Velocidade de Processamento	10^{-9} s	10^{-3} s
Tipo de Processamento	serial	paralelo
Confiabilidade dos Elementos	confiável	não-confiável
Tolerância a Falhas	quase nenhuma	grande
Tipo de Sinal	precisos, simbólicos	imprecisos
Tipo de Controle	centralizado	distribuído
Armazenamento de Informação	substituível	adaptável

Uma definição, elaborada por Hecht-Nielsen [HECH88], de modelos conexionistas de computação, também chamados de redes neuronais artificiais (RNAs) redes neurais, ou sistemas de processamento paralelo distribuído (PDP), é apresentada a seguir:

Um modelo conexionista é uma estrutura de processamento de informação distribuída e paralela. Ela é formada por unidades de processamento, comumente chamadas de nós, neurônios ou células, interconectadas por arcos unidirecionais, também chamados de ligações, conexões ou sinapses. Os nós possuem memória local e podem realizar operações de processamento de informação localizada. Cada célula possui uma única saída (axônio), a qual pode se ramificar em muitas ligações colaterais (cada ramificação possuindo o mesmo sinal da saída do neurônio). Formalmente, o sinal de saída do nó pode ser equacionado de diversas maneiras. Todo o processamento que se realiza em cada unidade deve ser completamente local, isto é, deve depender apenas dos valores correntes dos sinais de entrada que chegam dos neurônios através das conexões. Estes valores atuam sobre os valores armazenados na memória local da célula.

Os principais elementos usados na descrição das RNAs são a representação distribuída, as operações locais e o processamento não-linear [SIMP90]. Estes

atributos especificam duas aplicações básicas dos modelos conexionistas: situações onde poucas decisões tem que ser tomadas a partir de uma grande quantidade de dados, e situações onde um complexo mapeamento não linear deve ser aprendido. Geralmente, o ferramental matemático usado nesta tecnologia inclui: equações diferenciais, sistemas dinâmicos, álgebra linear, probabilidade e estatística.

Neste capítulo é feita uma revisão sobre os modelos conexionistas. Na seção II.1 são descritos os componentes básicos destes modelos, e como se interrelacionam. Na seção subsequente, é dedicada uma atenção especial às técnicas de aprendizado utilizadas nos modelos conexionistas. Na seção 11.3 são apresentados os conceitos de convergência e estabilidade. Finalizando este capítulo, são tecidas considerações sobre alguns dos modelos clássicos encontrados na literatura, destacando suas aplicações e propriedades principais.

II.1 - COMPONENTES DOS MODELOS CONEXIONISTAS

De acordo com Rumelhart [RUME86A], um modelo conexionista pode ser descrito por oito elementos principais:

- um conjunto de unidades de processamento;
- um estado de ativação;
- uma função de saída;
- um padrão de interconexão;
- uma regra de propagação;
- uma regra de ativação;
- uma regra de aprendizado; e
- um ambiente onde o sistema deve funcionar.

Unidades de Processamento

As células são o meio de representação do conhecimento, por exemplo conceitos de um domínio, existente na rede neural. Os nós podem representar pontos ("pixels"), caracteres (letras, números), palavras, ou outros conceitos, dependendo da aplicação. Na figura 2 temos uma ilustração de um neurônio como unidade limiar. As entradas que chegam a ele representam os dentritos. Cada dentrito possui um sinal que é adicionado (Σ). Depois da adição, o sinal é processado através de uma função limiar $f()$, a qual produz um sinal de saída. Nesta figura, o neurônio limiar pode ser considerado como uma representação simplificada dos neurônios biológicos.

Nas RNAs existem dois tipos de representação do conhecimento possíveis: a localizada e a distribuída. A representação localizada corresponde a um único neurônio representando um conceito. Na representação distribuída, o padrão de ativação de um conjunto de unidades é que possui significado [RUME86A]. Neste trabalho, as unidades de processamento da rede são designadas pela letra u , seguida de um índice i que indica a posição que o neurônio ocupa na rede.

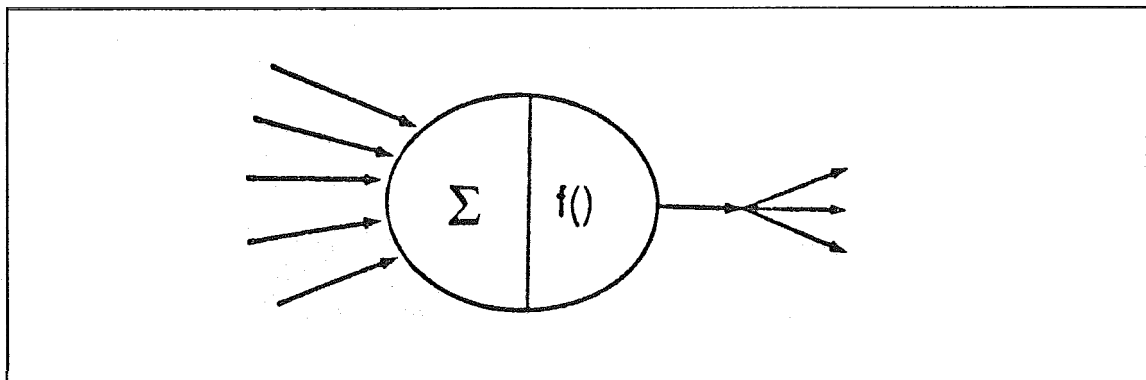


Figura 2. Neurônio como unidade limiar

Estado de Ativação

Cada célula u_i da rede computa um estado de ativação, que é um valor numérico líquido de saída. O cálculo desta ativação é computado a partir das ativações das células conectadas diretamente a este nó, e dos correspondentes pesos destas conexões.

O estado de ativação de todas as unidades da rede, ou seja, o estado de ativação do sistema, especifica o que está sendo representado na rede em um determinado instante t qualquer. Este estado de ativação do sistema pode ser representado por um vetor $a(t)$. Os valores das ativações existentes na rede podem ser discretos, por exemplo assumindo os valores $\{0,1\}$ ou $\{-1,0,+1\}$, como também podem ser contínuos, assumindo valores no intervalo $[0,1]$ ou $[-1,+1]$, que são computados pela regra de ativação a ser vista depois ([LIPP87], [GALLSS]).

Função de Saída

As unidades interagem entre si através de um valor que é transmitido pelas sinapses. Este valor é determinado pela ativação da unidade estimuladora. Formalmente, o valor de saída é dado por uma função do tipo $o_i(t) = g(a_i(t))$.

Padrão de Interconexão

Pode-se representar o padrão de interconexão da rede por uma matriz de pesos W , onde um elemento $w_{i,j}$ corresponde à influência da célula u_i sobre a célula u_j . Conexões com pesos positivos, chamadas de excitatórias, indicam reforço na ativação do neurônio u_j . Sinapses com pesos negativos, chamadas de inibitórias, indicam inibição na ativação da célula u_j . O conjunto das ligações excitatórias e inibitórias existentes na rede determina o comportamento da mesma.

Topologicamente, as RNAs podem ser organizadas em camadas. A camada de entrada da rede é instanciada externamente e não recomputa suas saídas. Portanto, não existem arcos de entrada em suas células. Os valores resultantes das células pertencentes a camada de saída são considerados os resultados finais da rede como um todo. Na figura 3, as unidades u_9 e u_{10} são consideradas as saídas da rede. As células que não pertencem nem à camada de entrada e nem à camada de saída são chamadas de intermediárias ou ocultas.

Observando a figura 3 quando u_{10} é ativado o valor de sua ativação é determinado pelas ativações de u_4 , u_5 , u_7 e u_8 e os pesos $w_{4,8}$, $w_{5,8}$, $w_{3,7}$, $w_{7,10}$ e $w_{8,10}$. Podem existir arcos que conectam nós da mesma camada, sendo chamados de sinapses intra-camadas. As ligações inter-camadas conectam células de camadas diferentes. No caso da rede da figura 3, os arcos que unem as unidades u_6 e u_7 são sinapses intra-camada inibitórias. A ligação que conecta o nó u_7 ao nó u_{10} é inter-camada excitatória.

Com relação aos arcos que conectam as unidades u_6 e u_7 , eles são chamados de ligações recorrentes. Estas conexões formam um ciclo, voltando para o neurônio de origem (aquele que foi ativado primeiro). Podem existir sinapses que ligam unidades de camadas diferentes, formando ciclos entre as mesmas. Por exemplo, uma ligação que conectasse a célula u_{10} a u_5 formaria um outro ciclo na rede da figura 3. Portanto, as RNAs podem ser classificadas em redes cíclicas, que possuem ciclos, e acíclicas, que não possuem ciclos [LIPP87].

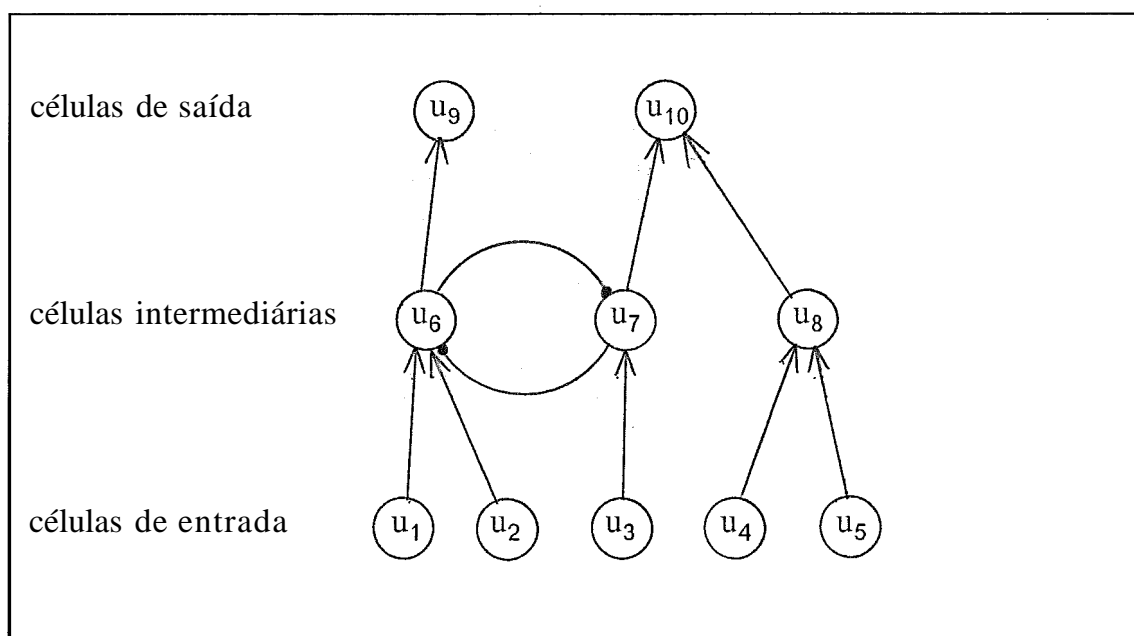


Figura 3. Exemplo de RNA

Regra de Propagação

Cada célula u_i computa sua nova ativação através de uma regra de propagação. Em geral, ela é definida como sendo uma função soma da

entrada líquida dos pesos ("net") das células u_j que estão diretamente conectadas a u_i conforme fórmula abaixo:

$$net_i = F\left(\sum_{j=1}^n w_{i,j}u_j - \Theta_i\right),$$

onde u_j é o estado da j -ésima unidade, $w_{i,j}$ é o peso da conexão da i -ésima para a j -ésima unidade e Θ_i é o limiar da i -ésima unidade. Este limiar, que pode ser nulo inclusive, deve ser superado para que ocorra a ativação da célula.

Existem variações da regra de propagação que utilizam os conceitos da lógica nebulosa proposta por Zadeh [ZADE65]. Geralmente, nestas definições emprega-se os operadores de máximo e mínimo. Estes operadores atuam sobre o chamado *produto "fuzzy"*, entre as entradas e os pesos dos nós, executando as operações de E e OU próprias dos conjuntos nebulosos. Neste caso, os pesos podem ser interpretados como graus de pertinência do conceito representado na entrada para o conceito representado na saída da célula ([ROMA88], [MACH89]).

Regra de Ativação

É necessário uma regra que calcule o valor de ativação de uma unidade no instante t . É preciso uma função f que calcule a nova ativação $a_i(t)$ utilizando as entradas líquidas (net). Geralmente, esta função possui a forma $a_i(t+1) = f(a_i(t), net_i(t))$, onde f é a função de ativação, também chamada de função limiar. Esta função mapeia os neurônios de entrada para um intervalo pré-especificado de saída. As quatro funções de ativação mais utilizadas são: linear, rampa, salto e sigmóide [SIMP90]. Uma ilustração das quatro é apresentada na figura 4.

A função linear, figura 4.(a), possui a seguinte equação: $f(x) = \alpha x$, onde α é uma constante real que regula a intensidade da atividade de x .

A figura 4.(b) representa a função limiar rampa. Ela é uma função linear limitada pelo intervalo $[-\gamma, +\gamma]$, definida pelas seguintes equações:

$$f(x) = \begin{cases} +\gamma & \text{se } x \geq \gamma \\ x & \text{se } |x| < \gamma \\ -\gamma & \text{se } x \leq -\gamma, \end{cases}$$

onde γ indica os valores de saída máximo e mínimo, sendo chamados de pontos de saturação.

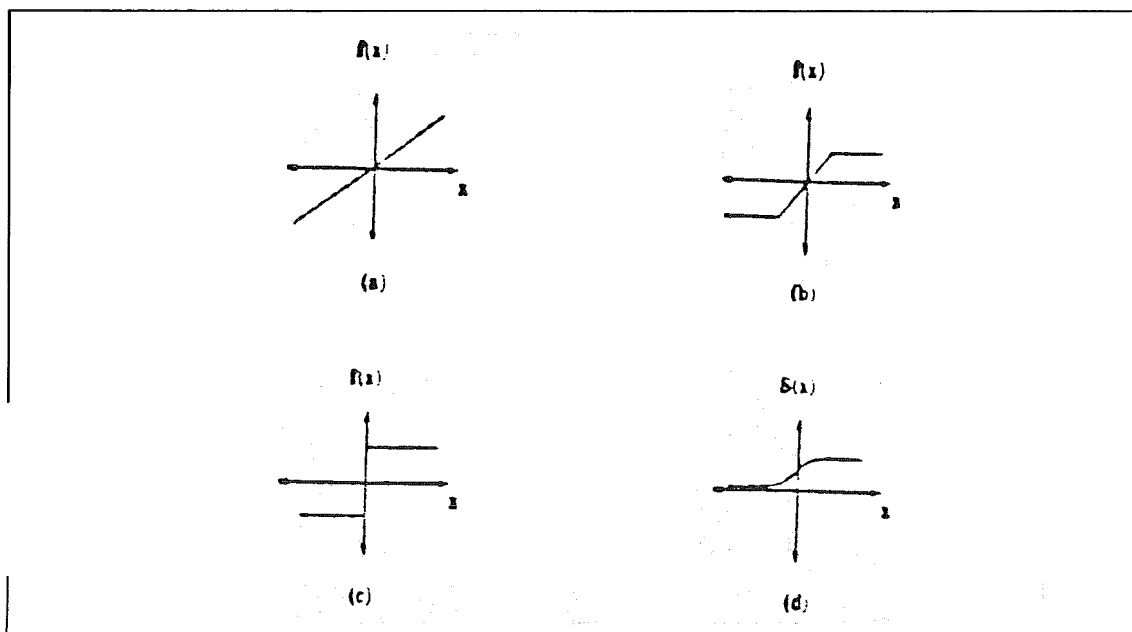


Figura 4. Exemplos de funções de ativação

A função salto, figura 4.(c), responde ao sinal de entrada emitindo o valor $+1$ se o sornatório for positivo, e -1 nos demais casos.

A figura 4.(d) representa a função sigmóide. Esta função limiar é monótona, não-decrescente, e sua resposta é gradual e não-linear. A função sigmóide mais conhecida é a função logística, cujos pontos de saturação são 0 e 1 . A equação que a define é apresentada a seguir:

$$S(x) = (1 + e^{-x})^{-1}.$$

Regra de Aprendizado - modificando a conectividade através da experiência

A modificação do processamento ou da estrutura de conhecimento de uma rede neuronal envolve modificar seu padrão de interconexão [RUME86A]. Em princípio, isto pode ser feito de três maneiras:

1. Desenvolvimento de novas conexões;
2. Perda de conexões existentes na rede; e
3. Modificação dos pesos das conexões já existentes.

Quando o padrão de interconexão for uma matriz de pesos W , os itens (1) e (2) podem ser simulados através de (3). Tomando-se uma ligação com peso zero, e modificando-o para um valor positivo ou negativo, equivale a desenvolver esta sinapse. Da mesma forma, alterar o peso de uma conexão para zero significa desconectá-la. Portanto, as regras de aprendizado alteram os pesos das sinapses das redes através da experiência.

Em geral, as regras de aprendizado podem ser consideradas como uma variante da Regra de Hebb [HEBB49]. Ele estabeleceu o princípio de que a alteração da eficiência sináptica é a base do aprendizado, segundo o postulado apresentado a seguir:

Quando o axônio A se encontra próximo do axônio B de forma a poder excitá-lo, e o faz repetidas vezes, algum processo desconhecido provoca o crescimento de sinapses entre as células A e B, facilitando assim a excitação de B por A.

Especificamente, se uma unidade u_j recebe uma entrada de uma outra u_i , e ambas estão fortemente ativas, o peso $w_{i,j}$ (de u_i para u_j) deve ser fortalecido. Uma extensão desta idéia é apresentada na equação a seguir:

$$\Delta w_{i,j} = g(a_i(t), t_i(t)) \cdot h(o_j(t), w_{i,j}),$$

onde $t_i(t)$ é uma espécie de unidade de entrada *instrutora* (professor) da unidade u_i . Esta equação estabelece que a mudança no peso da conexão de u_j para u_i é resultado do produto da função $g()$ (ativação de u_i e sua entrada *instrutora* t_i) pela função $h()$ (valor de saída de u_j e peso $w_{i,j}$).

Uma variação desta regra apresenta $h(o_j(t), w_{i,j}) = o_j(t)$ e $g(a_i(t)) = \eta \cdot (t_i(t) - a_i(t))$, onde η é uma constante de proporcionalidade que representa o

rateio de aprendizado. Esta regra é chamada de regra delta, pois o aprendizado é proporcional a diferença (delta) entre a ativação realmente encontrada e a ativação fornecida pelo professor. Esta regra é uma generalização da regra de convergência do perceptron, para o qual o *teorema* de convergência do perceptron foi provado.

Devido a sua importância, a próxima seção é dedicada ao aprendizado no contexto dos modelos conexionistas. Considerações acerca da dinâmica dos modelos conexionistas (convergência e estabilidade) são apresentadas na seção 11.3.

Ambiente

O último componente das RNAs é o ambiente onde a rede deve funcionar. É necessário especificar a natureza do ambiente, estabelecendo os possíveis padrões de entrada e de saída. Em alguns modelos, por exemplo o PDP [RUME86A], o ambiente é representado como uma função estocástica que varia ao longo do tempo sobre um espaço de padrões de entrada.

Geralmente, o ambiente é caracterizado como uma distribuição de probabilidade estável sobre um conjunto de padrões de entrada. Esta distribuição pode ser independente, ou não, de entradas ou de respostas passadas (histórico) do ambiente.

II.2 - APRENDIZADO EM MODELOS CONEXIONISTAS

Na sua essência, o conceito de aprendizado envolve mudança associada a aperfeiçoamento [PESS90]. Carbonell [CARB89] define o conceito de aprendizado, dentro do contexto da IA, como a habilidade de realizar tarefas novas que não podiam ser realizadas anteriormente, ou melhorar a realização de tarefas antigas, como resultado de mudanças produzidas pelo processo de aprendizado.

Uma classificação, apresentada por Rumelhart e Zipser [RUME85], relacionou a função do aprendizado em modelos conexionistas. Eles distinguiram quatro paradigmas que são apresentados na tabela 3.

Posteriormente, Rumelhart [RUME86A] apresentou uma outra classificação que é semelhante à apresentada na tabela 3. Neste caso existem apenas dois tipos de aprendizado: o associativo e o detetor de regularidades.

Tabela 3. Funcionalidade do aprendizado em RNAs

PARADIGMA	DESCRIÇÃO GERAL
Auto-Associador	Um conjunto de padrões e repetidamente apresentado e o sistema o armazena. Posteriormente, um padrão, ou parte de um, semelhante aos originais é apresentado. O sistema retoma o padrão original através de uma espécie de procedimento de <i>término de padrão</i> . Este é um processo de auto-associação onde um padrão é associado consigo mesmo, de tal forma que uma versão modificada do original pode servir de <i>deixa</i> para o procedimento de recuperação.
Associador de Padrões	Inicialmente, um conjunto de pares de padrões é apresentado ao sistema. Posteriormente, quando um membro do par é apresentado, o sistema faz a associação, produzindo o elemento correspondente do par.
Classificador	Neste caso há um número fixo de categorias nas quais são classificados os padrões de entrada. O objetivo é ensinar a rede a classificar corretamente os padrões de entrada, de tal forma que, quando apresentado um padrão, mesmo parcialmente modificado, a rede saiba classificá-lo corretamente.
Detetor de Regularidades	Existe uma população de padrões estímulo, sendo que cada membro possui uma probabilidade associada. O sistema desenvolve uma representação das características dos estímulos desta população, a qual captura as propriedades mais salientes dos padrões de entrada. Não existe, a priori, um conjunto de categorias no qual os padrões desta população possam ser classificados.

Nesta classificação, os três primeiros da tabela acima foram agrupados sob o rótulo de aprendizado associativo. Neste aprendizado, dois padrões a serem associados são apresentados, e a rede deve aprender a mapeá-los (no caso do auto-associador, o objetivo é mapeá-lo em si mesmo). No caso do detetor de regularidades, não existe, a priori, um conjunto de classes determinado para separar os padrões de entrada. Não é fornecido um padrão de saída, ele deve ser descoberto.

A classificação de Rumelhart [RUME86A] se assemelha àquela que classifica as redes quanto ao tipo de controle realizado durante o aprendizado (supervisionado ou não-supervisionado), apresentado na introdução desta tese. A classificação de Rumelhart e Zipser [RUME85] é mais específica do ponto de vista de como são processados (armazenados e recuperados) os padrões durante o treinamento das redes.

Uma descrição geral das diversas técnicas de aprendizado utilizadas nas RNAs, inspirado em Simpson [SIMP90], é apresentado na tabela 4.

Tabela 4. Principais técnicas de aprendizado em RNAs

APRENDIZADO	DESCRIÇÃO GERAL
Correção de Erros	Aprendizado supervisionado que ajusta os pesos das conexões entre os nós na proporção da diferença entre os valores desejados e computados de cada nó da camada de saída.
Reforço	Aprendizado supervisionado onde os pesos são recompensados quando o sistema executa ações apropriadas, e punidos caso ele não as execute.
Estocástico	Aprendizado supervisionado que usa processos aleatórios, probabilidade e relações de energia para ajustar os pesos dos arcos.
Sistemas "Hardwired"	As conexões e os respectivos pesos são pré-determinados, semelhante a um autômato de estados finito.
Regra de Hebb	Aprendizado onde o ajuste dos pesos das conexões é realizado em função da relação de valores dos dois nós que ela conecta. Pode ser aplicado tanto ao aprendizado supervisionado quanto ao aprendizado não-supervisionado.
Competitivo e Cooperativo	Aprendizado não supervisionado onde os processos competitivo e cooperativo são descritos em termos de redes com conexões recorrentes auto-excitáveis. Estes arcos podem ser inibidores dos nós vizinhos (competitivo), e/ou excitadores dos vizinhos (cooperativo).
Sistemas Conectados Aleatoriamente (SCA)	Aprendizado não supervisionado utilizado para suportar a teoria de que a mente é uma rede conectada aleatoriamente quando vista do nível macroscópico.

No aprendizado por Correção de Erros existia um problema devido a sua incapacidade de estender o aprendizado para uma rede com mais de duas camadas. Especificamente, o valor do erro acumulado em cada unidade intermediária, que deveria ser creditado para as unidades de saída, não era definido. Com o advento de modelos como o "backpropagation" [RUME86A] este problema foi solucionado. Este algoritmo consiste numa generalização da regra delta (ver seção 11.1, regra de aprendizado - modificando a conectividade através da experiência). Derivando-se a regra delta obtém-se [RUME86B]:

$\Delta w_{i,j} = \eta \cdot \delta_j \cdot o_i$, onde

$$\delta_j = \begin{cases} (t_j - o_j) \cdot f'_j(\text{net}_j) & \text{se a célula } j \text{ for de saída} \\ f'_j(\text{net}_j) \cdot \sum_k \delta_k \cdot w_{j,k} & \text{se a célula } j \text{ for intermediária,} \end{cases}$$

onde f é a função de ativação f' é a sua derivada. As unidades de saída usam a diferença entre os valores obtido e desejado, como na regra delta. As unidades intermediárias precisam utilizar os valores de δ previamente calculados. Para estas células, este cálculo é realizado de maneira recorrente. Desta forma, na primeira fase (a fase "forward"), a entrada é ativada e espera-se a ativação da saída, calculando-se δ_j para os nós de saída. Na segunda fase (a fase "backward"), a partir deste cálculo de δ_j , é feita a propagação dos sinais de erro para as camadas anteriores, efetuando-se as mudanças nos pesos. Apesar da pouca plausibilidade biológica, os resultados obtidos pelo "backpropagation" aumentaram consideravelmente o interesse em RNAs.

O aprendizado por Reforço pode ser modelado pela equação:

$$\Delta w_{i,j} = \alpha \cdot [r - \Theta_j] \cdot e_{i,j},$$

onde r é um valor escalar de sucesso/falha informado pelo ambiente, Θ_j é o valor do limiar de reforço para o j -ésimo neurônio de saída, $e_{i,j}$ é a elegibilidade canônica do peso da i -ésima para a j -ésima unidade de processamento, e α é uma constante ($0 < \alpha < 1$) que regula o rateio de aprendizado. A elegibilidade canônica é função da distribuição de probabilidade selecionada que é usada para determinar se o valor de saída computado é igual ao valor desejado. Ela é definida como $e_{i,j} = \partial \ln g_i / \partial w_{i,j}$, onde g_i é a probabilidade da saída desejada se igualar à saída computada.

O aprendizado Competitivo/Cooperativo pode ser descrito como um sistema de equações diferenciais do tipo $dx_i/dt = F_i(x_1, x_2, \dots, x_n) = F_i(x)$, $i = 1, 2, \dots, n$. Ele é competitivo se $\partial F_i/\partial x_j \leq 0 \forall j \neq i$, e cooperativo se $\partial F_i/\partial x_j > 0 \forall j \neq i$. A idéia fundamental deste aprendizado é possibilitar que os neurônios da rede disputem, de alguma maneira, para obtenção do direito de responder a determinados estímulos. Este tipo de aprendizado permite que as unidades se especializem, e passem a atuar como detetores de características ou classificadores de padrões sem auxílio externo [RUME85].

O aprendizado Estocástico permite criar RNAs que escapem do problema de *mínimos locais* (que é descrito na próxima seção), utilizando um procedimento de diminuição de energia chamado de "simulated annealing" [KIRK83].

Os Sistemas Conectados Aleatoriamente (SCA) se fundamentam em um princípio de organização dos caminhos de sensoriamento na mente. Este princípio postula que o posicionamento dos neurônios é ordenado e, geralmente, reflete alguma característica física do estímulo que está sendo percebido (por exemplo áreas visuais, áreas auditivas) [LIPP87]. Desta forma, o cérebro humano pode ser considerado como uma coleção estruturada de neurônios [KOH088].

II.3 - DINÂMICA: CONVERGÊNCIA E ESTABILIDADE

Dois importantes conceitos regem a dinâmica das redes neurais: a convergência e a estabilidade global. Biologicamente, *convergência* é a formação de similaridades sucessivas entre organismos ou associações antes distintas. Dentro do contexto de modelos conexionistas, a convergência está relacionada com a minimização do erro (eventual) entre as saídas computadas e desejadas dos neurônios. Em geral, o conceito de convergência está associado ao aprendizado supervisionado.

O conceito de *estabilidade* está fortemente relacionado com a manutenção, ou retorno, a um estado de equilíbrio, sólido e estável. No contexto das RNAs, a

estabilidade global é a estabilização de todas as ativações das unidades que compõem a rede, independente de qualquer entrada. Geralmente, o conceito de estabilidade está relacionado com a chamada *recursiva* existente nas redes cíclicas. Durante a execução de uma determinada tarefa, estas redes executam, eventualmente, vários ciclos de processamento até atingir um estado estável.

Quando se estuda a questão da convergência e da estabilidade em sistemas dinâmicos é comum se utilizar o conceito de *atrator*. Associando a cada RNA uma *superfície* de energia, de modo que, quando excitada com um padrão de atividade inicial, representado por um ponto sobre a referida superfície, a rede evoluirá dinamicamente, reduzindo a cada instante o valor da sua energia computacional, tendendo para o ponto de menor energia mais próximo que possa atingir. O atrator seria justamente este ponto de mínimo da superfície. O ponto de mínimo mais baixo de toda a superfície é chamado de mínimo global. Os demais pontos de mínimo existentes na rede são chamados de mínimos locais. Uma representação de um mínimo local é apresentado na figura 5.

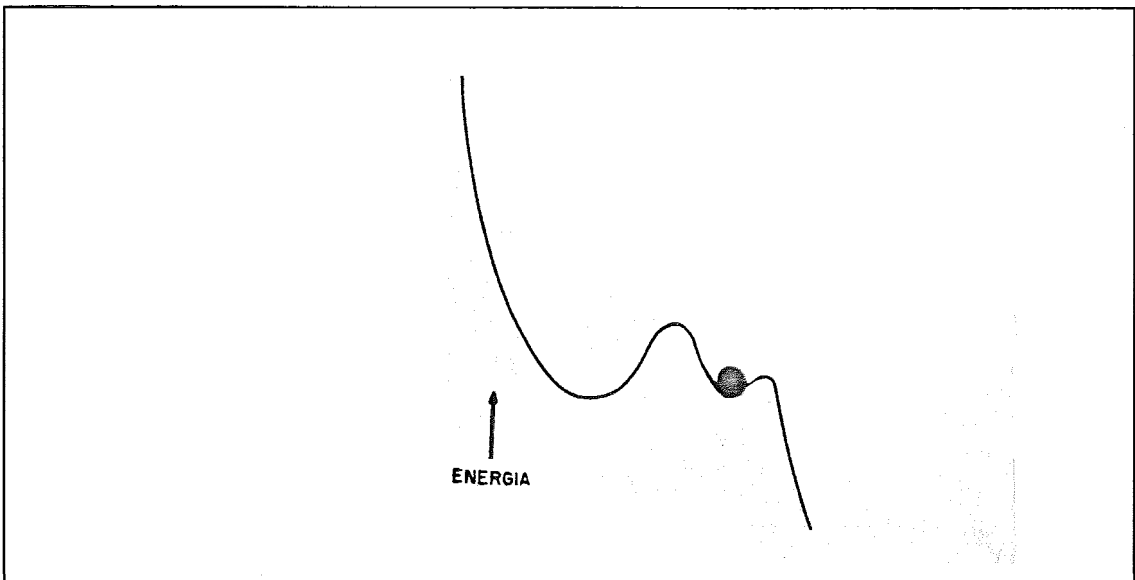


Figura 5. Representação de um mínimo local

Convergência

A convergência é definida como uma seqüência de números x_n que tendem a um limite x , se dado um $\varepsilon > 0$ pode-se obter um inteiro n_0 tal que $|x_n - x| < \varepsilon$

para todo $n > n_0$. Existem dois métodos de convergência que se fundamentam neste princípio: *convergência com probabilidade 1*, e *convergência pela diferença dos quadrados*. A convergência com probabilidade 1 é definida como sendo

$$\lim_{n \rightarrow \infty} P\{x_n = x\} = 1,$$

onde $P\{x\}$ representa a probabilidade de x . A convergência pela diferença dos quadrados é definida como sendo

$$\lim_{n \rightarrow \infty} E\{|x_n - x|^2\} = 0,$$

onde $E\{x\}$ representa o valor esperado de x . A convergência é uma forma de analisar se um procedimento de aprendizado qualquer está capturando corretamente o mapeamento entre os dados apresentados. Tal afirmativa é verdadeira se o mapeamento convergir para um valor, ou para um conjunto, fixo. Um *grau de erro* é utilizado para medir o desempenho do procedimento de aprendizado durante a realização desta tarefa.

Estabilidade Global

As RNAs globalmente estáveis são definidas como sistemas dinâmicos que mapeiam rapidamente todas as entradas em atratores. Estes atratores (pontos de convergência, equilíbrio, limites, etc) são locais onde a informação pode ser deliberadamente armazenada. Isto não garante que os sistemas fazem o mapeamento da maneira desejada.

Existem três teoremas principais que são usados para provar a estabilidade em RNAs. Cada um destes teoremas utiliza o *método direto de Lyapunov* [SIMP90]. Este método estuda a estabilidade de encontrar certas funções das variáveis de sistemas dinâmicos cujas derivadas, em relação ao tempo, possuem propriedades específicas. Os três teoremas são os seguintes:

1. Teorema de Cohen-Grossberg - usado para mostrar a estabilidade em auto-associadores não-adaptativos. Este tipo de RNA opera em tempo contínuo, utiliza chamada recursiva, e modifica os pesos das conexões que são desprezíveis, em relação aos valores de ativação dos neurônios, podendo ser consideradas constantes [COHE83];
2. Teorema de Cohen-Grossberg-Kosko - utilizado para descrever a estabilidade em auto-associadores adaptativos. É uma extensão do primeiro teorema. Neste caso, os auto-associadores adaptativos são redes que não possuem a restrição de ser um sistema que não pode aprender quando estiver recebendo informação [KOSKSS];
3. Teorema ABAM - usado para descrever as ativações de redes neurais de duas camadas globalmente estáveis, que podem aprender e se adaptar ao mesmo tempo (heteroassociadores). Este teorema representa um caso específico do teorema de Cohen-Grossberg-Kosko [KOSK88].

II.4 - APLICAÇÕES E PRINCIPAIS MODELOS CONEXIONISTAS

As RNAs tem sido utilizadas em diversas aplicações. A relação que é apresentada a seguir, baseada em Simpson [SIMP90], representa as áreas em que as redes tem sido usadas, ou que possuem aplicação em potencial. A relação é a seguinte:

- Processamento de Sinais;
- Reconhecimento de Padrões;
- Compressão de Dados;
- Processamento de Imagens;
- Processamento de Fala;
- Controle;
- Sistemas Ecológicos;

- Projeto de Circuitos Integrados;
- Processamento de Conhecimento;
- Tarefas de Classificação (ex. diagnóstico);
- Estudo de Conexões Neurobiológicas;
- Memória Associativa;
- Recuperação de Informações em Banco de Dados;
- Processamento de Texto e Sentenças;
- Busca em Grafos;
- Coloração de Grafos;
- Problema do Caixeiro Viajante;
- Alocação de Recursos; e
- Escalonamento de Tarefas.

Existem muitos modelos conexionistas, e uma quantidade razoável de publicações que se dedicam a classificá-los ([LIPP87], [HINT89], [SIMP90]). Dentre estes modelos, foram selecionados cinco que são apresentados a seguir:

1. Perceptron
2. Perceptron de Multi-Camadas
3. Classificador de Carpenter/Grossberg
4. Rede de Kohonen
5. Rede de Hopfield

A tabela 5 apresenta os referidos modelos. A tabela 6 corresponde a uma descrição das principais características dos mesmos.

Tabela 5. Apresentação dos principais modelos conexionistas

MODELO	PESQUISADORES	INÍCIO DE PUBLICAÇÃO	CLASSE DE TAREFAS
Perceptron	F. Rosenblatt	1957	Reconhecimento de padrões (ex: caracteres impressos)
Perceptron de Multi-Camadas (Backpropagation)	P. Werbos, D. Parker e D. Rumelhart	1974	Reconhecimento de padrões (ex: controle adaptativo de braços de robôs) e processamento de fala.
Classificador de Carpenter/ Grossberg (Sistema ART)	G. Carpenter e S. Grossberg	1978	Reconhecimento de padrões (ex: reconhecimento de sinais de radar ou sonar) e processamento de imagens.
Rede de Kohonen	T. Kohonen	1980	Reconhecimento de padrões (ex: reconhecimento de fala) e aprendizado da distribuição de probabilidades dos dados (ex: auto-organização de mapas de características)
Rede de Hopfield	J. Hopfield	1982	Reconhecimento de padrões (ex: reconhecimento de dados ou imagens completas a partir de fragmentos) e memória associativa.

Observando a tabela 5.(B), pode-se notar que existe uma preferência pelo uso da função sigmóide como função de ativação. Esta função permite modelar a saída de cada neurônio, em relação a sua ativação de entrada, de uma forma gradual e não-linear.

O primeiro modelo conexionista desenvolvido foi o Perceptron de duas camadas [ROSE59] que pode ser usado com valores contínuos. Esta rede gerou muito interesse pela habilidade de aprender a reconhecer padrões linearmente separáveis. Um problema com este procedimento de convergência do perceptron é que os limites de decisão podem oscilar continuamente quando as

entradas forem não separáveis, e quando ocorrer uma sobreposição de distribuições [SIMP90].

Tabela 6. Características dos principais modelos conexionistas

MODELO	PROPRIEDADES DAS CÉLULAS	PROPRIEDADES DAS REDES	APRENDIZADO
Perceptron	As células possuem entradas binárias e saídas que assumem os valores $+1$ ou -1 . A função de ativação é a função salto.	Rede acíclica de duas camadas.	Utiliza a técnica de Reforço
Backpropagation	As células são do tipo perceptron, e possuem valores contínuos. A função de ativação é a função sigmóide.	Rede acíclica de três camadas no mínimo.	Utiliza a técnica de Correção de Erros com o uso da regra delta generalizada.
Sistema ART	As células possuem entradas binárias, podendo assumir valores contínuos. A função de ativação é a função sigmóide.	Rede cíclica de três camadas.	Utiliza a técnica de aprendizado Competitivo e Cooperativo que foi introduzida pelo próprio Grossberg.
Rede de Kohonen	As células possuem entradas contínuas. A função de ativação é a função sigmóide.	Rede cíclica de duas camadas.	Utiliza a técnica de SCA. O próprio Kohonen foi um dos seus introdutores.
Rede de I-Hopfield	As células possuem entradas binárias e saídas que assumem os valores $+1$ ou -1 . A função de ativação é a função sigmóide.	Rede cíclica de uma camada.	Os padrões são armazenados no início, se assemelhando a técnica utilizada pelos sistemas "hardwired".

Ele não é apropriado para classes que não podem ser separadas por um hiperplano. Por exemplo, as distribuições para as duas classes do problema do OU-exclusivo (XOR) não são linearmente separáveis, logo não podem ser separadas por uma simples linha. O OU-exclusivo é o problema difícil clássico

para as RNAs. Ele representa o problema não linearmente separável mais simples que existe. Este problema foi utilizado por Minsky e Papert [MINS69] para ilustrar a fraqueza do perceptron, que contribuiu para o ostracismo desta área por mais de uma década.

No começo da década de setenta foram desenvolvidos os Perceptrons de Multi-Camadas. Estes são redes acíclicas com uma ou mais camadas de neurônios intermediários entre as camadas de entrada e de saída. Um algoritmo capaz de treinar os perceptrons de multi-camadas é o "backpropagation" (ver seção 11.2) [RUME86C].

O algoritmo "backpropagation" foi testado em uma série de problemas clássicos, tais como o OU-exclusivo, e em problemas relacionados com reconhecimento de padrões visuais [RUME86C]. Geralmente, o "backpropagation" usa, para aprender a classificar o OU-exclusivo, cerca de 150 iterações sobre a tabela verdade do mesmo. Na maioria dos casos, ele encontrou boas soluções para os problemas propostos, apesar do algoritmo, às vezes, fornecer uma configuração de pesos correspondente a um mínimo local da função de erro. Isto é devido ao fato do backpropagation utilizar o método do gradiente e, a princípio, a superfície do erro possui uma forma qualquer.

Uma aplicação bastante conhecida do "backpropagation" é o sistema NETtalk [SEJN87]. Com o treinamento, este sistema é capaz de aprender a pronunciar palavras em inglês de uma maneira correta. Ele também possui a capacidade de generalizar para palavras novas, que ainda não haviam sido apresentadas ao mesmo.

No final da década de setenta, Carpenter e Grossberg [GROS86] projetaram uma rede capaz de formar aglomerados de informações ("clusters"), e de ser treinada sem supervisão, chamada de sistema ART. Nesta rede, as sinapses são bi-direcionais. As ligações "bottom-up" tentam categorizar os estímulos de entrada, e os arcos "top-down" tentam aprender estas categorias. Os nós da camada de saída funcionam como unidades "winner-take-all", ou seja, apenas a unidade que recebe a maior excitação convergirá para a ativação 1, as demais evoluirão para a ativação 0.

Em linhas gerais, o sistema ART funciona da seguinte maneira: O algoritmo principal seleciona a primeira entrada como um exemplo para o primeiro aglomerado. A entrada seguinte é comparada com este primeiro exemplo. Ela é agrupada com o mesmo se a distância para o primeiro for menor que um certo limiar, chamado de limiar de vigilância. Caso contrário, este exemplo formará um novo aglomerado. Este processo se repete para todas as entradas existentes. O número de aglomerados cresce em função do limiar e da métrica da distância usada para comparar os exemplos de entrada dos aglomerados.

No início da década de oitenta, Kohonen [KOH84] propôs uma rede onde se corroborou os estudos teóricos sobre a organização dos caminhos de sensoriamento na mente. Segundo esta teoria, conforme apresentado na seção 11.2, o cérebro humano foi considerado como uma coleção estruturada de neurônios. Esta *ordem espacial das unidades de processamento* [KOH82] permitiu elaborar uma rede neural dotada de mecanismos que permitem formar representações estruturadas dos estímulos de entrada. Após o aprendizado, as unidades respondem a diferentes estímulos de maneira ordenada, formando um sistema de coordenadas de características sobre a rede.

O algoritmo de auto-organização, proposto por Kohonen [KOH88], é um quantificador de vetores seqüenciais, e funciona como um algoritmo de aglomeração. Se os estímulos formam aglomerados no espaço de entrada, o algoritmo faz com que os padrões de um aglomerado sejam mapeados para uma região localizada comum no mapa. Ele faz um arranjo destas regiões ao longo do mapa, de maneira a capturar o máximo possível da topologia geral dos aglomerados de entrada. Nesta rede, as células de entrada estão conectadas a todas as células de saída. As unidades de saída estão interconectadas em um arranjo bi-dimensional.

Kohonen demonstrou como utilizar o algoritmo de auto-organização para problemas de reconhecimento de fala [KOH88]. Neste experimento, a rede foi apresentada a conteúdos espectrais, em vez de fonemas, e observou-se a formação de um mapa fonotópico onde as distâncias entre os pontos no mapa

(unidades) são proporcionais às diferenças vetoriais entre os conteúdos espectrais originais.

Também na década de oitenta, o trabalho de Hopfield [HOPF82] contribuiu substancialmente para o ressurgimento das pesquisas em RNAs. A rede de Hopfield, bem como o classificador de Carpenter/Grossberg são mais apropriadas quando representações binárias permitem modelar a situação desejada. Por exemplo, imagens em preto e branco, onde os elementos de entrada podem ser representados pelos valores de cada ponto da imagem (0 = branco, 1 = preto) [LIPP87].

Esta rede pode ser usada como uma memória associativa ou para resolver problemas de otimização. Uma versão da rede original [HOPF82] pode ser usada como uma memória de acesso por conteúdo. Uma extensão da rede de Hopfield, com aplicação ao problema do caixeiro viajante e à representação do conhecimento, pode ser encontrada em Carvalho [CARV89].

Esta rede possui duas limitações quando usada como memória de acesso por conteúdo. Primeiro, apesar dos padrões armazenados, a rede pode convergir para um novo padrão diferente dos padrões exemplo existentes. Isto pode produzir a situação de não bntimento da rede, ou seja, a rede não casa com um padrão ja existente.

Uma segunda limitação é que o padrão exemplo será considerado instável se ele compartilhar muitos bits com outro padrão exemplo, o que pode ocasionar uma convergência da rede para este outro exemplo. Este problema pode ser eliminado, consequentemente melhorando seu desempenho, utilizando-se procedimentos de ortogonalização [LIPP87].

CAPÍTULO III

SISTEMAS ESPECIALISTAS CONEXIONISTAS

Uma das principais características dos modelos conexionistas é a sua capacidade de aprendizado. Esta característica tem atraído alguns pesquisadores da área de SEs. Tais pesquisadores vêem nesta capacidade dos modelos conexionistas uma forma de superar o *gargalo* de aquisição de conhecimento apresentado por Feigenbaum [FEIG82].

Em relação ao desempenho apresentado pelos SEs tradicionais, existe um aspecto importante a ser analisado. Estes sistemas conseguem demonstrar uma capacidade de raciocínio semelhante ao apresentado por especialistas humanos dentro de domínios de conhecimento restritos. Esta restrição quanto ao tamanho do domínio é o principal motivo de sucesso dos SEs e também fonte de limitações. Neste sentido, estes sistemas são considerados *frágeis* por responderem apropriadamente *apenas* em domínios restritos. A intervenção humana torna-se necessária quando se deseja efetuar mudanças, mesmo que pouco significativas, no conhecimento representado no domínio [HOLLSGA].

Como resultado destas motivações, alguns trabalhos integrando SEs e modelos conexionistas começam a surgir ([GALL88], [SAMA88], [SAIT88], [ROMA89], [MACH89]). Gallant [GALL88] criou o termo *Sistema Especialista Conexionista* para classificar tal categoria de SEs. Neste capítulo são apresentados alguns trabalhos encontrados na literatura. Uma atenção especial é dedicada ao Modelo Neural Combinatório [MACH89], pois o mesmo serviu de base para esta pesquisa.

III.1 - APLICAÇÕES NA ÁREA DE SISTEMAS ESPECIALISTAS

Alguns trabalhos que utilizam o paradigma conexionista aplicado a SEs começam a surgir. Relacionamos cinco destes a saber:

1. O SE Conexionista (SEC) proposto por Gallant [GALL88];
2. O RUBICON proposto por Samad [SAMA88];
3. O FUZZNET proposto por Romaniuk [ROMA89];
4. O SE para diagnóstico baseado no PDP (SDPDP) proposto por Saito [SAIT88]; e
5. O Modelo Neural Combinatório (MNC) proposto por Machado [MACH89].

A tabela 7 apresenta os referidos SEs conexionistas. A tabela 8 descreve as principais características das redes dos mesmos.

Tabela 7. Apresentação dos SEs Conexionistas			
Modelo	Pesquisadores	Ano de Publicação	Exemplos de Aplicações
Sistema Especialista Conexionista (SEC)	S. Gallant	1988	Diagnóstico de Doença do Sarcófago e Seleção de Tratamento
RUBICON	T. Samad	1988	Identificação de Títulos de Filmes
FUZZNET	S. Romaniuk e L. Hall	1989	Diagnóstico de Febre e Identificação de Pedras Preciosas
Sistema de Diagnóstico baseado no PDP (SDPDP)	K. Saito e R. Nakano	1988	Diagnóstico de Dor de Cabeça por Contração Muscular
Modelo Neural Combinatório (MNC)	R. Machado e A. Rocha	1989	Diagnóstico de Doenças Renais

Tabela 8. Características das redes dos SEs Conexionistas.

MODELO	PROPRIEDADES DAS CÉLULAS	PROPRIEDADES DAS REDES	APRENDIZADO
SEC	O grau de ativação das células pode assumir os valores -1, 0 ou +1 para falso, desconhecido e verdadeiro respectivamente. A função de ativação é a função salto.	Rede acíclica de três camadas. A camada de entrada representa os sintomas, a intermediária as doenças e a de saída os tratamentos.	Rede treinada com supervisão, de acordo com a técnica de Reforço.
RUBICON	As células possuem entradas binárias e saídas que assumem os valores 0 ou 1. A função de ativação é a função salto.	Rede acíclica de seis camadas conectadas na seguinte ordem: entrada, antecedente, coletora, regra, conseqüente e saída.	Rede treinada com supervisão com uma técnica semelhante a utilizada pelos sistemas "hardwired".
FUZZNET	As células podem assumir um valor de ativação no intervalo [0, 1], que corresponde a um grau de possibilidade nebuloso. A função de ativação é a função sigmóide.	Rede acíclica de quatro camadas no mínimo (entrada, intermediária(s), extra e saída). A camada extra é formada por duas células (positiva e negativa) estando conectadas a todas as células intermediárias e de saída.	Rede treinada com supervisão com uma técnica semelhante à utilizada pelos sistemas "hardwired".
SDPDP	As células são do tipo perceptron, podendo assumir um valor de ativação no intervalo [0, 1]. A função de ativação é a função sigmóide.	Rede acíclica de três camadas. A camada de entrada representa os sintomas, a intermediária combinações de sintomas, e a de saída as doenças.	Rede treinada com supervisão, de acordo com a técnica de Correção de Erros.
MNC	O grau de ativação das células pode assumir um valor no intervalo [0, 1], e representa um grau de possibilidade nebuloso. As células podem ser do tipo E-nebuloso ou OU-nebuloso.	Rede acíclica de 3 ou mais camadas. As camadas intermediárias são formadas por células E-nebuloso e representam combinações de evidências. A camada de saída é formada por células OU-nebuloso e representam as hipóteses.	Rede treinada com supervisão. Punições e recompensas para as sinapses são calculadas para cada exemplo de acordo com a regra de Hebb. Os pesos sinápticos são calculados normalizando-se os valores acumulados de recompensas e punições recebidas pelas sinapses.

Observando a tabela 6.(A) pode-se notar que os exemplos de aplicações destes sistemas estão inseridos em uma classe de tarefas chamada de *classificação*. A classificação é uma tarefa de processamento de informação, onde entidades específicas são mapeadas em categorias mais gerais [CHAN88]. Esta tarefa é subjacente a muitos domínios de problema, tais como diagnose, predição, monitoração, interpretação, identificação, depuração e seleção, dentre outros.

O SEC [GALL88] é formado por uma base de conhecimento (composto por uma matriz de pesos, nomes de variáveis e questões), um motor de inferência, chamado de MACIE ("Matrix Controlled Inference Engine"), e por último uma interface com o usuário. O algoritmo de aprendizado utilizado no sistema é o *algoritmo de Bolso*.

O algoritmo de Bolso é uma modificação do algoritmo do Perceptron. Ele computa os vetores de peso, chamados de perceptron P , os quais substituem ocasionalmente os vetores de peso w já existentes no *bolso*. Esta substituição ocorre quando este algoritmo, ao classificar um exemplo de treinamento E qualquer, consegue uma taxa de acerto superior à obtida pelos vetores de peso w já existentes.

Uma falha no algoritmo de Bolso é que não existe um limite (número) de execuções necessárias para garantir que os pesos *embolsados* são os melhores possíveis. Entretanto, se não existir muitos exemplos de treinamento ($< 10^5$), é possível checar periodicamente os pesos embolsados contra o conjunto de todos os exemplos de treinamento com o objetivo de avaliar seu desempenho.

O motor de inferência utiliza o paradigma conexionista para as seguintes tarefas:

- Inferência baseada em informação parcial, usando o encadeamento progressivo ("Forward Chaining");
- Identificação das variáveis de entrada desconhecidas que são fundamentais para a realização de inferências adicionais; e
- Produção de justificativas para as inferências realizadas utilizando o encadeamento regressivo ("Backward Chaining").

O RUBICON é uma arquitetura conexionista usada para implementação de SEs baseados em regras [SAMA88]. Ela utiliza tanto representações distribuídas como localizadas. Algumas características importantes deste sistema são:

- Número arbitrário de antecedentes e conseqüentes em qualquer regra;

- Inclusão e remoção de elementos na memória de trabalho; e
- Manipulação de expressões negativas nos antecedentes e conseqüentes das regras.

A exclusão de elementos se processa através de uma unidade adicional chamada DEL. Quando ativa, ela indica que aquela regra é para ser excluída da base de regras. Sinapses inibitórias são criadas, inibindo as conexões que chegam ao nó que representa o elemento a ser excluído.

A negação de elementos é feita com o uso de uma unidade adicional chamada NEG. Quando ativa, ela inverte o valor dos pesos das ligações, tornando inibitórias as excitatórias e vice-versa. Esta negação explícita permite ao RUBICON distinguir entre a falta de uma evidência (não saber), e o conhecimento de sua ausência (saber que não).

O FUZZNET [ROMA89] utiliza a teoria dos conjuntos difusos para fazer raciocínio aproximado em SEs baseados em regras. As principais características deste SE conexionista são:

- Número variável de antecedentes e conseqüentes das regras;
- Exclusão de informações da base de conhecimento; e
- Negação de expressões nos antecedentes e conseqüentes das regras.

Ele utiliza conceitos de lógica nebulosa para definir as operações de máximo (fOR), mínimo (fAND) e complemento (fNOT). Para tornar apta a distinção entre células que são modeladas pela função máximo (fOR), das que são modeladas pela função mínimo (fAND), utiliza-se a sinalização do viés. Um viés com valor zero indica que este neurônio é uma célula inversora (fNOT).

A remoção de fatos antigos e a substituição por novos na base de conhecimento se processa através da inclusão de novos arcos e novos nós. Este processo se caracteriza pelo aprendizado do complemento do fato que se deseja esquecer, ao invés de se remover conexões, ou mesmo células, da rede. Uma

vez disparado o fato desejado, seu complemento também é ativado, resultando numa conclusão indeterminada.

O SDPDP [SAIT88] é um sistema para diagnóstico médico baseado no processamento paralelo distribuído. Ele utiliza o algoritmo "backpropagation" ([HINT89], [RUME86C]) no seu processo de aprendizado.

Neste sistema foram desenvolvidos dois métodos para extrair conhecimento simbólico a partir da rede. O primeiro, chamado de *extração de fatores de relação*, utiliza o relacionamento existente entre os sintomas e as doenças. Por ser linear, este método não consegue expressar plenamente as características inerentes da RNA.

O segundo método, chamado de *extração de regras*, trata a rede como se fosse uma caixa preta, referenciando-se apenas as camadas de entrada e saída. As regras podem ser positivas ou negativas, e possuem a seguinte estrutura:

positivas - se (doença) então (sintomas positivos);

negativas - se (doença) então (sintomas positivos, sintomas negativos).

Este método é usado no processo de confirmação de sintomas. Neste processo, o paciente seleciona os sintomas e o SDPDP faz o diagnóstico gerando hipóteses. É feito o batimento das hipóteses geradas contra as doenças existentes nas regras produzidas pelo método acima. Desta maneira, confirma-se ou não os sintomas.

III.2 - O MODELO NEURAL COMBINATÓRIO

O Modelo Neural Combinatório (MNC) é um modelo conexionista de ordem superior voltado para tarefas de classificação. Trata-se de um modelo de rede capaz de realizar *aprendizado heurístico*. Este tipo de aprendizado é fundamental para a aquisição do conhecimento humano e é baseado no reconhecimento das regularidades do meio ambiente através de observações repetidas do mundo externo. Este modelo inspira-se nas ciências neurais, na lógica nebulosa, e em pesquisas recentes em análise do conhecimento especialista [MACH89].

O MNC é formado por nós que são conectados em uma rede de acordo com uma topologia definida. Cada célula representa um conceito simbólico do domínio de problema, tais como, evidências, hipóteses, estados do problema, etc. Cada célula calcula seu estado de ativação, pertencente ao intervalo $[0,1]$, baseado na sua informação de entrada. O estado de ativação representa também o valor de saída da célula. Pode ser interpretado como o grau de crença ou aceitação que o sistema coloca no conceito representado pela célula específica.

As unidades de processamento são conectadas através de ligações que podem ser classificadas como *excitatórias* ou *inibitórias* e são caracterizadas por um peso, variando entre 0 (não conectada) a 1 (sinapse plenamente conectada). O arco recebe como sinal de entrada um valor numérico entre 0 e 1 proveniente do usuário ou da saída de uma célula. As sinapses excitatórias atuam diretamente sobre o sinal de entrada. As ligações inibitórias atuam sobre a negação difusa do sinal de entrada, ou seja, dado um sinal de entrada X transformam-no em $1-X$. Em ambos tipos de sinapse o peso é usado como um fator de atenuação.

A rede é uma rede acíclica com três ou mais camadas. A Camada de Entrada recebe dados simbólicos do ambiente ou do usuário. Esta informação de entrada é apresentada ao sistema com um grau de crença ou grau de semelhança, variando entre 0 (crença nula) ou 1 (crença plena). As células da

camada de saída (OU-nebuloso) implementam o mecanismo de competição entre as diferentes vias provenientes das camadas inferiores. As células pertencentes às camadas combinatórias (E-nebuloso) representam abstrações intermediárias, capazes de reduzir a complexidade computacional da tarefa de classificação ([LEAO88], [THEO87]). Uma versão completa do MNC para 3 evidências de entrada e 2 hipóteses está ilustrada na figura 6.

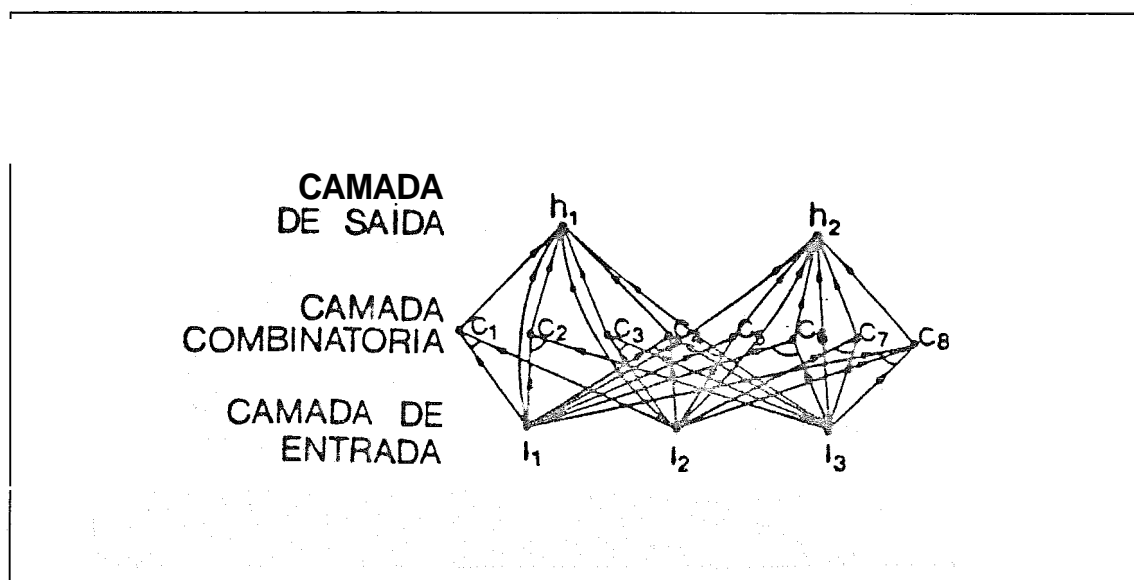


Figura 6. Versão completa do MNC para 3 evidências e 2 hipóteses

A versão completa do MNC é claramente infactível devido à explosão combinatória, aqui expressa na versão espacial, que exige $2^n - n$ células na camada intermediária [MACH89]. A explosão combinatória dos termos de ordem superior foi a razão apresentada por Minsky e Papert [MINS69] para concluir que os perceptrons de ordem superior eram impraticáveis.

Para evitar a desnecessária repetição de células na rede, foram incluídas ligações diretas entre as células de entrada e as células da camada de saída. Estas ligações diretas são observadas nos grafos de conhecimento eliciados de especialistas humanos ([LEAO88], [MACH90B]).

Porém o mais importante é limitar o tamanho dos aglomerados para evitar a explosão combinatória espacial. Como solução prática, define-se o *Modelo de Ordem m* , que inclui na camada intermediária somente as combinações com

comprimento igual ou menor que o limiar m . Machado [MACH89] sugere fazer m igual ou menor que o número mágico (sete mais ou menos dois) de Miller [MILL56].

A rede tem dois modos principais de operação: o *modo de consulta* e o *modo de aprendizado*. No modo de consulta, todas as células operam em paralelo. As evidências disponíveis são apresentadas aos neurônios da camada de entrada com seus correspondentes graus de crença. Esta informação é propagada progressivamente na rede até os nós-hipótese, de acordo com as regras de inferência dos neurônios E-nebuloso e OU-nebuloso.

As hipóteses, cujos graus de possibilidade ultrapassam um limiar de aceitação pré-definido (L_{aceit}), são apresentadas ao usuário como a solução do problema. O tempo total de computação é diretamente proporcional ao número de camadas, já que estamos lidando com redes acíclicas. A complexidade computacional da consulta é $O(n + s)$, onde n é o número de neurônios e s é o número de sinapses.

No Modo de Aprendizado, a rede constrói indutivamente representações internas complexas de seu meio ambiente usando um mecanismo de punição e recompensa inspirado no condicionamento operante. O sistema trabalha de acordo com o esquema de aprendizado supervisionado, usando uma base de dados de treinamento contendo exemplos (casos com sua classificação correta conhecida).

Baseado no conjunto de exemplos de treinamento, o algoritmo de aprendizado é capaz de ajustar os pesos das conexões de tal forma que as unidades internas da rede venham a representar aspectos importantes do domínio de problema. A seguir é apresentado um algoritmo de aprendizado indutivo proposto para o MNC.

III.2.1 - O Algoritmo de Aprendizado

O algoritmo trabalha de acordo com um mecanismo de aprendizado supervisionado e é capaz de operar, em muitos casos, usando apenas uma *iteração*. A iteração é a aplicação do algoritmo de aprendizado sobre todos os casos da base de dados de treinamento. O processo de aprendizado é realizado em dois passos executando-se os seguintes algoritmos:

- *Algoritmo de Punição e Recompensa;* e
- *Algoritmo de Poda e Normalização.*

O algoritmo de Punição e Recompensa usa uma versão simples do mecanismo de retropropagação e é capaz de trabalhar com dados de treinamento booleanos ou difusos. Para cada sinapse da rede são definidos acumuladores que acumularão as recompensas e as punições recebidas respectivamente.

Este algoritmo atua sobre a rede em função do *Fluxo Evidencial* e do *Grau de Importância* de um exemplo, cujas definições são apresentadas a seguir:

- Fluxo Evidencial numa sinapse é o produto (Ativação do neurônio de origem . Peso sináptico) se a sinapse for excitatória, e $((1 - \text{Ativação do neurônio de origem}) \cdot \text{Peso sináptico})$ se a sinapse for inibitória.
- Grau de Importância de um exemplo é uma atribuição que é feita a cada exemplo de treinamento existente. O impacto de um exemplo no aprendizado será diretamente proporcional a seu grau de importância.

Existem duas versões para o algoritmo de Punição e Recompensa correspondendo aos dois modos de aprendizado disponíveis:

- *A Versão de Arranque* no caso de *aprendizado a partir do nada;* e
- *A Versão de Refinamento* no caso de *aprendizado por refinamento de um conhecimento disponível.*

Na Versão de Arranque (ver algoritmo no apêndice 1), parte-se de uma rede sem conhecimento do domínio do problema: uma rede não treinada de ordem limitada com a topologia descrita anteriormente. Depois de realizar uma varredura (iteração) sobre a base de exemplos de treinamento, o algoritmo de arranque produz uma rede treinada não operante com valores de acumuladores no intervalo $(-T, +T)$ anexados a seus arcos. (T é o número de exemplos da base de dados de treinamento).

Em alguns casos faz-se necessário aplicar a versão de Refinamento do algoritmo de Punição e Recompensa (ver apêndice I). Nesta versão, parte-se de uma rede que já detém algum conhecimento sobre o domínio do problema (obtida por exemplo pela conversão de grafos de conhecimento, eliciados de um especialista, para uma RNA), ou de um treinamento anterior. Na versão de refinamento, somente as hipóteses solução e as hipóteses erroneamente aceitas são processadas em cada exemplo.

O algoritmo parte dos valores pré-existentes nos acumuladores de recompensas e punições. Uma regra delta é empregada, permitindo modular as punições e recompensas de acordo com os desvios ocorridos nas crenças computadas para as hipóteses.

A rede resultante da aplicação do algoritmo de Punição e Recompensa, para se tornar operante, deve passar pelo algoritmo de Poda e Normalização (ver apêndice I). O objetivo do processo de poda é remover todas as sinapses fracas ou negativas (aquelas com valores líquidos negativos ou baixos em seus acumuladores). Isto é feito definindo-se um Limiar de Poda, para os pesos sinápticos (o qual não deve exceder $\sqrt{L_{\text{aceit}}}$). O objetivo do processo de normalização é normalizar os valores líquidos dos acumuladores (recompensas - punições) para o intervalo $[0, 1]$ produzindo uma rede operante.

O algoritmo identifica e retém todos os caminhos patognomônicos (aqueles com valor nulo no acumulador de punições e positivo no acumulador de recompensas), mesmo quando apresentam valores líquidos pequenos em seus acumuladores. Isto é feito atribuindo-se pesos para suas sinapses com valores maiores que $\sqrt{L_{\text{aceit}}}$, o que garante automaticamente a aceitação da hipótese

correspondente. A escolha dos limiares de poda e aceitação são importantes para o bom desempenho da rede. Esses limiares podem ser determinados empiricamente testando-se sistematicamente o modelo contra diversos valores de limiares e seleccionando-se a rede que produz o melhor desempenho.

11122 - O Problema do João, Maria, Pedro e Diana

Este exemplo mostra como treinar uma rede para realizar diagnóstico médico diferencial. Neste caso, trabalha-se apenas com evidências positivas, raciocinando por "default" como usualmente os médicos o fazem. O banco de dados de treinamento é formado pelos quatro seguintes casos de pacientes com a doença d_1 ou d_2 , que apresentam ou não os sintomas s_1, s_2, s_3, s_4 :

João(d_1, s_1, s_2, s_3)

Diana(d_1, s_1, s_2, s_4)

Maria(d_2, s_1, s_3, s_4)

Pedro(d_2, s_2, s_3, s_4)

Aplicando a Versão Arranque do algoritmo de aprendizado a uma rede não treinada de ordem 3 usando apenas sinapses excitatórias, com 4 células de entrada (correspondentes aos sintomas s_1, s_2, s_3, s_4) e 2 células de saída (correspondentes às doenças d_1 e d_2).

A figura 7 mostra a rede podada usando-se um limiar de poda igual a 0.4. Pode-se observar o surgimento de duas diferentes estruturas nesta rede: O nó A representa uma estrutura forte de conhecimento chamada de *gérmen de conhecimento*. Os nós B e C representam informação factual sobre os pacientes João e Diana. Portanto, convivem na mesma rede dados factuais e conhecimento. Trata-se de uma propriedade notável desta rede, que funciona como uma base de conhecimento e uma base de dados. Se for realizada uma poda nessa rede, usando-se um valor de limiar 0.6, somente os gérmenes de conhecimento permanecerão. Note-se que é possível extrair facilmente regras a partir destas redes, como por exemplo:

Se o paciente apresenta os sintomas s_1 e s_2 então o paciente tem a doença d_1 com grau de possibilidade 1

Se o paciente apresenta os sintomas s_3 e s_4 então o paciente tem a doença d_2 com grau de possibilidade 1

Essas duas regras são capazes de discriminar todos os casos das doenças d_1 e d_2 no conjunto de exemplos.

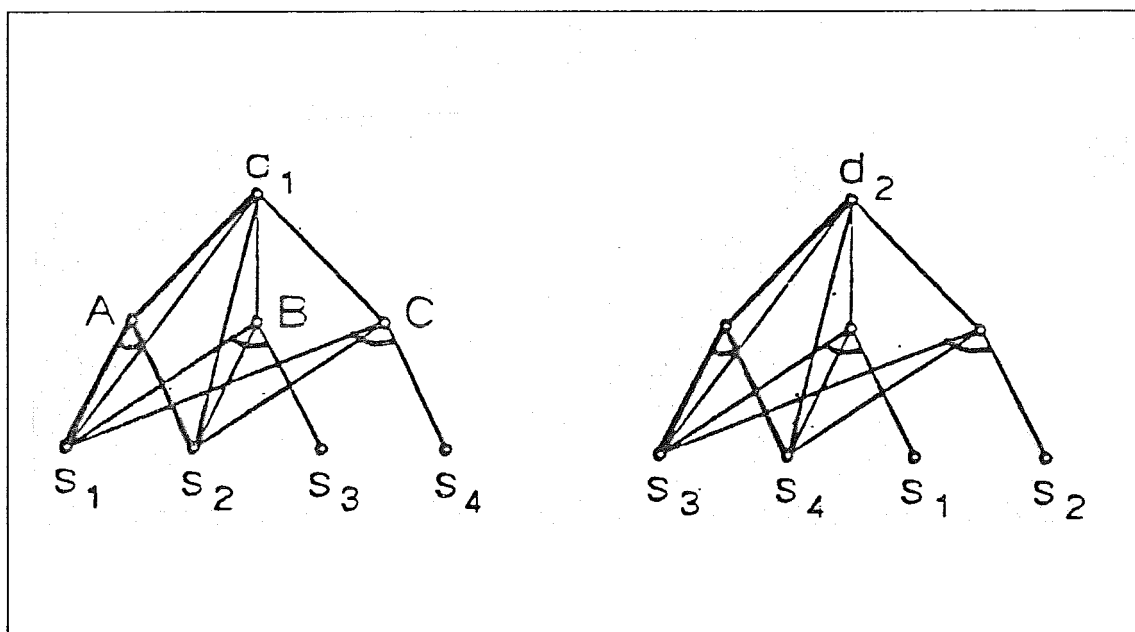


Figura 7. Rede podada usando-se o limiar 0.4 no problema do diagnóstico médico. (Os arcos espessos tem peso 1 e os arcos finos tem peso 0.5).

CAPÍTULO IV

ALGORITMOS GENÉTICOS

Os algoritmos genéticos são algoritmos de busca inspirados em mecanismos de seleção natural [GOLD89]. 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 [AUST90A].

Baseado nesta analogia com o processo de evolução biológica das espécies, chamada de *metáfora* biológica, os algoritmos genéticos 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 algoritmos genéticos foram desenvolvidos por Holland [HOLL75], quando da sua primeira tentativa de compreender os sistemas adaptativos. Em seu trabalho, ele propôs um arcabouço, chamado de sistema classificador, no qual uma população de regras, codificadas como cadeias de bits (0, 1), evolui com base em um dado estímulo intermitente, e com base num reforço do seu meio ambiente, que permite determinar quais respostas são apropriadas em presença do estímulo.

Segundo Booker [BOOK89], os sistemas adaptativos, ou sistemas de aprendizado, enfrentam algum subconjunto dos seguintes problemas:

- a constante novidade encontrada no curso dos dados relacionados com o ambiente, geralmente ruidosos ou irrelevantes (como por exemplo sistemas de reconhecimento de padrões visuais);
- a contínua necessidade de requisitos para ação (encontrado por exemplo em sistemas de controle de robôs);
- objetivos definidos de forma implícita ou inexata (tais como a aquisição de moeda, ouro, ou outro recurso, em ambientes complexos);
- punição ou recompensa esparsa, necessitando de uma longa seqüência de passos para a ação (por exemplo: a busca de alimento por um organismo).

Com o objetivo de enfrentar estes problemas os sistemas de aprendizado devem ser capazes de:

- criar categorias que identifiquem as regularidades relevantes no seu ambiente;
- usar o fluxo de informação encontrado ao longo do caminho até o objetivo para refinar continuamente o seu modelo do ambiente;
- atribuir ações apropriadas para as categorias intermediárias encontradas no caminho para o objetivo.

Estes sistemas devem constantemente absorver novas informações e planejar uma classificação das hipóteses conflitantes sem comprometer as capacidades que já possuem. Requisitos de consistência são substituídos pela competição entre alternativas.

Uma definição de algoritmos genéticos, elaborada por Grefenstette [GREF86], é apresentada a seguir:

Os algoritmos genéticos (AGs) são procedimentos iterativos que mantêm uma população de elementos individuais, comumente chamados de cromossomos, que são considerados soluções ou candidatos a soluções dentro de um domínio

específico, e que permitem determinar quais respostas são mais apropriadas em função do estímulo exterior (ambiente). Durante cada iteração, a população de cromossomos corrente (atual) é rateada em função da sua eficiência como solucionadores do domínio. Com base nesta avaliação, uma nova população de soluções candidatas é formada, mediante a utilização de operadores genéticos (por exemplo operadores de reprodução, de recombinação e de mutação).

Grefenstette [GREF87] apresentou o modelo usual (padrão) para os AGs que é mostrado na figura 8. Neste padrão de AG, a população $P(0)$ é a população inicial dos cromossomos, $A(t)$ é a amostra dos cromossomos de maior aptidão no instante t , e $P(t)$ é a população de cromossomos atualizada.

```
procedimento AG
começo
  inicialize população  $P(0)$ ;
  avalie  $P(0)$ ;
   $t = 1$ ;
  repita
    selecione  $A(t)$  a partir de  $P(t)$ ;
    gere um novo  $A(t)$  usando os operadores genéticos;
    substitua os elementos mais fracos de  $P(t)$  por  $A(t)$ ;
    avalie  $P(t)$ ;
     $t = t + 1$ ;
  até (condição de término);
fim.
```

Figura 8. Padrão de um Algoritmo Genético

A execução do ciclo central pode ser apresentada de maneira mais detalhada como sendo [BOOK89]:

1. A partir do conjunto de cromossomos selecione, de acordo com sua aptidão, os cromossomos mais fortes para serem **pais** que produzem **filhos** através da atuação dos operadores genéticos sobre seus componentes;
2. Aplique os operadores genéticos sobre os cromossomos pais criando cromossomos filhos;
3. Substitua os cromossomos mais fracos, existentes na população, pelos filhos recém gerados;
4. Reavalie a população atual usando uma função de avaliação.

Neste capítulo são revistos os AGs. Na seção IV.1, eles são descritos em termos de seus componentes. Na seção subsequente são apresentados o teorema fundamental dos AGs e o teorema de propagação dos esquemas, ambos propostos por Holland [HOLL75]. Na seção IV.3 são apresentadas algumas aplicações dos AGs. Inicialmente, nesta seção são revistos os conceitos de sistemas classificadores e sistemas de atribuição de crédito, também propostos por Holland [HOLL75]. Finalizando esta seção, e o capítulo, são apresentados exemplos de integração entre os AGs e os modelos conexionistas.

IV.1 - COMPONENTES DOS ALGORITMOS GENÉTICOS

De acordo com Montana [MONT89], os AGs podem ser descritos em função de cinco componentes principais:

- um conjunto de cromossomos;
- uma função de avaliação;
- um procedimento de inicialização;
- um conjunto de operadores genéticos; e
- um conjunto de parâmetros.

Conjunto de Cromossomos

Geralmente, o ambiente, com suas entradas e saídas, é representado por cadeias de símbolos de comprimento fixo em um alfabeto binário (0, 1). Dentro do sistema, cada ponto do espaço do problema pode ser representado por uma cadeia gerada a partir do alfabeto do ambiente. Esta cadeia, comumente chamada de cromossomo, serve de material genético, e possui posições específicas para representar símbolos [AUST90A].

Em um instante de tempo t qualquer, o sistema mantém uma população $P(t)$ de cromossomos, que representam o conjunto solução corrente do problema. No sistema classificador de Holland [HOLL75], o cromossomo representa uma regra com a forma de condição/ação representada sob a forma de uma cadeia de tamanho fixo (k -símbolos) usando um alfabeto binário $\{0,1\}$ estendido com o símbolo de *sem importância* (#). A parte de condição especifica quais os tipos de mensagem que satisfazem (ativam) a regra e a parte da ação especifica qual a mensagem que deve ser enviada quando a regra for satisfeita.

Os classificadores individuais devem ser simples, de definição compacta visando a torná-los úteis para o aprendizado. Uma definição complexa torna difícil ao algoritmo de aprendizado achar e explorar os blocos de construção, também chamados de aglomerados de regras, com os quais constroem-se novas regras [BOOK89]. O maior obstáculo é criar uma especificação simples para a parte de condição da regra. Cada condição deve especificar exatamente o conjunto de mensagens que a satisfaz.

Apesar de muitos conjuntos necessitarem uma lista explícita para sua definição, existe uma classe de subconjuntos, chamados de hiperplanos, que pode ser especificada de maneira compacta [GOLD89]. - Especificamente, seja $\{1,0\}^k$ o conjunto das possíveis mensagens de k -bits. Usando-se o símbolo # como um indicador de sem importância pode-se designar o conjunto de hiperplanos como o conjunto de todas as cadeias ternárias de comprimento k sobre o alfabeto $\{1,0,\#\}$. Por exemplo, a cadeia $1###\dots\#$ especifica o conjunto de todas as mensagens que começam com 1, enquanto a cadeia $00\dots0\#$ designa o conjunto $\{00\dots01,00\dots00\}$ consistindo exatamente de duas mensagens, etc. É

fácil checar quando uma dada mensagem satisfaz a condição. A condição e a mensagem são comparadas posição a posição. A mensagem satisfaz à condição se as entradas em todas as posições diferentes de # forem idênticas.

A forma do classificador é apresentada a seguir. Seja M_1 uma condição e M uma mensagem que é adicionada à lista de mensagens se a condição M_1 for satisfeita. O formato do classificador é M_1/M , onde a condição M_1 é separada da ação M por uma barra.

Desta maneira operações lógicas primitivas (AND, OR e NOT) podem ser realizadas através da combinação de condições. A condição AND pode ser expressa por um classificador com múltiplas condições, como por exemplo $M_1, M_2/M$, sendo que M só será adicionada à lista de mensagens se M_1 e M_2 estiverem na lista. De forma semelhante os pares de classificadores M_1/M e M_2/M expressam a condição OR, onde M é adicionado à lista se M_1 , ou M_2 , ou ambos estiverem na lista. A condição NOT é expressa por um classificador com a condição antecedida por um sinal negativo, por exemplo: $-M_3/M$, onde M só é ativada se ocorrer a condição $-M_3$.

A partir destas primitivas pode-se construir expressões lógicas mais complexas. Considere a seguinte expressão booleana

$$(M_1 \text{ AND } M_2) \text{ OR } ((\text{NOT } M_3) \text{ AND } M_4).$$

Ela é expressa pelo conjunto de classificadores abaixo. Observe que a mensagem M só será incluída na lista de mensagens se a expressão booleana for satisfeita.

$$M_1, M_2/M, -M_3, M_4/M.$$

Esta representação binária para os cromossomos é a utilizada pelos AGs clássicos, por exemplo o de Holland [HOLL75]. Em versões mais recentes dos AGs, podemos encontrar cromossomos que estão representados por números

reais, por exemplo o de Montana [MONT89].

Função de Avaliação

A chave para se entender os AGs é a compreensão de como eles manipulam uma classe especial de blocos de construção chamados *esquemas*. Um esquema é um conjunto de todas as cadeias que possuem certos valores definidos em posições designadas dentro das mesmas [BOOK87]. Ele é definido sobre o alfabeto $0, 1, \#, *$, onde $\#$ indica que esta posição a nível de cromossomo e sem importância. Um novo símbolo de sem importância é o $*$, que indica que é sem importância a nível de esquema.

Por exemplo, o conjunto de todas as cadeias que possuem o valor 0 na primeira posição é um esquema que possui apenas um valor definido. Este esquema pode ser especificado pela cadeia $0***...***$, onde cada $*$ indica que esta posição não é importante para a definição do mesmo a nível de cromossomo. As cadeias que pertencem à região do espaço de busca designado por um esquema são chamadas de *instâncias* deste esquema. Cada cadeia de comprimento k é uma instância de exatamente 2^k esquemas distintos. O conjunto de todos os esquemas de comprimento k pode ser definido pelo alfabeto $\{1, 0, \#, *\}^k$.

O AG envia as construções futuras em direção a usar bons blocos de construção. Veremos como o AG rapidamente explora o espaço de esquemas, que é um espaço muito grande, implicitamente, ordenando e explorando os esquemas de acordo com a força das regras que os empregam.

Eventualmente, os sistemas classificadores podem possuir muitas instâncias de um esquema S qualquer. A média das aptidões destas instâncias, chamada de Potência Média de S , é calculada pela fórmula

$$f(S,t) = (1/\{\text{número de instâncias de } S\}) \cdot \sum f(C,t),$$

onde C é instância de S . Supondo um sistema que possua os seguintes classificadores em um dado instante t : C_1 com a condição $11###110...0$ e

aptidão $f(C_1,t)=4$, e C_2 com a condição $00\#\#1011\dots 1$ e aptidão $f(C_2,t)=2$. Se existirem apenas estas duas instâncias do esquema $S = *0\#\#1^{**}\dots*$ a média das aptidões deste esquema será igual a

$$f(S,t) = 1/2.[f(C_1,t) + f(C_2,t)] = 3.$$

Este valor pode ser considerado como uma estimativa do valor médio de S . Tal estimativa pode induzir o sistema ao erro, mas ela é suficientemente boa para servir de guia heurístico se o sistema possuir procedimentos que compensem estimativas errôneas. Isto o AG faz através da avaliação de exemplos adicionais para o esquema, ou seja, ele constroi novos classificadores (instâncias) e os submete ao algoritmo "Bucket Brigade" (ver seção IV.3.1) [BOOK89].

Chamando $f(t)$ a média das aptidões médias dos esquemas pode-se usar este valor para direcionar a construção de mais instâncias de esquemas com $f(S,t)$ acima da média $f(t)$, em detrimento de instâncias de esquemas abaixo da média. O esquema S está acima da média se $f(S,t)/f(t) > 1$. Seja $N(S,t)$ o número de instâncias do esquema S no sistema em um instante t , e $N(S,t+1)$ o número de instâncias de S após a construção de n novos classificadores. Uma heurística para utilizar a informação $f(S,t)/f(t)$ teria de precisar o aumento ou a diminuição do número de instâncias de S no momento $t+T$ segundo o rateio

$$N(S,t+T) = c.[f(S,t)/f(t)].N(S,t),$$

onde c é uma constante arbitrária. Desse modo privilegia-se o crescimento do número de instâncias dos esquemas com maior aptidão média.

Procedimento de Inicialização

O processo de inicialização pode ser realizado de diversas maneiras. O AG clássico utiliza um procedimento de geração que segue a heurística acima, a qual apresenta algumas vantagens. A amostragem de instâncias de cada esquema é realizada com intensidade proporcional a sua aptidão média. Apenas esquemas com aptidão média elevada (acima de $f(t)$) são amostrados.

Isto direciona a média geral para cima, propiciando um critério de aumento contínuo que os esquemas devem satisfazer para estar acima da média. Esta heurística, chamada de política do *arrocho*, utiliza uma distribuição de instâncias, não se limitando a trabalhar apenas com a melhor e mais recente das mesmas.

Embora estas possibilidades existam em princípio, não há um meio de calcular e usar o conjunto de médias $\{f(S,t)/f(t)\}$ devido a sua enorme cardinalidade. Entretanto, os AGs são capazes de fazer implicitamente o que é impossível explicitamente conforme mostrado a seguir. O algoritmo atua sobre um conjunto $B(t)$ de n cadeias $\{C_1, C_2, \dots, C_n\}$ usando o alfabeto $\{1, 0, \#\}$ com aptidões $f(C_j, t)$ [BOOK89]. Os passos que o AG segue para gerar novos classificadores são:

1. Calcule a aptidão média $f(t)$ para as cadeias em $B(t)$ e atribua o valor normalizado $f(C_j, t)/f(t)$ para cada cadeia C_j em $B(t)$;
2. Atribua a cada cadeia em $B(t)$ uma probabilidade proporcional ao seu valor normalizado. Usando esta distribuição de probabilidade selecione m pares de cadeias a partir de $B(t)$, sendo $m \ll n$ e n o tamanho da população, e faça cópias delas (usando o operador de reprodução);
3. Aplique os operadores genéticos (recombinação, mutação, etc) para cada par copiado, formando $2m$ novas cadeias;
4. Atribua uma aptidão inicial aos filhos segundo a regra:

$$P_{t_{ini}} = k \cdot \{\text{média das aptidões dos pais}\}$$

$$k = \text{constante}, 0 < k \leq 1;$$
5. Substitua as $2m$ cadeias de menor aptidão em $B(t)$ pelas $2m$ cadeias geradas, caso estas sejam maiores que aquelas;
6. Aplique o Sistema de Atribuição de Crédito para reavaliar as aptidões de cada cadeia;
7. Adicione 1 a t e volte ao Passo 1.

O AG executa implicitamente o esquema de busca heurística descrito anteriormente. Pode ser provado que os operadores genéticos do Passo 3 mantem a ênfase do esquema de busca heurística proposto, bem como propiciam instâncias novas para os vários esquemas em $B(t)$ de acordo com aquela ênfase. Esta prova, elaborada por Holland [HOLL75], é considerada o teorema fundamental dos AGs e é apresentada na seção IV.2.

Algumas versões de AGs encontradas na literatura utilizam, no procedimento de inicialização, uma distribuição exponencial de $e^{-|x|}$, por exemplo o AG de Montana [MONT89], outros utilizam uma distribuição probabilística uniforme entre $-1,0$ e $1,0$, por exemplo o AG de Whitley [WHIT90].

Conjunto de Operadores Genéticos

Os operadores genéticos são responsáveis pelo processo de *mutação adaptativa* [WHIT90]. Este processo é uma forma de manter a diversidade da população, em resposta a crescente homogeneidade genética da mesma. Apresentamos a seguir uma relação dos principais operadores genéticos encontrada na literatura:

1. Operador de Reprodução;
2. Operador de Recombinação;
3. Operador de Mutação;
4. Operador de Gradiente.

Operador de Reprodução

Nas versões mais simples dos algoritmos genéticos, o operador de reprodução é implementado como uma busca através da população de cromossomos, gerando cópias iguais às daqueles já existentes na população. Os cromossomos que possuem maior aptidão tem uma probabilidade maior de gerar filhotes. Um classificador com múltiplas cópias na população tem maiores chances de ser selecionado no futuro.

Operador de Recombinação

Um operador genético importante é o de recombinação, que troca um segmento selecionado aleatoriamente entre os pares de cromossomos pais. Este operador é aplicado aos pares de cadeias da seguinte forma: selecione uma posição aleatória i , $1 \leq i \leq k$, e permuta os segmentos a esquerda da posição i em duas cadeias. Na figura 9 temos um exemplo de operação de recombinação.

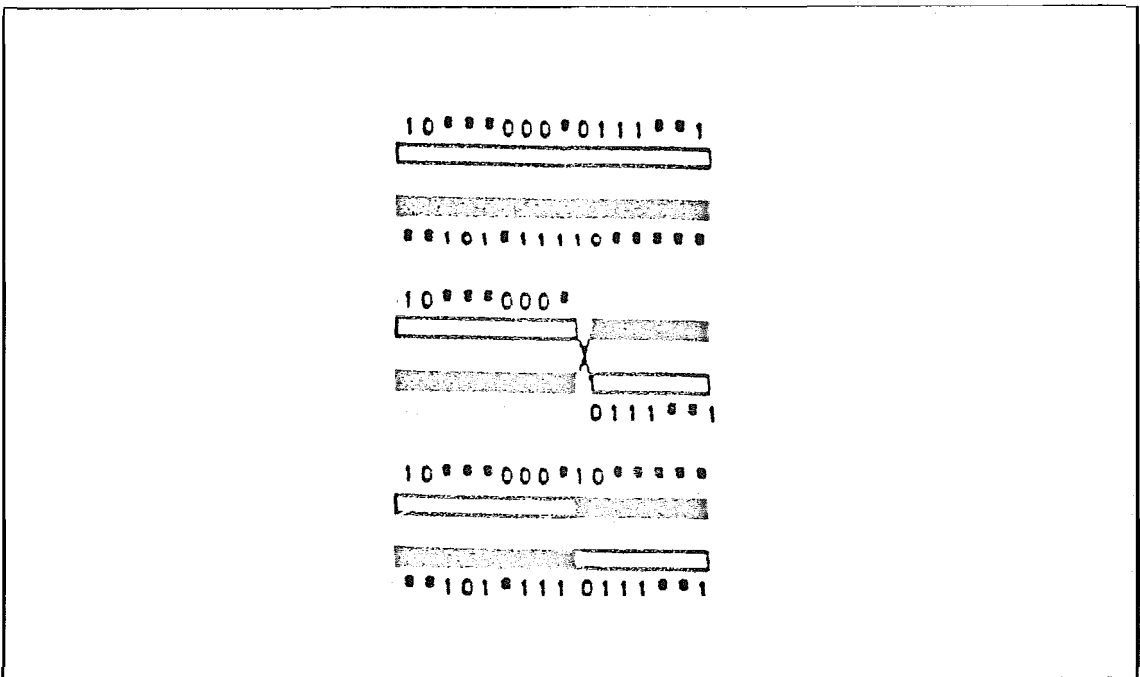


Figura 9. Exemplo de operação de recombinação

Operador de Mutação

Um outro tipo de operador importante é o operador de mutação. Este operador gera um filho a partir da modificação (substituição, adição ou eliminação) de um ou mais elementos que compõem a cópia do cromossomo pai [AUST90A].

Operador de Gradiente

O operador de gradiente seleciona um cromossomo pai, e gera um filhote adicionando às entradas um múltiplo do gradiente em relação a função de

avaliação. Um exemplo deste tipo de operador foi utilizado no experimento de Montana [MONT89]. Neste experimento, os pesos são ajustados apenas depois de calcular o gradiente para todos os membros do conjunto de treinamento. Na figura 10 temos um exemplo de operação dos operadores de mutação, recombinação e gradiente.

Goldberg [GOLD89] apresenta os operadores de inversão e reordenamento. Estes operadores são utilizados na manutenção de esquemas de comprimento muito grande. Tais esquemas possuem um tempo de sobrevivência muito baixo, pois os operadores de recombinação tendem a romper os mesmos. Estes operadores foram aplicados em alguns jogos, e seu desempenho foi decepcionante pelo fato dos domínios dos mesmos não serem suficientemente difíceis de serem solucionados (epistáticos).

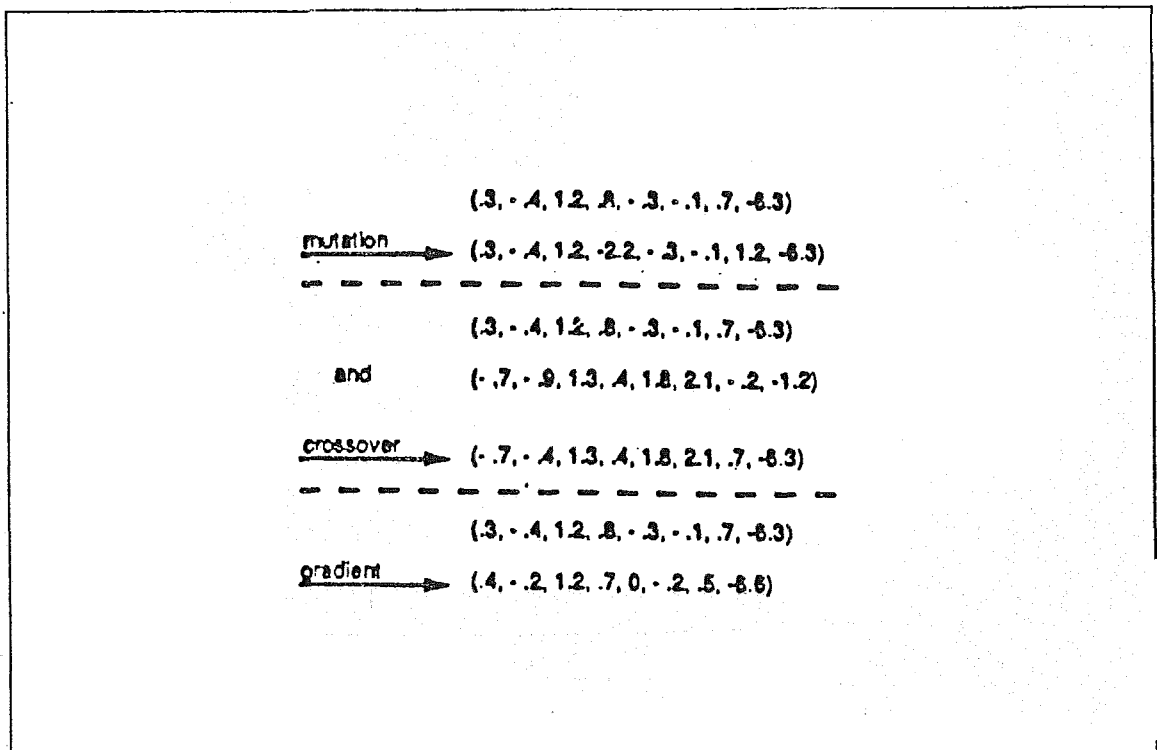


Figura 10. Operação dos operadores genéticos

Conjunto de Parâmetros

Os parâmetros são valores que influenciam no desempenho do AG. Apresentamos uma relação de parâmetros, baseado em Austin [AUST90A], a seguir:

1. tamanho da população;
2. taxa de operadores;
3. intervalo de geração;
4. seleção de estratégia;
5. fator de escalada.

Tamanho da População (N)

O tamanho da população de cromossomos afeta o desempenho global dos AGs. 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.

Taxa de Operadores (T)

As taxas dos operadores medem a frequência com que cada tipo de operador genético é utilizado. Representam também a influência que cada tipo de operador está exercendo sobre a população de cromossomos. Por exemplo, analisando as taxas dos operadores de recombinação e mutação.

Se a taxa dos operadores de recombinação for muito alta, alguns cromossomos de bom desempenho podem ser removidos mais rapidamente do que a seleção possa desenvolvê-los. Se a taxa de recombinação for muito baixa, a busca pode estagnar.

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 muito alta resulta essencialmente numa busca aleatória.

Intervalo de Geração (G)

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.

Seleção de Estratégia (S)

As estratégias de seleção correspondem aos critérios utilizados para a escolha de cromossomos durante a reprodução. Um exemplo de estratégia de reprodução, baseado em Austin [AUST90A], é a *seleção pura* onde os cromossomos são reproduzidos em função da sua aptidão.

Fator de Escalada

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 a diversidade da população. Esta situação deve ser evitada para que o espaço de busca seja suficientemente explorado. 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.

IV.2 - TEOREMA FUNDAMENTAL DOS ALGORITMOS GENETICOS

Este teorema foi reescrito como um procedimento para o enviezamento progressivo da distribuição de probabilidades sobre o espaço $\{1,0,\#\}^k$.

Teorema. Seja P_{cross} a probabilidade que um par selecionado tem de ser recombinado e seja P_{mut} a probabilidade de que uma mutação possa ocorrer. Se $p(S,t)$ é a fração da população ocupada pelas instâncias de S no instante t , logo

$$p(S,t+1) \geq [1 - \lambda(S,t)] \cdot [1 - P_{\text{mut}}]^{d(S)} \cdot [f(S,t) / f(t)] \cdot p(S,t)$$

fornece a fração esperada da população ocupada pelas instâncias de S no instante $t+1$ sobre o algoritmo genético.

A explicação dos termos que compõem o teorema fundamental, extraída de Holland [HOLL75], é a seguinte:

- $[f(S,t) / f(t)]$ é a aptidão média observada do esquema S em relação a aptidão média de toda a população;
- $[1 - \lambda(S,t)] [1 - P_{\text{mut}}]^{d(S)}$ são termos de erro que resultam da dispersão das instâncias de S devido à recombinação e à mutação respectivamente:
 - $\lambda(S,t) = P_{\text{cross}}(l(S)/k) \cdot p(S,t)$ é o limite superior das perdas das instâncias de S resultantes de recombinações que fogem ao esquema original. Isto ocorre quando o ponto de recombinação cai dentro do intervalo de comprimento $l(S)$ determinado pelos locais mais externos do esquema;
 - $[1 - P_{\text{mut}}]^{d(S)}$ fornece a proporção de instâncias de S que escapam do esquema original devido a uma mutação que ocorre em um dos pontos $d(S)$ que definem o esquema S .

Holland [HOLL75] apresentou também uma proposta de modelagem do processo de propagação dos esquemas que é definido pelo seguinte teorema:

Teorema. Selecione um limite ϵ para o erro de transcrição decorrente da reprodução e recombinação. Escolha l tal que $l/k \leq \epsilon/2$. Então em uma população de tamanho $M = C_1 \cdot 2^{l/k}$, obtida a partir de uma amostra aleatória uniforme de $\{1,0\}^k$, o número de esquemas propagados com um erro $< \epsilon$ excede largamente M^3 .

A demonstração de ambos os teoremas pode ser encontrada em Holland [HOLL75]. Durante este procedimento instâncias de esquemas que não existiam previamente podem ser gerados. A manipulação implícita de um grande número de esquemas através da operação de $2n$ mensagens a cada passo é chamada de **paralelismo implícito** [BOOK89]. Portanto, com a modelagem apresentada anteriormente, concluí-se que o AG produz, de forma implícita, justamente o processo de amostragem sugerido pela heurística.

IV.3 - APLICAÇÕES DOS ALGORITMOS GENÉTICOS

Os AGs são muito flexíveis e tem sido utilizados em uma variedade de aplicações. Uma relação destas aplicações, baseada em Booker [BOOK89], Goldberg [GOLD89] e Austin [AUST90A], é apresentada a seguir:

- Processamento de Imagens;
- Processamento de Sinais;
- Controle de Processos Dinâmicos;
- Reconhecimento de Padrões;
- Aplicações Militares;
- Construção de Autômatos Finitos a partir de exemplos;
- Escalonamento de Tarefas;
- Busca em Grafos;
- Problema do Caixeiro Viajante;

- Simulação de Modelos Biológicos de Comportamento;
- Descoberta de Novas Conectividades em Redes Neurais; e
- Processamento de Linguagem Natural (ex: reconhecimento de tempo de verbos).

Antonisse e Keller [ANTO87] apresentam um experimento com o uso de operadores genéticos para capturar características da descrição e do processamento de domínios de conhecimento de ordem superior. Este AG foi utilizado em aplicações militares.

Grefenstette [GREF88] apresenta um sistema classificador, chamado RUDI, que utiliza algoritmo genético para realizar a atribuição de crédito em sistemas de descoberta de regras. Neste trabalho, Grefenstette apresenta resultados analíticos e experimentais que suportam a hipótese de que vários níveis de atribuição de crédito podem melhorar o desempenho dos sistemas classificadores.

Booker [BOOK88] apresenta o GOFER, que é um sistema classificador que utiliza o AG para criar representações internas (modelos) do ambiente. Este sistema usa regras para representar objetos, metas e relacionamentos. O modelo é usado para direcionar o comportamento, e o aprendizado é disparado sempre que este modelo provar ser inadequado para gerar um comportamento para uma dada situação. Na seção IV.3.1 são revistos os conceitos de sistemas classificadores.

Também dentro do campo de modelos conexionistas, os algoritmos genéticos vem atraindo atenção como uma forma de buscar uma rede neuronal ótima para um determinado problema. Na seção IV.3.2 são reportados dois experimentos, o primeiro elaborado por Montana e Davis [MONT89], e o segundo realizado por Whitley et alli [WHIT90], que integram algoritmos genéticos com modelos conexionistas.

IV.3.1 - Visão Geral do Sistema Classificador

Quando se focaliza a questão do aprendizado em sistemas classificadores torna-se interessante distinguir três níveis de atividade, a saber [BOOK89]: *Sistema Classificador Propriamente Dito* (SCPD), *Sistema de Atribuição de Crédito* (SAC) e *Sistema de Descoberta de Regras* (SDR). Na figura 11 é apresentada a organização geral do sistema classificador.

O SCPD está no nível mais baixo e corresponde à parte do sistema que interage diretamente com o ambiente. Ele assemelha-se a um SE, apesar de ser menos dependente do domínio. Os SCPDs serão discutidos com mais detalhes na sub-seção IV.3.1.1.

O segundo nível de atividade, o SAC, se justifica em função da necessidade que o sistema tem de identificar quais as regras que efetivamente contribuem para a solução do problema colocado pelo ambiente. Em geral as regras do SCPD variam quanto à utilidade e muitas podem estar incorretas. O sistema deve avaliar estas regras de alguma forma. Tal atividade é chamada de *atribuição de crédito*. Uma classe particular de algoritmos para executar esta tarefa, chamados de "Bucket Brigade", serão discutidos na sub-seção IV.3.1.2.

O terceiro nível de atividade corresponde ao Sistema de Descoberta de Regras (SDR). Tal sistema se justifica pois mesmo que milhões de regras sejam avaliadas, isto pode corresponder a uma minúscula parte das regras úteis plausíveis. A concentração do esforço na seleção das melhores dentre estes milhões de regras não dá a garantia de que o sistema exauriu suas possibilidades de desenvolvimento.

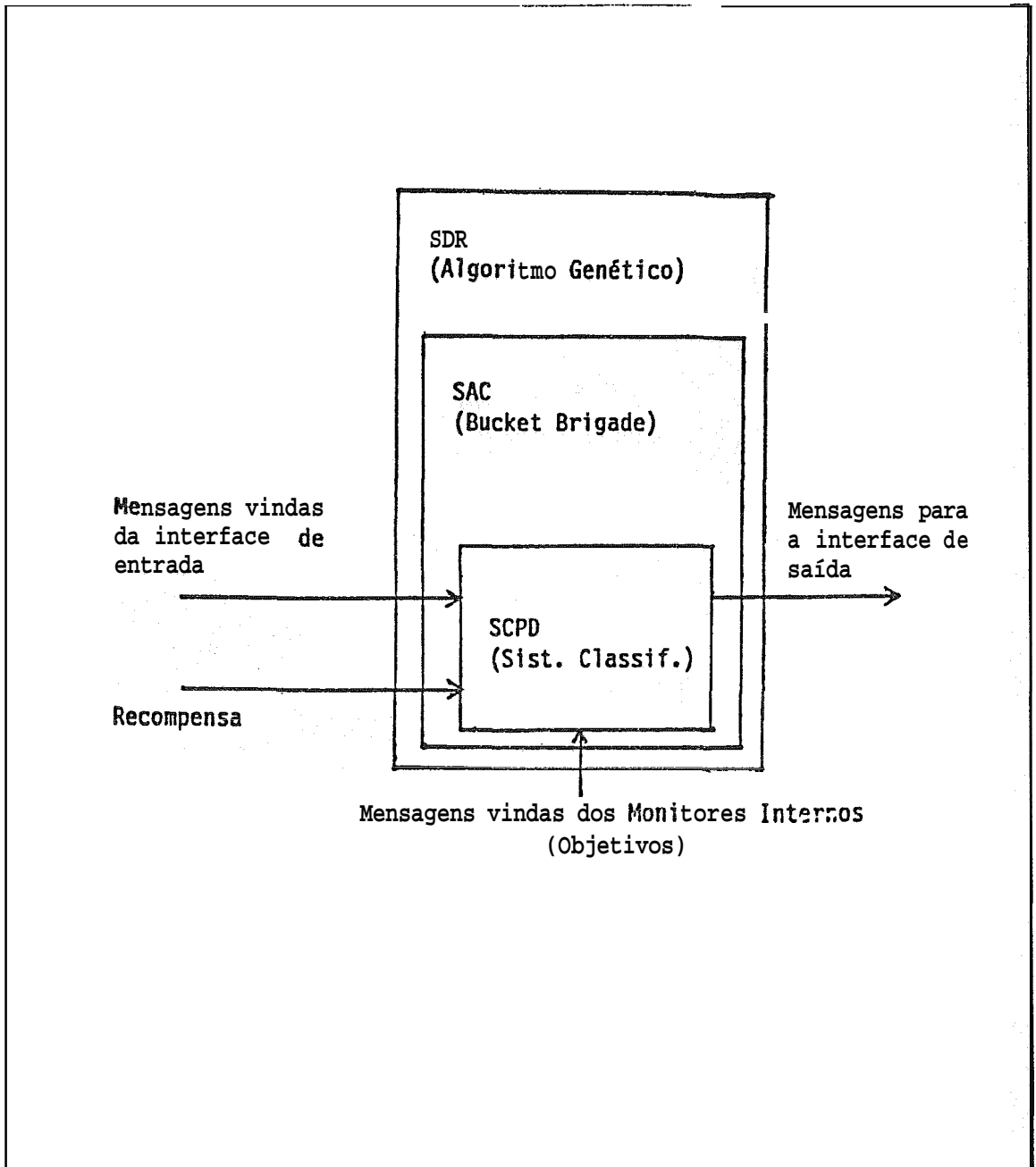


Figura 11. Organização Geral de um Sistema **Classificador**

Eventualmente, nenhuma das regras examinadas *É* boa. Assim sendo o sistema deve ser capaz de gerar novas regras em substituição às menos úteis existentes no momento. As regras podem ser geradas de forma aleatória, através de operadores de mutação, ou por intermédio da execução de uma enumeração pré-determinada.

Porém tais procedimentos independentes da experiência produzem melhoramentos de maneira muito lenta, inviabilizando sua aplicação em problemas reais. De alguma forma, o procedimento de descoberta de regras deve usar a experiência acumulada no sistema para determinar quais blocos de construção são úteis para a formação de regras plausíveis. O SDR aqui discutido emprega os AGs, os quais foram analisados na seção IV.1.

As regras em um sistema classificador formam uma população de indivíduos evoluindo em função do tempo. Se o sistema classificador for bem projetado as interações das regras com o AG produzirão um sistema capaz de maximizar o reforço positivo oriundo do ambiente [DAVI87].

IV.3.1.1 - Sistemas Classificadores Propriamente Ditos (SCPDs)

Os Sistemas Classificadores Propriamente Ditos (SCPDs) são formados por regras, chamadas de classificadores, que possuem as seguintes peculiaridades [GOLD89]: elas trabalham enviando mensagens, possuem elevado paralelismo e são altamente padronizadas (ver seção IV.1). Na figura 12 tem-se os elementos básicos do sistema classificador.

A lista de mensagens contem todas as mensagens atuais, ou seja, aquelas geradas pela interface de entrada e aquelas geradas pelas regras satisfeitas.

A interface de saída traduz algumas mensagens em ações efetivas que são ações que modificam o estado do ambiente.

A execução do ciclo básico do sistema classificador consiste dos seguintes passos [BOOK89]:

1. Adicione todas as mensagens da interface de entrada à lista de mensagens;
2. Compare todas as mensagens da lista de mensagens com as condições de todos os classificadores e registre os casamentos ocorridos;
3. Para cada conjunto de casamentos, satisfazendo a parte de condição de algum classificador, envie a mensagem, correspondente à parte de ação deste classificador, para uma lista de novas mensagens;
4. Substitua a lista atual de mensagens pela lista de novas mensagens;
5. Traduza as mensagens da lista de mensagens para ações usando a interface de saída, produzindo assim a saída atual do sistema;
6. Retorne ao Passo 1.

IV.3.1.2 - Sistema de Atribuição de Crédito (SAC)

A principal tarefa de aprendizado que qualquer sistema baseado em regras operando num ambiente complexo enfrenta é a tarefa de atribuição de crédito. De alguma forma, o sistema deve identificar quais as regras responsáveis pelo seu sucesso e a relevância das condições encontradas na obtenção deste sucesso [GREF88].

Os algoritmos "Bucket Brigade" são projetados para resolver tal problema de atribuição de crédito em sistemas classificadores. O objetivo final é atribuir a cada classificador um valor associado chamado *aptidão*. O algoritmo "Bucket Brigade" ajusta esta aptidão para refletir a utilidade do classificador em

relação a todo o sistema. Portanto, a aptidão é usada como base para a competição. A cada passo todos os classificadores satisfeitos fazem uma concorrência baseada na sua aptidão. Apenas os classificadores com os lances mais altos tem o direito de incorporar suas mensagens na lista de mensagens para o próximo passo [BOOK89].

Este processo de concorrência é especificado a seguir. Seja $f(C,t)$ a aptidão do classificador C no instante t . Dois fatores são importantes para o processo de concorrência: primeiro a relevância com relação à situação atual e segundo a utilidade demonstrada no passado. A relevância é, principalmente, uma questão de especificidade da parte de condição das regras, ou seja, uma condição mais específica, satisfeita pela situação atual, contem mais informação sobre aquela situação. A aptidão da regra reflete sua utilidade passada.

Em algumas versões mais simples de competição, o lance é o produto destes dois fatores. O valor resultante será 0 no caso da regra ser irrelevante, ou seja a condição não satisfaz à situação atual, ou no caso de sua utilidade (aptidão) ser 0. O valor resultante será elevado se a regra for altamente específica para a situação e se sua utilidade for elevada. Para implementar o processo de concorrência modifica-se o Passo 3 no ciclo de execução (ver seção IV.3.1.1).

3) Para cada conjunto de comparações que satisfaçam a parte de condição do classificador C calcule o lance de acordo com a seguinte fórmula:

$$B(C,t) = b.R(C).f(C,t),$$

onde $f(C,t)$ é a aptidão, $R(C)$ é a especificidade, que é igual ao número de não_# na parte da condição de C dividido pelo comprimento da mensagem, e b é uma constante menor que 1 (em geral 0,125 ou 0,0625). O valor do lance determina a probabilidade de que o classificador coloque sua mensagem na nova lista de mensagens.

Quando um classificador vencedor C coloca sua mensagem na lista de mensagens ele deve pagar por este privilégio tendo sua aptidão $f(C,t)$ reduzida pelo valor do lance $B(C,t)$. Logo a fórmula é

$$f(C,t+1) = f(C,t) - B(C,t).$$

Por sua vez os classificadores $\{C'\}$, responsáveis pela ativação de C (fornecedores), terão sua aptidão aumentada de uma fração do valor do lance $B(C,t)$ pago por C . Os fornecedores são os classificadores cujas mensagens casaram com a parte de condição de C . O valor do lance de C é distribuído entre os elementos de $\{C'\}$ da seguinte forma:

$$f(C',t+1) = f(C',t) + a.B(C,t),$$

onde $a = 1/(\text{número de membros de } \{C'\})$.

IV.3.2 - INTEGRANDO AGS E MODELOS CONEXIONISTAS

No primeiro trabalho são treinadas redes neuronais acíclicas usando AG [MONT89]. O objetivo deste experimento é classificar corretamente os chamados "lofargrams" que são imagens baseadas em intensidade acústica de sonar. Esta energia é função de frequência e tempo, sendo representada sob a forma de espectrogramas. .

No segundo caso são relatados experimentos que usam AG na otimização dos pesos das conexões de redes acíclicas, em problemas clássicos como o OU-exclusivo e em duas versões do problema da adição (somador de bits de 2 posições). São também relatados experimentos que mostram como os algoritmos genéticos podem ser usados para encontrar conectividades interessantes para pequenas redes conexionistas [WHIT90].

IV.3.2.1 - Elementos dos AGs Propostos

Na tabela 7 temos uma análise comparativa dos elementos dos AGs dos dois experimentos.

No trabalho de Montana [MONT89], a base de dados inicial é formada de 1200 pedaços de "lofargrams" retangulares de tamanho fixo. Cada um destes é centralizado em um tipo de linha determinado por um especialista, sendo que apenas 30% são interessantes. O conjunto de treinamento é composto de um subconjunto de 236 exemplos. O experimento consiste em treinar uma rede acíclica para classificar quais os tipos de linha que são interessantes e quais não são.

Neste trabalho é definido um conjunto de parâmetros que influenciam o desempenho do algoritmo. Em geral, seus valores são constantes durante a execução do AG. Os principais parâmetros são: "Parent-Scalar" que determina com qual probabilidade um indivíduo é escolhido para ser pai, e "Operator-Probabilites" que é uma lista de parâmetros que determina qual a probabilidade que cada operador do conjunto de operadores tem de ser selecionado pelo AG.

Tabela 9. Comparação entre os elementos que compõem os AGs de Montana e Whitley.

ELEMENTOS DO AG	AG DE MONTANA	AG DE WHITLEY
Cromossomos	Os pesos na rede são codificados como uma lista de números reais.	Os pesos das conexões das redes são codificadas como cadeias binárias. Oito bits são usados para representar cada peso, e são interpretados como um número no intervalo de -127 a 127.
Função de Avaliação	A rede é processada sobre o conjunto de treinamento e é retornado o somatório dos quadrados dos erros.	Igual à função utilizada por Montana.
Procedimento de Inicialização	Os pesos da população inicial são escolhidos aleatoriamente com a distribuição exponencial de $e^{- x }$.	Os pesos da população inicial são escolhidos em uma distribuição probabilística uniforme entre -1,0 e 1,0.
Operadores Genéticos	Foram criados operadores que podem ser agrupados nas categorias de mutação e recombinação.	Foram criados apenas operadores de mutação.

No trabalho de Whitley [WHIT90] foram implementados dois algoritmos o GENITOR e o GENITOR II. O GENITOR funciona como um AG padrão. Dois pais são selecionados em uma população e cópias destes são recombinadas produzindo dois filhotes. Um destes filhos é descartado aleatoriamente o outro substitui a cadeia de menor aptidão existente. Nunca um filho substitui um pai.

O GENITOR II é uma versão distribuída do GENITOR. Nesta versão, são usadas pequenas representações distribuídas da população de cadeias, em vez de uma única população. Usando várias populações menores em paralelo foi possível explorar a informação dentro de cada subpopulação sem destruir totalmente a diversidade genética da população inteira. Entre estas subpopulações existem trocas periódicas de material genético (cromossomos).

IV.3.2.2 - Principais Características dos Trabalhos

Trabalho de Montana

A principal característica do experimento reside na utilização de operadores *heurísticos*, ou seja, a sua probabilidade de seleção é função do seu desempenho passado. Estes operadores atuam sobre os cromossomos, que representam as redes (ver figura 13), gerando variações dos mesmos.

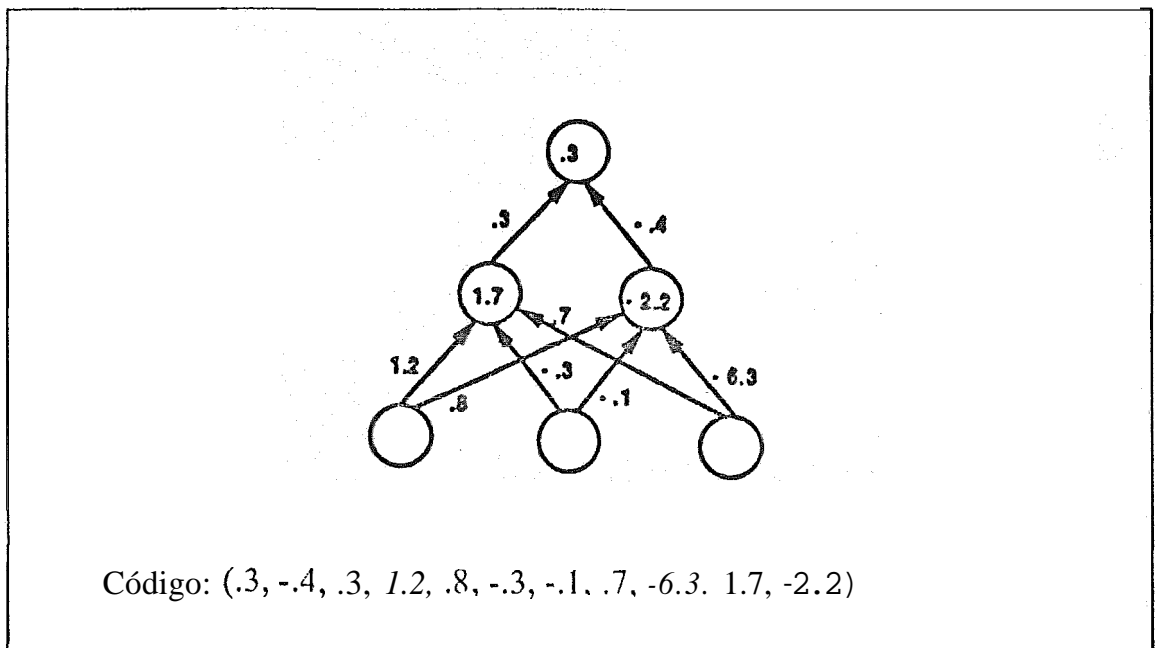


Figura 13. Codificando uma rede em um cromossomo

Foram realizadas análises de desempenho das categorias de operadores de mutação e de recombinação. Dado um conjunto de operadores foram realizadas séries de dez execuções independentes, armazenando a melhor avaliação em função do número de iterações em cada uma delas. Em seguida foi feita a média destes resultados para reduzir as variações introduzidas pela natureza estocástica do algoritmo. Ao final foi feita uma comparação do AG proposto com o modelo "backpropagation" [RUME86B].

O resultado deste último experimento realizado é apresentado na figura 14. De acordo com este gráfico conclui-se que o AG teve um desempenho melhor que o "backpropagation".

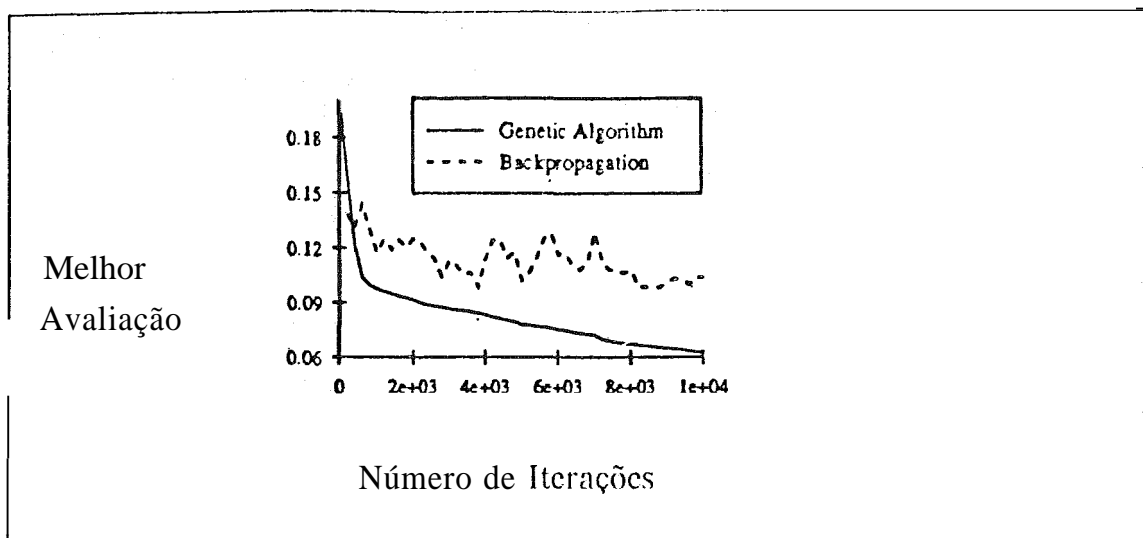


Figura 14. Comparação do AG de Montana com o "backpropagation"

Trabalho de Whitley

Definir a conectividade de uma rede acíclica de tal forma que ela só utilize as conexões estritamente necessárias é uma tarefa difícil e não muito bem compreendida [WHIT90]. Neste caso, inicialmente é definida uma rede bastante grande para realizar uma determinada tarefa. Em seguida, aplica-se o AG para descobrir as combinações de conexões que são suficientes para aprender a executar uma determinada tarefa usando "backpropagation". Esta abordagem é semelhante às técnicas de poda utilizadas nos modelos conexionistas [MACH89].

O GENITOR foi utilizado para podar as ligações em uma rede acíclica. O GENITOR não remove uma conexão de cada vez, mas explora a interação de possíveis ligações para encontrar a combinação de conexões com uma determinada propriedade desejada. As ligações podem ser removidas e reintroduzidas durante a recombinação. Este processo pode ser usado não só para reduzir o tamanho da rede, mas também para descobrir topologias com capacidade de aprendizado.

Para utilizar o AG na definição da conectividade da rede é necessário algum mecanismo que recompense as redes que usam menos ligações. Este é um problema não trivial, pois a recompensa ou a punição direta da rede baseada

apenas no número de conexões existentes pode dar uma vantagem seletiva a redes que não estão aptas a aprender, ou em caso extremo, a rede pode tentar ganhar todas as recompensas ou evitar todas as punições através da poda de todas as suas conexões [WHIT90].

Esta abordagem de *poda* genética foi testada no problema do OU-exclusivo e no somador de bits de 2 posições fortemente conectado. No problema do OU-exclusivo, começou-se com uma rede que tinha dois nós na camada intermediária, e com a adição de ligações diretas entre as camadas de entrada e de saída. O GENITOR reduziu esta rede para uma com apenas uma célula na camada intermediária.

No problema do somador de bits de 2 posições, a rede inicial era formada por quatro nós na camada de entrada, quatro na camada intermediária e três na camada de saída. Acrescentou-se um nó, chamado de "true", que foi utilizado no aprendizado do viés. Inicialmente, existiam ao todo 53 ligações na rede acíclica. O AG reduziu esta quantidade para 29 ligações. Na figura 15 temos a rede final obtida.

Foram realizados testes com a rede obtida, variando aleatoriamente os valores dos pesos em experimentos de aprendizado. Foi tirada uma média com 50 execuções, e o resultado obtido mostrou que esta rede aprendeu de maneira mais rápida que a rede inicial.

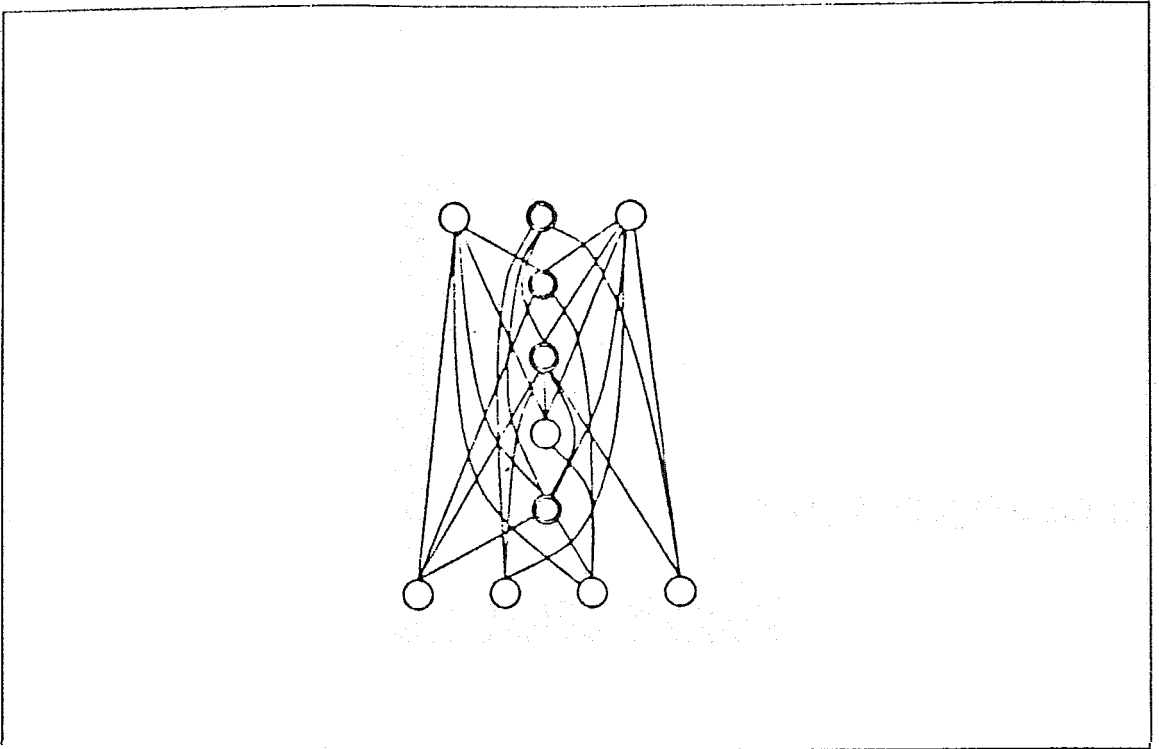


Figura 15. Rede obtida pelo GENITOR para o somador de bits de 2 posições

As principais conclusões do trabalho apresentado por Whitley [WHIT90] poderiam ser resumidas da seguinte maneira: *Existe um grande potencial de uso dos AGs para definir RNAs, onde os pesos e a conectividade são obtidos simultaneamente. Os AGs podem procurar uma conectividade que facilite a busca no espaço do problema enquanto procuram determinar um conjunto apropriado de pesos. Estes dois processos podem ser realizados para complementar um no outro. Assim sendo, consegue-se aumentar a velocidade do aprendizado e melhorar o desempenho da rede. O resultado final são redes menores que aprendem mais rápido.*

CAPÍTULO V

O MODELO CONEXIONISTA EVOLUTIVO (MCE)

Neste capítulo propomos o MCE, um modelo classificador baseado no Modelo Neural Combinatório (MNC) [MACH91], apresentado no capítulo III, capaz de alterar a topologia da rede neural visando alcançar um desempenho ótimo.

Para um domínio de problema pequeno, o MNC pode aprender a partir do zero, executando uma ou mais iterações sobre o conjunto de exemplos pois a explosão combinatória na camada intermediária pode ser evitada através da limitação dos aglomerados de informação originais a uma ordem máxima.

Quando se trata de domínios grandes torna-se necessário a redução drástica desta ordem máxima na rede não treinada ou o uso de heurísticas para reduzir o número de combinações consideradas. Tais heurísticas poderiam fazer uso dos grafos de conhecimento de especialistas, propostos por Rocha [ROCH89], onde os especialistas informam o conjunto de evidências (sintomas) por hipótese (doença), e quais as combinações relevantes para a classificação (diagnóstico).

Infelizmente, isto pode resultar em uma estrutura inicial insuficiente para representar o conhecimento heurístico necessário para se obter um desempenho preciso do sistema, no primeiro caso pela falta de combinações relevantes de ordem superior, ou, no último caso por deficiências, eventuais, no conhecimento eliciado do especialista.

Como uma proposta para suprir esta limitação surge a idéia de dotar o sistema com um *aprendizado evolutivo*, o qual permitiria preencher os vazios ("gaps") existentes no conhecimento atual através da geração de novas regras plausíveis, sem que ocorra a destruição das capacidades já estabelecidas. Este aprendizado evolutivo é obtido mediante o emprego de algoritmos genéticos (AGs).

O MCE é resultante da combinação do MNC com AGs. Os AGs são um método de busca que mantém a informação sobre o ambiente acumulando-a durante o período de adaptação. Os AGs utilizam tal informação acumulada para restringir o espaço de busca e gerar novas soluções plausíveis [AUST90A].

Os AGs têm sido usados para descobrir novas topologias de conectividade em modelos conexionistas ([WHIT90], [MUHL90]). Minsky e Papert, no epílogo de seu livro [MINS88], corroboram esta idéia da seguinte forma: *o futuro dos modelos conexionistas não está relacionado com a busca de um esquema único e universal capaz de resolver todos os problemas de uma vez, mas na evolução de uma tecnologia multi-facetada de projeto de redes.*

Mühlenbein [MUHL90] conjectura que a próxima geração de modelos conexionistas seria composta de redes com a capacidade de evoluir sua topologia. Ele chamou de *redes genéticas* tal geração de modelos.

V.1 - ARQUITETURA DO MCE

Este trabalho se propõe a aplicar os AGs em um sistema especialista conexionista, construído segundo o MNC [MACH89], produzindo um Modelo Conexionista Evolutivo (MCE).

O MCE, a exemplo do MNC, é um modelo classificador envolvendo um conjunto de saídas (hipóteses) e um conjunto de entradas (evidências). O processo de aprendizado no MCE apresenta três ângulos:

- Ajuste de pesos: efetuado pelo algoritmo de punição e recompensa do MNC;
- Poda de sinapses irrelevantes: executada pelo algoritmo de poda do MNC;
- Alteração da topologia da rede: com a criação de novas vias realizada pelo AG.

A figura 16 apresenta a arquitetura do MCE. O elemento básico é a via (também chamada de aglomerado). As vias são sub-redes que ligam um conjunto de evidências a uma determinada hipótese (por exemplo A, B, C na figura 16).

A aptidão de uma via é medida pela sua potência, isto é: o fluxo evidencial corrente que a via injeta na hipótese quando alimentada por evidências plenas.

Como podemos observar as vias ("pathways") do MNC formam a população sobre a qual o AG atuará, pela aplicação de operadores genéticos existentes na base de operadores genéticos.

Visando a determinar qual a população ideal de operadores genéticos, foi introduzida na arquitetura um algoritmo genético de segundo nível, chamado de Algoritmo Meta-Genético (AMG), que se responsabiliza pela evolução nesta população. Além de determinar a população ideal de operadores genéticos, o AMG indica ao AG quais as frequências com que devem ser aplicados os diversos operadores genéticos.

A figura 17 apresenta como exemplo, o processo de reprodução de vias pela operação de recombinação. Sejam a hipótese H1, e as evidências E1, E2, E3 e E4. As evidências E1 e E2 estão conectadas ao aglomerado pai P1, e E3 e E4 estão conectadas ao pai P2, conforme a figura 17.(A). Como consequência dois novos caminhos são formados. No primeiro, as evidências E1 e E3 estão conectadas ao aglomerado filho F1, e no segundo, E2 e E4 estão conectadas ao filho F2, conforme figura 17.(B). Nesta figura está representada uma operação de recombinação que é descrita com mais detalhes na seção V.2.1.

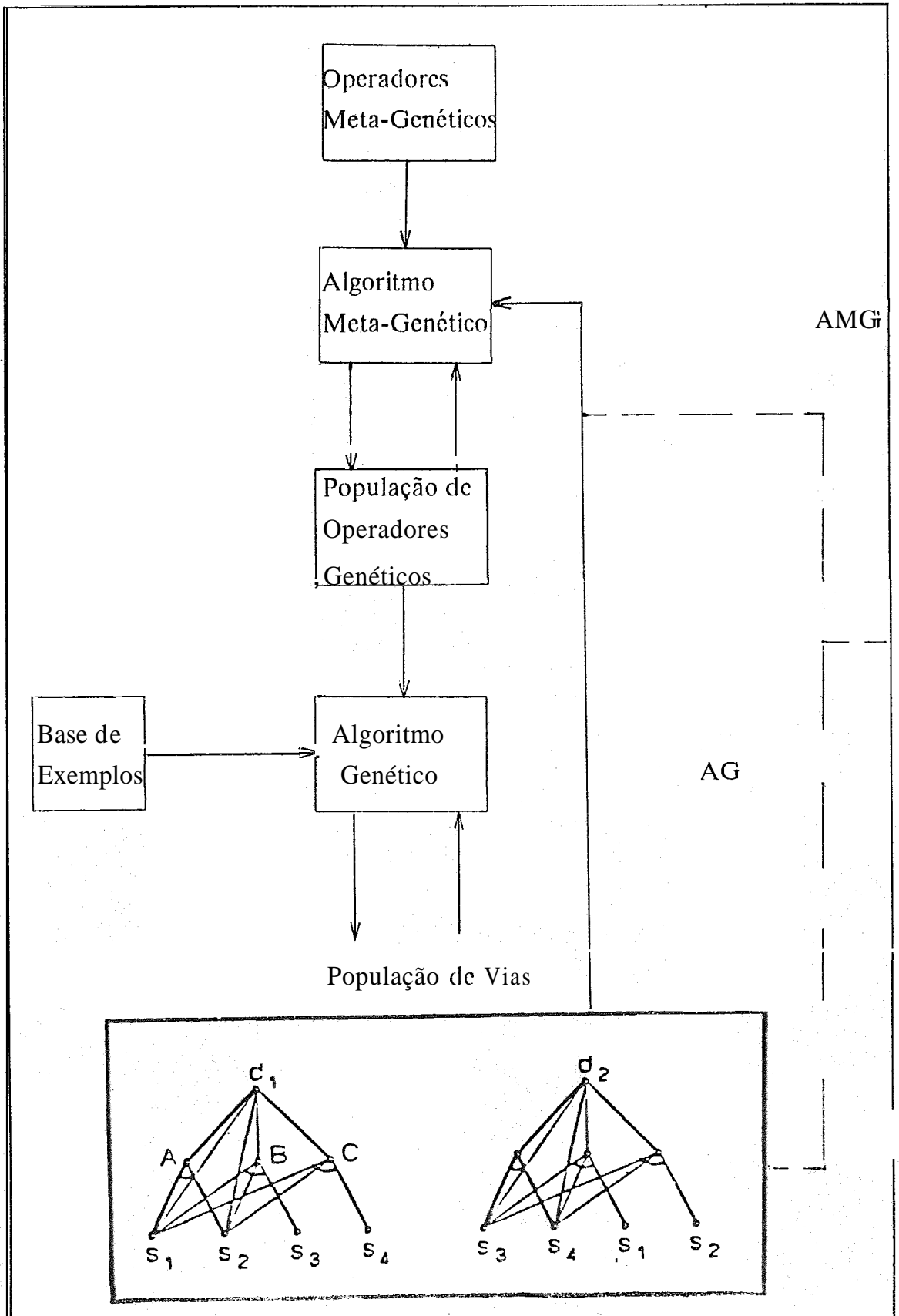


Figura 16. Arquitetura do MCE

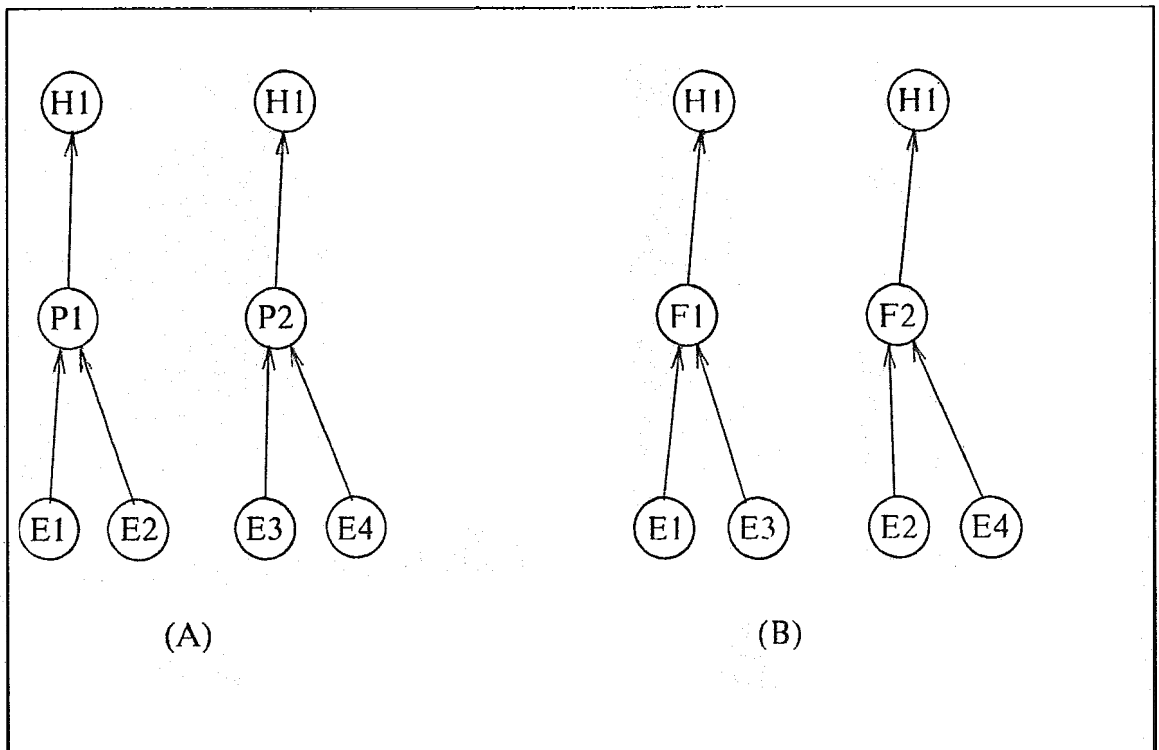


Figura 17. Exemplo de reprodução de vias

No MCE, como já dito, a população é o conjunto de vias existentes na RNA. Isto é possível devido a modularidade que as vias, também chamadas de aglomerados de informação, apresentam. A população global é a própria rede. Por outro lado, o desempenho do sistema não depende de um indivíduo mais apto, mas de uma sociedade de indivíduos que competem e convivem numa rede.

Esta proposta evolutiva permite ter uma RNA com uma topologia plástica, a qual é capaz de absorver regras novas, plausíveis ou **não**, sem perda das capacidades existentes em situações já experimentadas.

Este esquema difere radicalmente de outras implementações que combinam AGs com modelos conexionistas, nos quais cada elemento da população é uma RNA completa ([MONT89], [WHIT90]). Por exemplo, no sistema desenvolvido por Montana e Davis [MONT89], as redes competem entre si durante a execução do AG, sendo que as RNAs mais potentes são escolhidas

para reproduzir. A rede mais apta é escolhida ao final como solução. Sob este aspecto, o MCE se assemelha ao sistema classificador de Holland [HOLL75]. No MCE, a solução do problema é representada pela população inteira de vias.

A tabela 8, baseada em Denis [DENI89], compara o sistema classificador de Holland [HOLL75] com o MCE, e proporciona uma idéia mais clara de como os AGs atuam em cada um desses sistemas.

No MCE, o papel dos classificadores é exercido pelas vias. Na RNA do MNC, as vias podem ser vistas como um esquema num classificador de Holland [HOLL86]. Por exemplo: se um paciente apresenta o sintoma A e o sintoma B, então o paciente tem a doença D independentemente dos resultados dos demais exames.

O papel do algoritmo de "Bucket Brigade" é desempenhado no MNC pelo algoritmo de punição e recompensa. Enquanto o primeiro calcula a aptidão dos classificadores, o segundo calcula os pesos dos arcos, os quais são utilizados para calcular a aptidão das vias, que no MCE é representada pela suas potências.

Tabela 10. Comparação entre o Sistema Classificador de Holland e o MCE

SISTEMA CLASSIFICADOR	MODELO COSEXIOSISTA EVOLUTIVO
<p>*Sist. Classificador Propriamente Dito Classificador</p> <p>Condição/Ação</p> <p>Passagem de Mensagens</p> <p>*Sist. de Atribuição de Crédito</p> <p>Aptidão do Classificador</p> <p>*Algoritmo Genético</p> <p>Operador de Recombinação</p> <p>Atuação na Condição da Regra</p> <p>Classificadores de maior aptidão são elegíveis para reprodução</p> <p>Potencial inicial de filhos é função da aptidão do pais</p> <p>Recombinação aleatória</p> <p>Número de classificadores fixo</p> <p>Eliminação de classificadores fracos</p>	<p>Rede Neuronal</p> <p>Via (neurônio oculto) ligando um conjunto de evidências a uma hipótese</p> <p>Dendritos/Axônio</p> <p>Fluxos Evidenciais em um Arco</p> <p>*Algoritmos de Punição e Recompensa</p> <p>Potência da Via</p> <p>*Algoritmo Genético</p> <p>Operadores de Mutação e Recombinação</p> <p>Atuação nos Dendritos</p> <p>Vias de maior aptidão são elegíveis para reprodução</p> <p>Potencial inicial de filhos é estimada usando a base de exemplos</p> <p>Mutação e recombinação heurísticas e aleatórias, limitadas pelos exemplos</p> <p>Número de vias variável até um valor máximo</p> <p>Eliminação de vias fracas</p>

Nas seções V.2 e V.3 são explicados com mais detalhe o AG e o AMG da arquitetura do MCE. O AG implementado no MCE apresenta peculiaridades que o distingue dos AGs clássicos. Essas diferenças são discutidas na seção v.4.

V.2 - ATUAÇÃO DO ALGORITMO GENÉTICO NO MCE

Em grande parte, o AG proposto segue o padrão apresentado no capítulo IV. Em nosso sistema, o algoritmo atua sobre as vias da rede gerando novas vias, alterando a topologia da rede. Ele exige como pré-requisito para começar a atuar, uma rede que detenha um conhecimento inicial, seja por eliciação de conhecimento especialista, seja por um treinamento prévio.

Foram implementadas duas versões do MCE no sistema. Na primeira, chamada de GAREFINE, os filhotes (as vias) são incluídos na rede com sua aptidão real já estimada. Na segunda, chamada de GENALG, os filhotes gerados herdam um percentual das aptidões dos pais, e ao longo do tempo tem essas aptidões iniciais ajustadas para seus valores reais.

Uma análise das diferenças existentes entre o GAREFINE e o GENALG será apresentada na sub-seção V.2.2.

Ambas as versões utilizam uma rotina de preparação, chamada de GENPREP, que realiza as seguintes funções:

1. Cria uma estrutura auxiliar de dados para acelerar o processamento. Tal estrutura é composta das aptidões dos aglomerados de informação, dos códigos das evidências que estão ligadas a cada via e pelo endereço da sinapse superior que conecta cada via à hipótese;
2. Computa o novo tamanho da população de neurônios para esta iteração. No caso do GENALG, isto é feito mediante o uso de um procedimento de redução populacional que pode ser especificado pelo engenheiro de conhecimento indicando:
 - o tamanho inicial da população;
 - o tamanho final da população;
 - a iteração a partir da qual a redução populacional deve se processar;
 - a velocidade da queda do tamanho da população.

No caso do GAREFINE, o procedimento de redução populacional não é invocado pois o tamanho da população permanece igual àquele especificado quando da geração do modelo pelo engenheiro de conhecimento. Esta versão do MCE trabalha com uma população de tamanho pulsante entre dois extremos. Durante uma iteração, nesta população é acrescentada uma área de "buffer" para receber os filhotes produzidos. Ao final de uma iteração o tamanho da população é reduzido ao tamanho original pela eliminação das vias mais fracas;

3. Poda a rede eliminando as vias de menor aptidão até que o tamanho fique igual ao especificado pelo procedimento de redução de população;
4. Cálculo dos limiares de reprodução para as vias de cada hipótese. Estes limiares são definidos como sendo a média das aptidões dos aglomerados de informação existentes na rede para cada hipótese;
5. Identificação dos operadores genéticos válidos no arquivo de operadores genéticos;
6. Execução do algoritmo meta-genético se o usuário tiver requerido sua ativação.

O AG é invocado para cada exemplo de treinamento processado durante a iteração, forçando-o a realizar operações genéticas que venham a ajudar na solução do exemplo sob análise. Isto atua como uma heurística poderosa, direcionando a busca no espaço de aglomerados de informação.

O programa GAREFINE trabalha por iterações (gerações), as quais são compostas das seguintes fases:

1. Preparação da estrutura auxiliar (GENPREP);
2. Reprodução. O programa varre a base de exemplos, segundo uma seqüência aleatória, para fazer a reprodução dos aglomerados de informação. Os aglomerados gerados são adicionados à população usando o espaço disponível existente na rede e na área do "buffer";

3. Atribuição de crédito. Uma passagem na base de exemplos é executada com o MNC para calcular os pesos das sinapses, podar a rede e realizar o cálculo das potências (aptidões) dos aglomerados de informação;
4. Redução da população. O tamanho da população é reduzido àquele especificado pelo engenheiro de conhecimento através da eliminação das vias menos potentes (esvaziamento da área de "buffer").

De maneira mais detalhada, para cada via tanto o GAREFINE quanto o GENALG executam as seguintes funções:

1. Verifica se a via corrente é elegível para reprodução. Isto é feito comparando o fluxo evidencial potencial com o limiar de reprodução. Se o fluxo da via for superior então ela é elegível;
2. Calcula a probabilidade de reprodução da via e faz um sorteio. A probabilidade de reprodução da via é proporcional a diferença entre sua aptidão e a aptidão média da população (limiar de reprodução). Um fator de fertilidade das vias é utilizado nesse cálculo, permitindo modular a taxa de reprodução;
3. Caso a via seja sorteada, determina a operação genética a ser aplicada (também por sorteio), usando a distribuição de probabilidade de seleção associada aos operadores genéticos;
4. Se a operação for de recombinação procura um parceiro (outro aglomerado de informação dentro da mesma hipótese);
5. Aplica a operação genética gerando um filhote, ou dois se a operação for de recombinação;
6. Testa se os filhotes já existem na população. Caso já exista um aglomerado igual na mesma hipótese, o filhote é desprezado. Caso contrário, ele é incluído na rede.

O AG atua sobre redes já treinadas e portanto relativamente pequenas. A complexidade computacional é limitada em função da existência de um teto para a população de neurônios.

O mérito dos elementos da população é dado pela aptidão da via (o fluxo evidencial corrente injetado no neurônio hipótese quando na entrada dispomos de evidências plenas). O cálculo do mérito depende dos pesos, que são obtidos pelo balanço entre punições e recompensas nos acumuladores das sinapses da rede durante o processo de treinamento do MNC.

A seleção das vias para reprodução pode ser feita em função do fluxo evidencial corrente ou do fluxo evidencial potencial, comparando um dos dois com o limiar de reprodução. No primeiro caso, força-se uma aderência ao caso, ou seja, tenta-se fazer mutações que ajudem a solucionar o caso presente usando apenas evidências disponíveis.

Um teste é feito para verificar se os filhotes gerados são inéditos na subrede da hipótese em questão. Os pais permanecem na rede, sendo os filhotes acrescentados na rede substituindo os elementos mais fracos. No caso do GAREFINE, os filhotes são incluídos, inicialmente, na área de "buffer". Dessa forma, o sistema não perde as capacidades adquiridas anteriormente.

O sistema converge para vias de elevada aptidão devido à política do arrocho. A política do arrocho corresponde a ter um crescimento contínuo do limiar de reprodução exigindo valores de aptidão cada vez maiores para reprodução. Esta política é garantida pelo tamanho limitado da população, e pelo fato de que os aglomerados de informação já existentes só são substituídos por outros aglomerados com aptidão superior. No GAREFINE pode ocorrer a inclusão de um aglomerado com aptidão reduzida, caso a população esteja aquém da dimensão especificada pelo engenheiro de conhecimento, o que poderia a vir a diminuir a aptidão média da população.

O fator de fertilidade permite que seja modulada a velocidade da evolução, permitindo uma maior ou menor taxa de reprodução. As probabilidades de seleção de operadores genéticos permitem modular a taxa de aplicação de cada operador genético (eg. taxa de recombinação e taxas de diversos tipos de mutação).

As probabilidades de seleção de operadores genéticos são calculadas em função do seu mérito relativamente aos demais operadores. As operações genéticas são sorteadas, tendo maior chance aquelas que tem maior mérito. O mérito de um operador genético é medido pela aptidão média dos aglomerados gerados por ele no passado.

Os operadores genéticos são representados sob a forma de um vetor (cadeia de códigos) o que facilita a implementação de um algoritmo meta-genético. É apresentada a seguir uma descrição destes operadores.

V.2.1 - Operadores Genéticos

Os operadores genéticos possuem a seguinte estrutura:

- **Descrição do Operador Genético:** é um código onde estão codificadas as informações que especificam o operador genético. Estas informações são:
 - tipo de operador: na presente implementação foram criados operadores de mutação (sub-divididos em operadores de eliminação, substituição e adição), e operadores de recombinação simples;
 - ponto selecionado para execução da operação: no caso dos operadores de mutação por substituição ou de mutação por eliminação indica qual o critério de seleção da evidência, pertencente ao conjunto de evidências do pai, que será substituída ou eliminada, e no caso dos operadores de recombinação indica os pontos de recombinação, dentro dos conjuntos de evidências dos pais, onde serão feitas as quebras para formação dos dois novos conjuntos de evidências (filhotes);
 - critério de evidências externas ao aglomerado pai, por exemplo: aleatória, a de maior crença, etc;
- **Mérito do Operador Genético:** é o valor que mede a aptidão do operador genético referente a reprodução. Este valor é calculado pela aptidão média

dos aglomerados gerados por este operador, no passado assumindo um valor no intervalo $[0, 1]$;

- **Probabilidade de Reprodução do Operador Genético:** esta probabilidade possui um papel duplo. No seu primeiro papel representa a probabilidade de um operador poder gerar novas vias dentro da rede. Nesta função, apenas os operadores genéticos com um mérito acima da média da população são elegíveis para reprodução. Um segundo papel corresponde a probabilidade dele ser selecionado pelo AMG para ser usado na geração de novos operadores genéticos.

Os operadores genéticos de recombinação simples e mutação podem ter muitas variantes dependendo do emprego de processos aleatório ou heurístico para a geração dos filhotes.

O processo heurístico tenta realizar as operações aparentemente mais úteis no sentido de melhor resolver o exemplo de treinamento que está sendo analisado. Por exemplo: o operador de mutação por adição, quando aplicado a uma via, pode selecionar a evidência de maior crença externa à mesma, existente no exemplo, para ser acrescentada a esta via.

V.2.1.1 - Exemplos de Operadores Genéticos

São apresentados a seguir quatro esquemas ilustrativos que representam as atuações dos operadores genéticos propostos. Os três primeiros estão relacionados com o operador genético de mutação e o último com o operador genético de recombinação. As evidências (entradas) estão representadas pelas letras A, B, C, D, X, Y e Z, e a hipótese (saída) pela letra H.

Esquema 1: Operador de Mutação por Eliminação

Este operador gera um filhote através da eliminação de uma das evidências que estão conectadas ao pai. Por exemplo: seja o aglomerado pai *PI* formado

pelas evidências A, B e C, conectando à hipótese H conforme a figura 18.(A). A aplicação do operador de mutação por eliminação a *PI* gera um filhote *FI* que possui apenas as evidências A e R segundo a figura 18.(B).

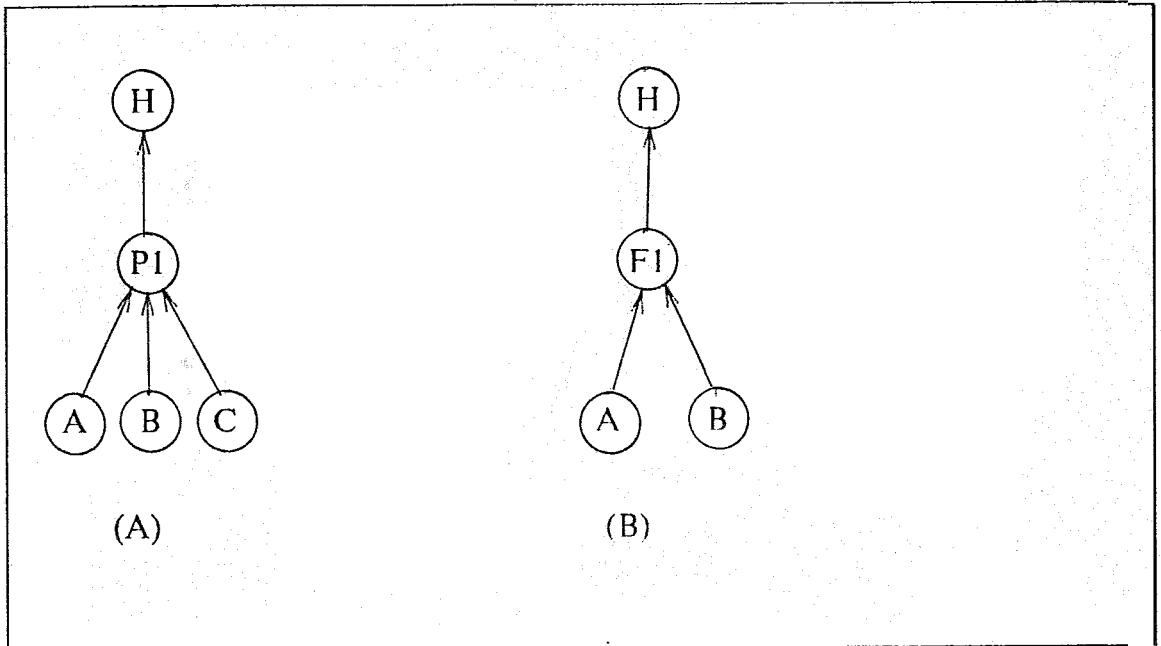


Figura 18. Operação de Mutação por Eliminação

Esquema 2: Operador de Mutação por Substituição

Este operador gera um filhote através da substituição de uma das evidências que estão conectadas ao pai. Por exemplo: seja a via formada pelo pai *PI* e pelas evidências A e B, conectado à hipótese H conforme a figura 19.(A). A operação de mutação por substituição aplicada a *PI* gera um filhote *FI* que possui as evidências A e D conforme está mostrado na figura 19.(B).

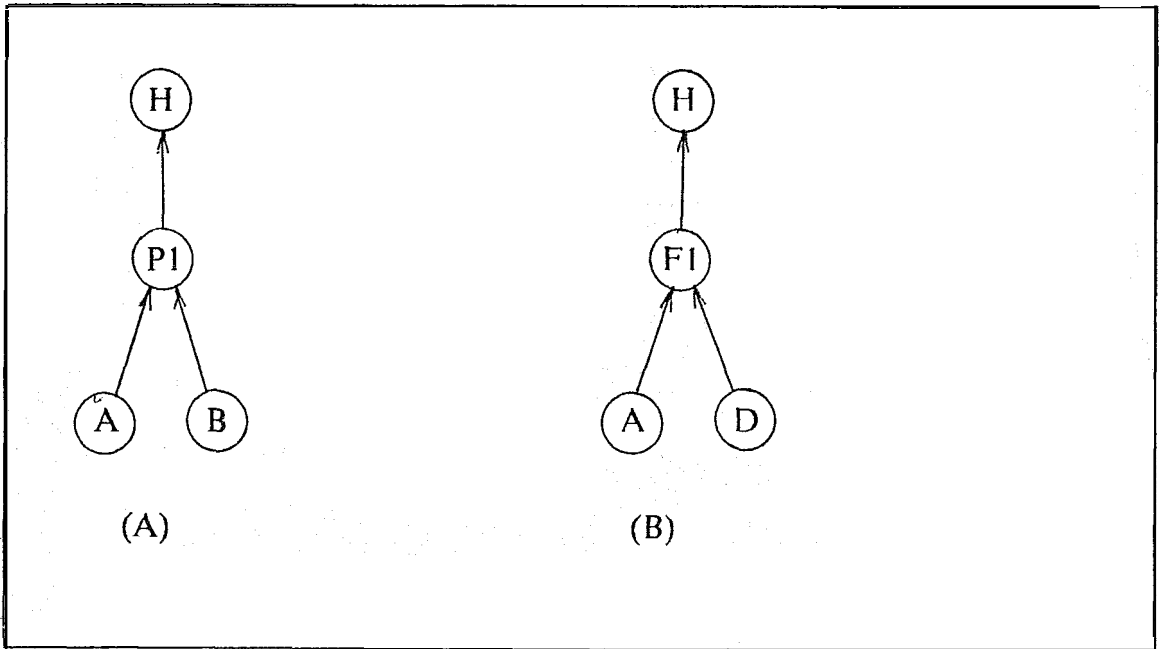


Figura 19. Operação de Mutação por Substituição

Esquema 3: Operador de Mutação por Adição

Este operador gera um filhote através da adição de uma evidência às já conectadas ao pai. Neste caso, seja o aglomerado pai *P1* formado pelas evidências A e B, conectado à hipótese H conforme a figura 20.(A). A aplicação do operador de mutação por adição a *P1* gera um filhote *F1* que possui as evidências A, B e C, segundo a figura 20.(B).

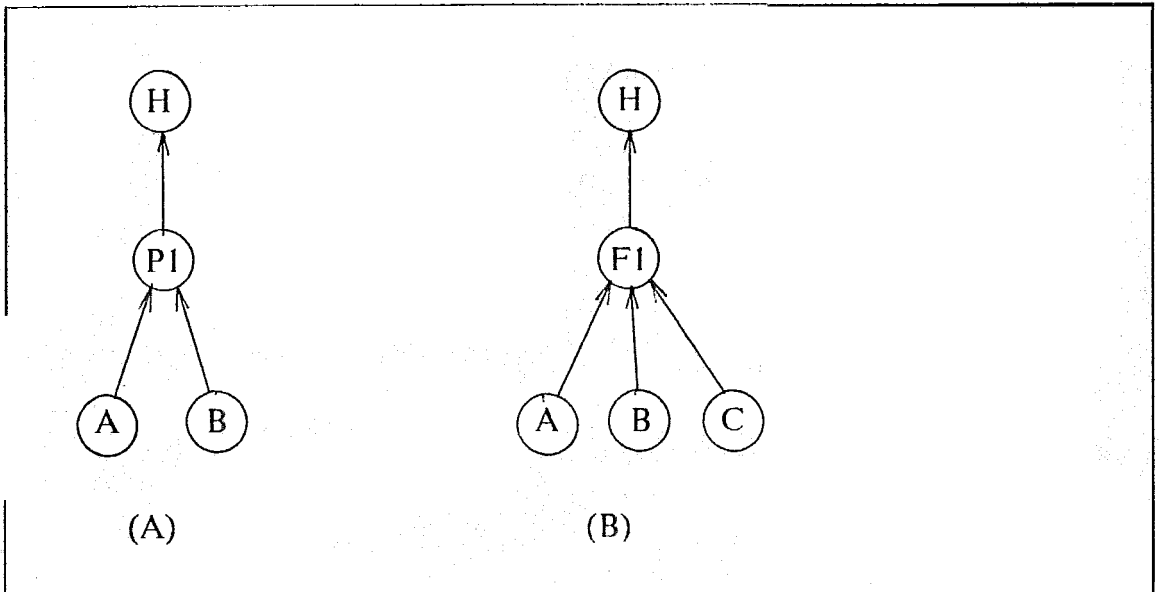


Figura 20. Operação de Mutação por Adição

Esquema 4: Operador de Recombinação

Este operador gera dois filhotes através da combinação das evidências de dois pais. Por exemplo: sejam os aglomerados de informação pai P1 e P2, com as evidências (A B C D) e (X Y Z) conectadas a cada um respectivamente. Os pais estão conectados à hipótese H conforme a figura 21.(A). Neste caso, o operador de recombinação seleciona aleatoriamente dois pontos de quebra, um para cada conjunto de evidências de cada via, e gera dois filhotes a partir destes pontos. No exemplo da figura 21.(A) estes pontos estão indicados por uma aspa (').

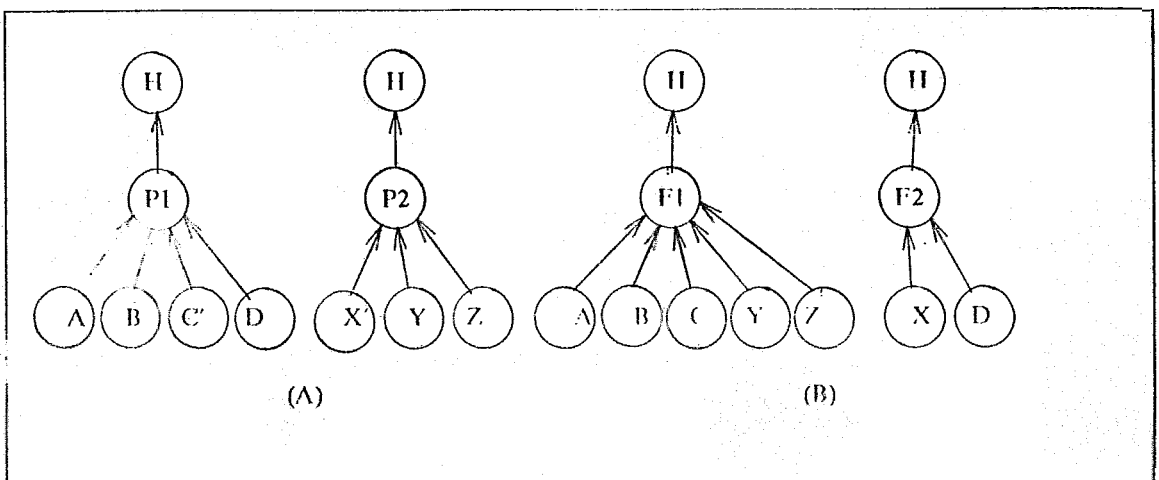


Figura 21. Operação de Recombinação

Desta forma, o conjunto de evidências do pai P1 foi segmentado em (A B C) e (D), e o do pai P2 foi dividido em (X) e (Y Z). O operador de recombinação combina as evidências dos pais gerando dois filhotes F1 e F2, estando ligadas as evidências (A B C Y Z) ao primeiro filhote e as evidências (X D) ao segundo, conforme apresentado na figura 21.(B).

No MCE, os operadores genéticos de mutação por adição e de mutação por substituição permitem a formação de aglomerados de informação com evidências que ainda não estavam sendo usadas pelo aglomerado. A atuação desses operadores genéticos permite a manutenção da diversidade genética da população das vias.

V.2.2 - Comparação entre as duas versões da MCE

Como já foi dito, o MCE possui duas versões, sendo que a principal é representada pelo programa GAREFINE, e a segunda pelo programa GENALG. No GAREFINE, a ordem aleatória dos exemplos utilizada na fase de reprodução visa a evitar que os primeiros exemplos da base venham a ser privilegiados com um número maior de seleções. Isto ocorreria no caso de um "overflow" na área de "buffer", sem que o programa tivesse passado por todos os exemplos.

Também já foi dito que os pesos são calculados na leitura de cada exemplo. Ao término de uma passagem na base de exemplos pode-se calcular a aptidão definitiva dos aglomerados existentes na rede. Ao final de uma iteração, a área adicional do "buffer" é descartada e, a população é reduzida àquele tamanho especificado pelo engenheiro de conhecimento. Esta redução é feita mediante a eliminação das vias mais fracas existentes na RNA.

O GENALG é a segunda versão do AG. No início da operação deste algoritmo, quando a população é ainda menor que o limite especificado pelo engenheiro de conhecimento, não se efetua a morte de aglomerados de informação, e a política do arrocho não vale.

Os filhotes entram na rede herdando uma parte da aptidão dos pais conforme uma taxa de herança. A aptidão inicial impede que os filhotes consigam estabelecer classificações durante a inferência. Com isso impede-se que a introdução de filhotes possa deteriorar o desempenho do sistema.

Nesta versão do AG, espera-se o seguinte comportamento dos filhotes introduzidos na rede. Caso representem mutações felizes, irão crescer sua aptidão através de um balanço positivo entre recompensas e punições ao longo do tempo. Ao cabo de algum prazo poderão influenciar decisivamente nos processos de inferência, e terão o direito de reproduzir. Caso representem mutações infelizes, verão sua aptidão diminuir através de um balanço negativo entre recompensas e punições ao longo do tempo. Chegará um momento que serão eliminados da rede, ou pelo processo de poda do MNC, ou pela substituição por filhotes mais potentes que eles.

V.3 - ATUAÇÃO DO ALGORITMO META-GENÉTICO NO MCE

O AMG segue, em grande parte, o padrão clássico de AG. Este algoritmo atua sobre a população de operadores genéticos e indica quais os melhores operadores a serem usados, e informando ao AG com que frequência os mesmos devem ser empregados, determinando a população ideal de operadores genéticos. Ele também é responsável pela reprodução dos operadores genéticos mais fortes. A seleção destes operadores é realizada em função da probabilidade de reprodução dos operadores genéticos, a qual depende diretamente do mérito dos mesmos.

O AMG é invocado antes do início de uma iteração. Quando ativado, ele executa as seguintes tarefas:

1. Computa o limiar de reprodução dos operadores genéticos. Este limiar é a média dos méritos dos operadores genéticos da população;
2. Reajusta o mérito dos operadores genéticos, garantindo um mérito mínimo para os operadores mais fracos;

3. Calcula as probabilidades de reprodução dos operadores genéticos;
4. Para cada operador genético efetua um sorteio com fins de reprodução;
5. Para cada operador genético sorteado para reprodução sorteia o operador meta-genético a ser aplicado (substituição ou recombinação);
6. Caso o operador meta-genético sorteado seja o de recombinação ele procura por um parceiro;
7. Realiza a operação meta-genética gerando filhotes, computando o mérito de nascimento dos mesmos. Este mérito dos novos operadores é calculado inicialmente como sendo a aptidão média dos operadores genéticos pais;
8. Verifica se os filhotes gerados são válidos e inéditos, desprezando-os caso não sejam;
9. Em caso positivo, e caso tenham um mérito de nascimento superior ao mérito dos elementos mais fracos da população, a substituição desses por aqueles é efetivada.

A população de operadores genéticos é de tamanho fixo. Portanto, vale aqui também a política do arrocho, que tende a propiciar um mérito médio crescente na população.

O mérito dos operadores genéticos é determinado computando-se a aptidão (potência) média dos aglomerados gerados pelos mesmos. Isto faz com que o tempo para estabilização desses méritos seja mais longo, pois é necessário que se forme uma população significativa de vias descendentes desses operadores.

O AMG dispõe de parâmetros como taxa de fertilidade e taxa de herança de mérito. Tais parâmetros influenciam diretamente no cálculo da probabilidade de geração de novos operadores genéticos. Esta tarefa de geração é efetuada pelos operadores meta-genéticos. São utilizados no AMG os seguintes operadores meta-genéticos:

- substituição - gera um novo operador genético a partir da substituição aleatória de uma posição no código do operador genético pai gerando um novo operador genético;
- recombinação - gera dois novos operadores genéticos a partir da combinação dos códigos de dois operadores pais, usando um ponto de recombinação escolhido aleatoriamente.

Por sua vez esta probabilidade é função da utilização, ou seja, ela está relacionada com os méritos dos operadores existentes na população de operadores.

Garante-se um mínimo de fertilidade para os operadores genéticos mais fracos, evitando-se as probabilidades de seleção nulas para os mesmos durante o processo de evolução da RNA, o que impediria a avaliação dos seus méritos.

V.4 - DIFERENÇAS ENTRE O MCE E O AG CLÁSSICO

Na tabela 9 são apresentadas as principais diferenças existentes entre os AG clássico e o que empregamos na construção do MCE.

Tabela 11. Diferenças entre o MCE e o AG Clássico

AG DO MCE	AG CLÁSSICO
<p>* Elementos Os elementos são conjuntos de características.</p> <p>Não é permitida a duplicidade de elementos.</p> <p>Os elementos são parte de uma solução global.</p> <p>* Esquemas Hiperplanos formados a partir de um alfabeto ternário. Cada elemento se assemelha a um esquema no sistema de Holland.</p> <p>* Operações As operações genéticas são limitadas pelo caso sob análise. São utilizadas heurísticas e conhecimento.</p> <p>Não existe a operação de reprodução e só os elementos mais fracos são substituídos de uma geração para outra.</p>	<p>* Elementos São cadeias de características determinadas pela posição que ocupam dentro das mesmas.</p> <p>É permitida a duplicidade de elementos.</p> <p>Cada elemento é uma solução candidata ao problema.</p> <p>* Esquemas Hiperplanos formados a partir de um alfabeto binário.</p> <p>* Operações As operações genéticas tem um caráter aleatório.</p> <p>Existe uma operação de reprodução e toda a população é substituída por uma nova a cada geração.</p>

No AG clássico, os esquemas de comprimento muito grande possuem um tempo de sobrevivência baixo, pois os operadores de recombinação tendem a romper estes esquemas. Segundo Goldberg [GOLD89], torna-se necessária a atuação de operadores de inversão e reordenamento para manter a sobrevivência destes esquemas. No MCE, a chance de se formar aglomerados é igual para todas as evidências (entradas), independente da posição dentro do vetor de entrada.

Interessa-nos manter no MCE uma variedade genética rica, visando a uma maior robustez do sistema classificatório. Por isso proibimos a duplicidade que tende a eliminar a variedade genética com a ocupação redundante do espaço disponível com registros redundantes.

Ao contrário dos AGs clássicos, onde as operações genéticas são essencialmente aleatórias, as operações genéticas do MCE são constrangidas a operarem dentro dos limites das informações de um exemplo. Isto funciona como uma heurística poderosa, no sentido de capturar o máximo de informação dos exemplos, restringindo o espaço de busca do AG. Além disso é possível incorporar nos operadores genéticos outras heurísticas e conhecimento do domínio do problema.

No MCE, a substituição dos elementos mais fracos por novos elementos mais fortes garante a política do arrocho, sem que ocorra a perda de capacidades já bem estabelecidas. Isto ocorre com mais segurança numa população de tamanho fixo, como no caso do MCE, do que no AG clássico.

Podemos visualizar as operações genéticas do MCE sob o ângulo do AG clássico se modelarmos os elementos do mesmo através de uma cadeia usando um alfabeto ternário $\{1, -1, *\}$ no MCE, onde:

- 1 - representa a existência de uma evidência (ligação excitatória);
- -1 - representa a existência da negação de uma evidência (ligação inibitória);
- * - representa a ausência de uma evidência no aglomerado.

Por exemplo, seja o aglomerado A formado pelo seguinte conjunto de evidências (a, b, -c, g). Poderia-se modelar este aglomerado em um esquema com o seguinte formato:

1 1 -1 * * * 1 * * * * * * * * * * * * * * * *

Apresentamos a seguir uma simulação das operações genéticas existentes no MCE, atuando sobre este alfabeto ternário, em um AG clássico hipotético.

V.4.1 - Simulando as Operações Genéticas do MCE no AG Clássico

A operação de mutação por adição do MCE é realizada através da mudança de um símbolo de ausência de evidência (*) para 1 ou -1. No caso do esquema anterior, se quisermos acrescentar uma evidência d obteríamos um novo esquema, representando o conjunto de evidências (a, b, -c, d, g), com o seguinte formato:

1 1 -1 1 * * 1 * * * * * * * * * * * * * * * *

A operação de mutação por eliminação seria realizada através da mudança de 1 ou -1 para um símbolo de ausência de evidência (*), sendo portanto equivalente a mutação do AG clássico. No caso do esquema anterior, se quisermos eliminar uma evidência g obteríamos um novo esquema, representando o conjunto de evidências (a, b, -c, d), com o seguinte formato:

1 1 -1 1 * * * * * * * * * * * * * * * *

A operação de mutação por substituição seria realizada através da mudança de uma evidência ativada com o 1 ou -1 para um símbolo de ausência de evidência (*), e a mudança de um outro símbolo de ausência de evidência (*) para 1 ou -1. Portanto ela é equivalente a realização de duas operações de mutação no AG clássico. Por exemplo, usando o esquema anterior, se quisermos substituir uma evidência d por uma evidência -e, obteríamos um novo esquema, representando o conjunto de evidências (a, b, -c, -e), com o seguinte formato:

1 1 -1 * -1 * * * * * * * * * * * * * * * *

A operação de recombinação do MCE poderia ser simulada da seguinte forma. Sejam os aglomerados A e B, representados pelos conjuntos de evidências (a, b, -c, d) e (-e, -f, g, h) respectivamente, conforme os esquemas abaixo:

A = 1 1 -1 1 * * * * * * * * * * * * * * * *

$$B = * * * * -1 -1 1 1 * * * * * * * * * * * * * * * *$$

Desejamos executar a operação de recombinação sobre os conjuntos de evidências obtendo os esquemas C e D, representados pelos conjuntos (a, b, g, h) e (-e, -f, -c, d) respectivamente, conforme os esquemas abaixo:

$$C = 1 1 * * * * 1 1 * * * * * * * * * * * * * * * *$$

$$D = * * -1 1 -1 -1 * * * * * * * * * * * * * * * *$$

Para se obter estes esquemas o AG clássico teria que executar uma operação de recombinação dupla com pontos de recombinação nas posições 2 e 6.

Uma idéia que surge nesta simulação seria a criação de um *operador de fusão* no MCE, que geraria um esquema a partir da fusão das evidências de outros esquemas. Isto se aproximaria da operação de recombinação simples do AG clássico. Não implementamos o operador genético de fusão no MCE pois ele favoreceria a formação de aglomerados (vias) muito grandes (com muitas evidências), as quais tenderiam a decorar os exemplos da base de exemplos. Os aglomerados pequenos, ao contrário, favorecem a capacidade desejada de generalização no sistema.

CAPÍTULO VI

DESCRIÇÃO DO AMBIENTE DE EXPERIMENTAÇÃO

Utilizou-se como ambiente de experimentação o sistema NEXT, desenvolvido por Machado et alli [MACH91]. A experimentação foi realizada nos problemas: OU-exclusivo, do João e Maria (ver capítulo 111), no problema 2-3 aglomerados (ver capítulo VII).

Neste capítulo, descrevemos o ambiente NEXT, ao qual acrescentamos o MCE, enfatizando sua arquitetura e seu método de construção de sistemas especialistas. Embora o NEXT seja uma ferramenta de uso geral, vamos descrevê-lo usando o jargão de sistemas de diagnóstico médico. Em seguida são apresentados os índices utilizados na medição do erro em sistemas classificatórios, categoria onde está inserido o nosso experimento.

VI.1 - DESCRIÇÃO DO NEXT

O NEXT é um ambiente para o desenvolvimento de sistemas especialistas conexionistas, dedicados a solução de problemas de classificação. Ele apresenta como sua característica mais marcante a capacidade de aprender com a experiência. O sistema pode receber um conhecimento inicial eliciado a partir de especialistas e refinar posteriormente este conhecimento com o uso. Pode também extrair todo seu conhecimento diretamente a partir de um conjunto de exemplos (casos com o diagnóstico correto conhecido).

O sistema foi desenvolvido e testado num computador IBM 3090 baseando-se nos softwares : VM-CMS e APL2.

VI.1.1 - A Arquitetura do NEXT

A ferramenta NEXT ("the Neural Expert Tool") é um ambiente para o desenvolvimento de sistemas especialistas conexionistas dedicados a solução de problemas de classificação, tais como: identificação, seleção, diagnóstico, reparo, monitoração [MACH90A]. Na figura 22 temos uma visão geral da arquitetura do NEXT.

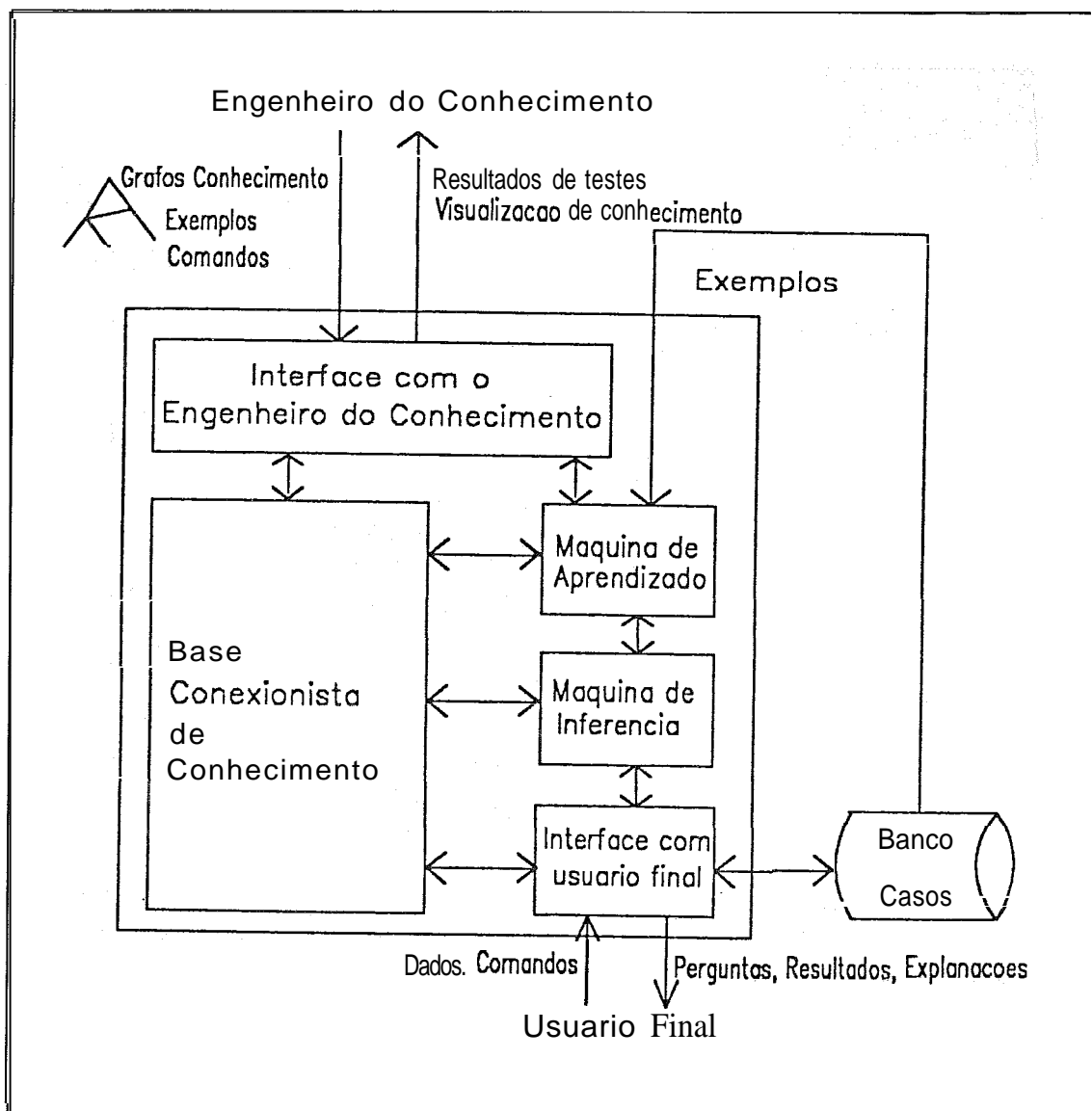


Figura 22. Arquitetura do KEXT

A relação dos componentes do NEXE é a seguinte:

1. Base Conexionista de Conhecimento
2. Máquina de Inferência
3. Máquina de Aprendizado
4. Interface com o Usuário Final
5. Interface com o Engenheiro do Conhecimento
6. Banco de Casos/Exemplos

7. Módulo de Aquisição de Conhecimento do Especialista

Base Conexionista de Conhecimento

A base conexionista de conhecimento é composta de uma rede semântica e de uma rede neuronal artificial (RNA). A rede semântica representa os conceitos e os objetos do domínio do problema (por exemplo: doenças, exames, sintomas, utilidades) e as relações entre os mesmos. Uma das principais relações é a relação "INFLUENCIA" que permite representar o diagrama de influência do domínio do problema.

A computação de incerteza relativa ao diagrama de influência é realizada através da RNA. Esta rede baseia-se no modelo neural combinatório (MNC) [MACH89] e apresenta uma topologia acíclica com 3 camadas:

- Camada de Entrada: cujos neurônios são ativados com as informações disponíveis sobre o paciente.
- Camada Intermediária: usando neurônios do tipo E-nebuloso que representam padrões de dados de entrada.
- Camada de Saída: usando neurônios do tipo OU-nebuloso que representam as doenças diagnosticáveis pelo sistema.

Máquina de Inferência

A máquina de inferência tem como objetivos básicos:

- Calcular o grau de possibilidade de cada hipótese e apresentar ao usuário aquelas que ultrapassem um limiar de aceitação. Isto é feito propagando as crenças da camada de entrada até a camada de saída;
- Determinar a questão Ótima a ser perguntada a seguir para o usuário [MACH90B];
- Explicar ao usuário o raciocínio empregado para atingir a solução de um problema.

Máquina de Aprendizado

A máquina de aprendizado é capaz de indutivamente selecionar e construir unidades internas nas camadas intermediárias da rede para representar regularidades importantes do domínio do problema.

Esta máquina de aprendizado baseia-se nos algoritmos de Punição e Recompensa (versões de arranque e refinamento) e Poda e Normalização do MNC citados no capítulo III (seção 111.2) e apresentados no apêndice 1 da tese. O MCE foi acrescentado a esse componente do NEXT para dar plasticidade a topologia da rede neuronal.

O Banco de Casos/Exemplos

Este banco tem como objetivo capturar a experiência adquirida pelo sistema ao resolver problemas e de fornecer exemplos para a máquina de aprendizado. Também permite a realização de consultas intermitentes.

VI.1.2 - Método de Construção de SEs no NEXT

Para se construir um sistema especialista usa-se a metodologia inerente ao NEXT, executando-se as seguintes fases:

Fase 1 - Especificação do Domínio do Problema

Nesta fase, a rede semântica é construída através de:

1. Especificação das Classes de Objetos chamadas de Objetos Intensionais
2. Especificação das Relações entre Objetos Intensionais
3. Instanciação das Classes de Objetos produzindo Objetos Extensionais
4. Especificação das Relações entre Objetos Extensionais

Fase 2 - Definição de uma Visão do Domínio do Problema

O NEXT permite criar e manter diferentes visões do domínio do problema. Uma visão especifica quais as partes da rede semântica que podem ser vistas, bem como as respectivas RNAs para executar o raciocínio aproximado.

Fase 3 - Aquisição do Conhecimento do Especialista

A aquisição de conhecimento de especialistas é realizada no NEXT através da eliciação de *grafos de conhecimento*. Os grafos de conhecimento são grafos acíclicos que descrevem as linhas de raciocínio usadas pelo especialista para inferir hipóteses a partir das evidências disponíveis ([LEAO90], [MACH90C] e [ROCH90]).

Fase 4 - Aquisição de um Conjunto de Exemplos

Nesta fase, exemplos (casos com a classificação correta) são coletados e introduzidos no sistema para refinar a base conexionista de conhecimento.

Fase 5 - Construção e Treinamento dos Modelos de Raciocínio

As tarefas cognitivas executadas por um especialista tais como diagnóstico, seleção de tratamento, prognóstico são representadas por classificadores, chamados no sistema de Modelos de Raciocínio. Cada modelo de raciocínio necessita de uma rede neuronal combinatória para computar o grau de possibilidade das hipóteses baseado nas evidências disponíveis e em suas crenças (ver seção III.2).

Os modelos de raciocínio podem receber o conhecimento de três maneiras distintas a saber:

1. Conversão direta dos grafos de conhecimento em RNAs;
2. Treinamento a partir do nada usando um conjunto de exemplos através do (uso da versão de arranque do algoritmo de aprendizado do MNC);

3. Refinamento do modelo de raciocínio (combinação das duas anteriores). O refinamento pode ser realizado usando-se:

- o algoritmo de refinamento do MNC; e
- o MCE, introduzido nesta tese.

Durante a fase de treinamento é possível enxugar-se a entrada dos modelos de raciocínio utilizando-se o procedimento de seleção de atributos existente no NEXT.

Fase 6 - Avaliação dos Modelos de Raciocínio

O NEXT oferece diversas técnicas para a avaliação dos modelos de raciocínio a saber:

- re-substituição: são usados para teste os mesmos exemplos usados durante o treinamento do sistema;
- "hold-out": é utilizado para teste um conjunto de exemplos diferentes daqueles usados para o treinamento;
- k-excluídos: separa-se k exemplos para teste e usa-se os restantes para treinamento. Isto é repetido separando-se um novo conjunto de k exemplos até se exaurir a base de exemplos. O desempenho é obtido pela média.

Calcula-se as taxas de erro de classificação usando-se diferentes medidas de grau de erro descritos na próxima seção.

Para modelos onde só é admissível uma única hipótese como resposta (exemplo: modelos de falha simples), o NEXT apresenta a tabela de contingência entre as classificações realizadas pelo modelo e as classes reais dos exemplos. Realiza também o teste de hipótese qui-quadrado de associação entre os resultados e as classes reais dos exemplos.

VI.2 - MEDIÇÃO DO ERRO EM SISTEMAS CLASSIFICATÓRIOS

Uma hipótese, em um caso sob análise, pode ocupar os seguintes estados em relação a um limiar de aceitação:

1. (A) - aceita;
2. (R) - rejeitada; e
3. (I) - indecida.

Em uma situação de múltipla falha, diversas situações podem ocorrer no confronto das respostas do sistema em relação a classificação real do caso. Por exemplo, utilizando a convenção acima, seja a seguinte cadeia de caracteres **ARARAR** correspondente a classificação real do caso, em um modelo com 6 hipóteses como saída. O sistema poderia devolver a seguinte cadeia **IARAR** como resposta da sua classificação. Portanto é necessário possuir índices que sejam capazes de medir este tipo de erro.

Os índices aqui definidos procuram expressar o erro do sistema ao fazer uma classificação. No caso, eles são usados para analisar o desempenho do sistema especialista construído mediante o uso da ferramenta NEXT. Estes indicadores são apresentados a seguir:

1. Grau de Erro R
2. Grau de Erro Pessimista
3. Grau de Erro Otimista
4. Grau de Erro de Crença Superior
5. Grau de Erro por Casamento
6. Grau de Erro por Casamento Expurgado
7. Grau de Erro Médio Quadrático

Grau de Erro R

Ele é computado adicionando-se o número de hipóteses erradamente aceitas, o número de hipóteses erradamente rejeitadas, a soma dos intervalos entre o limiar de aceitação e o grau de possibilidade mínimo de hipóteses indecidas, e a soma dos intervalos entre o grau de possibilidade máximo e o limiar de aceitação de hipóteses sem solução indecidas. O resultado é dividido pelo número de hipóteses da saída do modelo.

Grau de Erro Pessimista

Ele é calculado de maneira semelhante ao grau de erro R assumindo que todas as hipóteses indecidas tornar-se-ão incorretas.

Grau de Erro Otimista

Ele é calculado de maneira semelhante ao grau de erro R assumindo que todas as hipóteses indecidas tornar-se-ão corretas.

Grau de Erro de Crença Superior

Ele é calculado assumindo como hipóteses solução aquelas que apresentarem as maiores crenças. Para problemas de falha simples apenas uma hipótese é selecionada (a de crença máxima).

Grau de Erro por Casamento

Este indicador assume o valor 0 se existir um casamento perfeito entre a resposta do sistema e a situação real. Caso contrário ele assume o valor 1.

Grau de Erro por Casamento Expurgado

É semelhante ao indicador anterior. A diferença é que este índice é computado

para exemplos sem hipóteses indecidas.

Grau de Erro Médio Quadrático

Computa-se o erro médio quadrático produzido na saída do modelo.

CAPÍTULO VII

AVALIAÇÃO DO MODELO CONEXIONISTA EVOLUTIVO

Neste capítulo, reportamos os resultados dos experimentos realizados com o MCE. Inicialmente são descritas as RNAs construídas, segundo o Modelo Neural Combinatório (MNC) isoladamente, e combinado com o MCE, que foram aplicadas em problemas específicos constantes na literatura (OU-exclusivo, João e Maria, e 2-3 aglomerados). Em todos os ensaios descritos neste capítulo foi empregada a versão GAREFJNE do MCE, a qual é atrelada à versão de arranque do algoritmo de aprendizado do MNC.

Nos experimentos desse capítulo, o AG empregado fez uso dos seguintes operadores genéticos:

- Eliminação aleatória - gera um filhote (nova via) a partir da remoção aleatória de uma evidência do aglomerado (via) pai;
- Adição aleatória - gera um filhote acrescentando aleatoriamente uma evidência (constante no exemplo sob análise) ao conjunto de evidências do aglomerado pai;
- Substituição aleatória - gera um filhote a partir da substituição de uma evidência do pai, escolhida aleatoriamente por uma evidência do exemplo, selecionada aleatoriamente;
- Recombinação aleatória - gera dois filhotes combinando evidências de dois aglomerados pais, usando um ponto de recombinação escolhido aleatoriamente.

Os operadores genéticos acima mencionados são empregados com igual probabilidade de seleção. Em nenhum dos experimentos foi acionado o algoritmo meta-genético.

Neste capítulo, apresentamos, para cada experimento, tabelas e gráficos da evolução das taxas de erro apresentados ao longo do tempo (iterações). Foram empregadas as diversas taxas de erro computados pelo NEXT, conforme apresentado anteriormente na seção VI.2. Nas tabelas deste capítulo, serão usadas as seguintes abreviações:

- ITER. - número da iteração corrente;
- ERRO-R - taxa de erro R;
- PESSIM. - taxa de erro pessimista;
- OTIMIS. - taxa de erro otimista;
- CRENÇA SUPER. - taxa de erro de crença superior;
- CASAM. - taxa de erro por casamento;
- CASAM. EXPUR. - taxa de erro por casamento expurgado; e
- MÉDIO QUAD. - taxa de erro médio quadrático.

As taxas de erro mencionadas acima foram computadas como sendo a média dos graus de erro apresentados pelos exemplos do conjunto de teste. Nos gráficos faremos referência apenas aos graus de erro R, crença superior e médio quadrático, que acreditamos ser os mais significativos do conjunto de teste do sistema. Nos referenciamos aos mesmos pelos números 1, 2 e 3 respectivamente. Tal escolha permite uma melhor visualização dos gráficos. Também são feitas análises sobre as modificações experimentadas pelo conhecimento inicial de cada domínio.

VII.1 - O PROBLEMA DO OU-EXCLUSIVO

O problema do OU-exclusivo é o problema difícil clássico para as RNAs, sendo também considerado um problema difícil para os AGs. Como já foi dito, trata-se do menor problema não-linearmente separável. Partimos de uma rede não treinada de ordem 1, usando sinapses excitatórias e inibitórias.

Esta rede não resiste a uma sessão de treinamento com o MNC, que causaria o desaparecimento de todos os caminhos. O conjunto de exemplos, usado tanto para treinamento como para teste, é a tabela de verdade do OU-exclusivo. Rodamos o MCE com uma população máxima de 15 vias, e uma área de "buffer" de 10%.

A tabela 10 apresenta as taxas de erro computadas pelo NEXT ao longo de 5 iterações.

ITER.	ERRO-R	PESSIM.	OTIMIS.	CRENÇA SUPER.	CASAM.	CASAM. EXPUR.	MÉDIO QUAD.
1	0,5	0,5	0,5	0,5	0,5	1,0	0,5
2	0,125	0,125	0,125	0,0	0,25	0,25	0,125
3	0,125	0,125	0,125	0,0	0,25	0,25	0,125
4	0,125	0,125	0,125	0,0	0,25	0,25	0,125
5	0,0	0,0	0,0	0,0	0,0	0,0	0,0

Devido à natureza epistática (não-linear) do domínio de conhecimento deste problema, utilizamos um procedimento de busca cega, desativando temporariamente a função de avaliação por 4 iterações.

Conforme podemos observar a taxa de erros CRENÇA SUPER. cai a zero já na segunda iteração. As demais taxas de erro caem a zero na quinta iteração. O desempenho nesse ensaio revelou-se bastante superior a outros trabalhos

correlatos, por exemplo Austin [AUST90B] que atinge uma taxa de erros calculada pelo erro médio quadrático de 44% ao cabo de 34 iterações.

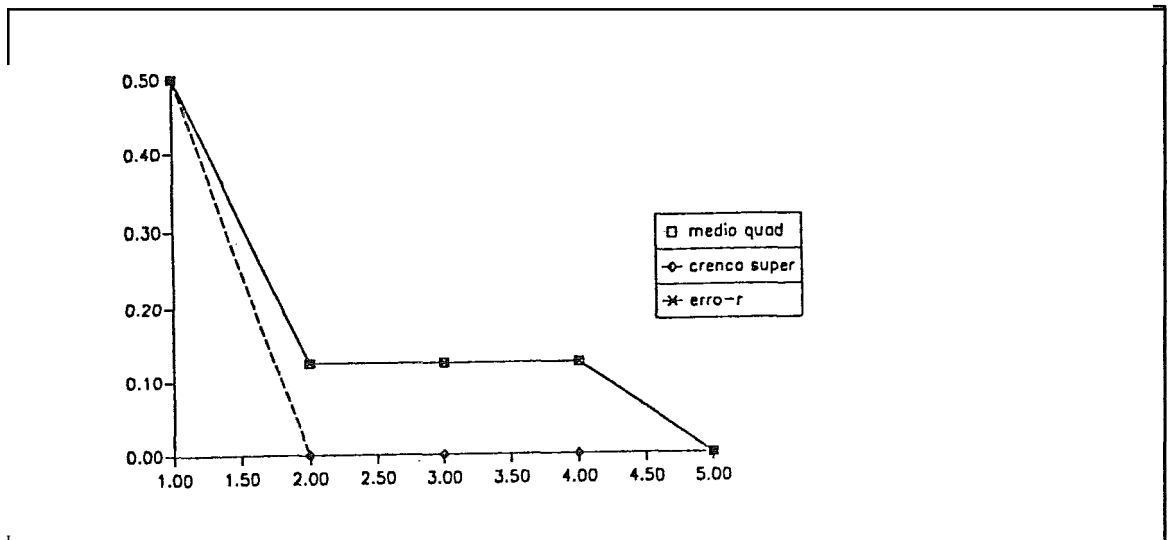


Figura 23. Gráfico de evolução das taxas de erro para o OU-exclusivo

Na figura 23, temos um gráfico de evolução das taxas de erro para o OU-exclusivo. Acreditamos que o sucesso rápido da busca cega empregada nesse ensaio deve-se ao fato de o espaço de busca 'de topologias no problema do OU-exclusivo ser muito pequeno.

VII.2 - O PROBLEMA DO JOÃO E MARIA

Relembrando o capítulo III, esse problema visa a discriminar dentre as doenças D1 e D2, em função das evidências E1, E2, E3 e E4. O conjunto de exemplos usados tanto para treinamento quanto para teste é (E1, E2, E3) e (E1, E2, E4) para D1, e (E1, E3, E4) e (E2, E3, E4) para D2. Entende-se a ausência de uma evidência num exemplo, como sua negação.

Com relação a este problema, realizamos 3 experimentos. Em todos eles usamos uma população máxima de 30 vias uma área de "buffer" corresponde a 10% dessa população. No primeiro, usamos como ponto de partida uma rede treinada de ordem 1. No segundo, temos uma rede treinada de ordem 3 com a

eliminação do gérmen de conhecimento. Por último, partimos de uma rede de ordem 1 com ligações invertidas. Em todos os experimentos só foi permitido o uso de sinapses excitatórias.

VII.2.1 - Rede Treinada com ordem 1

A rede treinada de ordem 1, usada como ponto de partida, apresenta a topologia representada na figura 24. Esta rede possui as taxas de erro médio quadrático, de crença superior e erro-R de 50%. pois o problema João e Maria é tipicamente um problema de ordem 2, quando utilizamos sinapses excitatórias.

Conforme apresentado no capítulo III temos que a solução deste problema é obtida com o aparecimento dos gérmenes de conhecimento: (E1-E2) para a doença D1 e (E3-E4) para a doença D2, com potência 1. Rodamos o MCE durante 15 iterações com a função de avaliação de aptidão ativada, estando os resultados obtidos apresentados na tabela II e na figura 25.

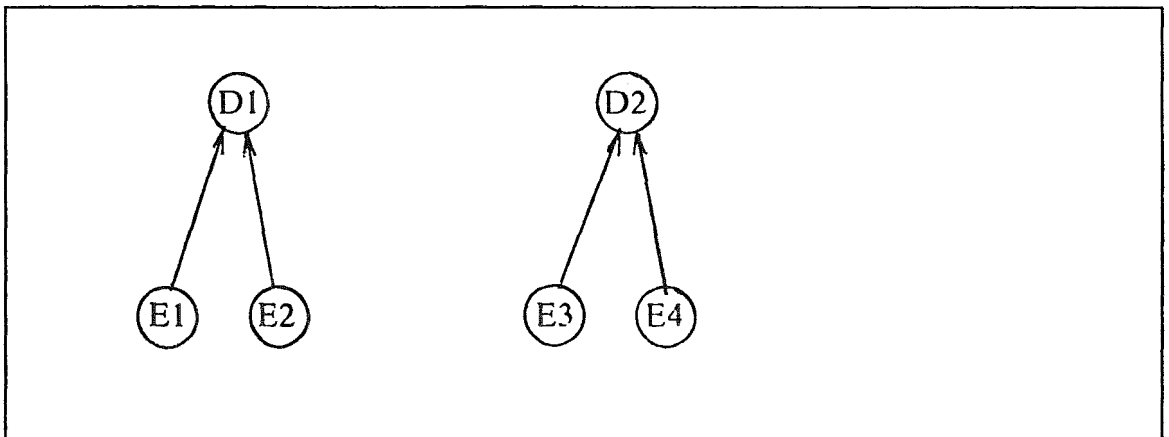


Figura 24. Rede treinada de ordem 1 para o problema do João e Maria

Tabela 13. Resumo dos resultados para o João e Maria ordem 1							
ITER.	ERRO-R	PESSIM.	OTIMIS.	CRENÇA SUPER.	CASAM.	CASAM. EXPUR.	MÉDIO QUAD.
1	0,5	0,5	0,5	0,5	1,0	0,5	0,5
3	0,0	0,0	0,0	0,0	0,0	0,0	0,0313
6	0,0	0,0	0,0	0,0	0,0	0,0	0,0313
9	0,0	0,0	0,0	0,0	0,0	0,0	0,0313
12	0,0	0,0	0,0	0,0	0,0	0,0	0,0235
15	0,0	0,0	0,0	0,0	0,0	0,0	0,0235

Conforme podemos observar já na terceira iteração todas as taxas de erro, com exceção do grau de erro médio quadrático descenderam a zero, em função da geração dos gérmenes de conhecimento requeridos por esse domínio de problema. A taxa de erro médio quadrático apresenta comportamento descendente ao longo das 15 iterações do experimento.

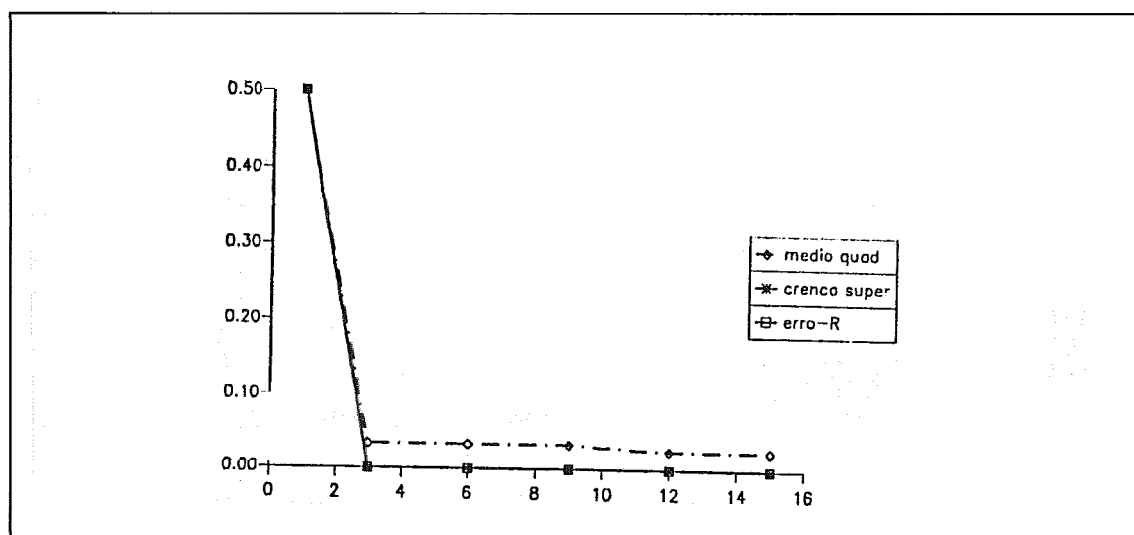


Figura 25. Gráfico da evolução das taxas de erro para o problema João e Maria ordem 1

VII.2.2 - Rede Treinada com ordem 3

Nesse experimento, partimos de uma rede treinada de ordem 3 (que resolve completamente o problema), da qual removemos os gérmenes de conhecimento (E1-E2) da rede de D1, e (E3-E4) da rede de D2, visando deliberadamente provocar uma degradação de desempenho.

A rede assim mutilada apresenta taxas de grau de erro-R, erro relativo à crença superior, e erro médio quadrático, iguais a 50%.

Rodamos o MCE durante 10 iterações, sendo as taxas de erro obtidas ao longo do experimento mostradas na tabela 12 e na figura 26 sob a forma de gráfico.

ITER.	ERRO-R	PESSIM.	OTIMIS.	CRENÇA SUPER.	CASAM.	CASAM. EXPUR.	MÉDIO QUAD.
1	0,5	0,5	0,5	0,5	1,0	1,0	0,96
2	0,5	0,5	0,5	0,5	1,0	1,0	0,5
4	0,25	0,25	0,25	0,25	0,5	0,5	0,5
6	0,25	0,25	0,25	0,25	0,5	0,5	0,2656
8	0,25	0,25	0,25	0,25	0,5	0,5	0,2656
10	0,0	0,0	0,0	0,0	0,0	0,0	0,0313

Conforme podemos observar, o algoritmo genético é capaz de regenerar os gérmenes de conhecimento extirpados, fazendo com que todas as taxas de erro, a exceção do erro médio quadrático, atinjam a marca de zero quando da décima iteração.

Na figura 26 temos um gráfico da evolução dos graus de erro para o João e Maria ordem 3 com a eliminação do germen de conhecimento.

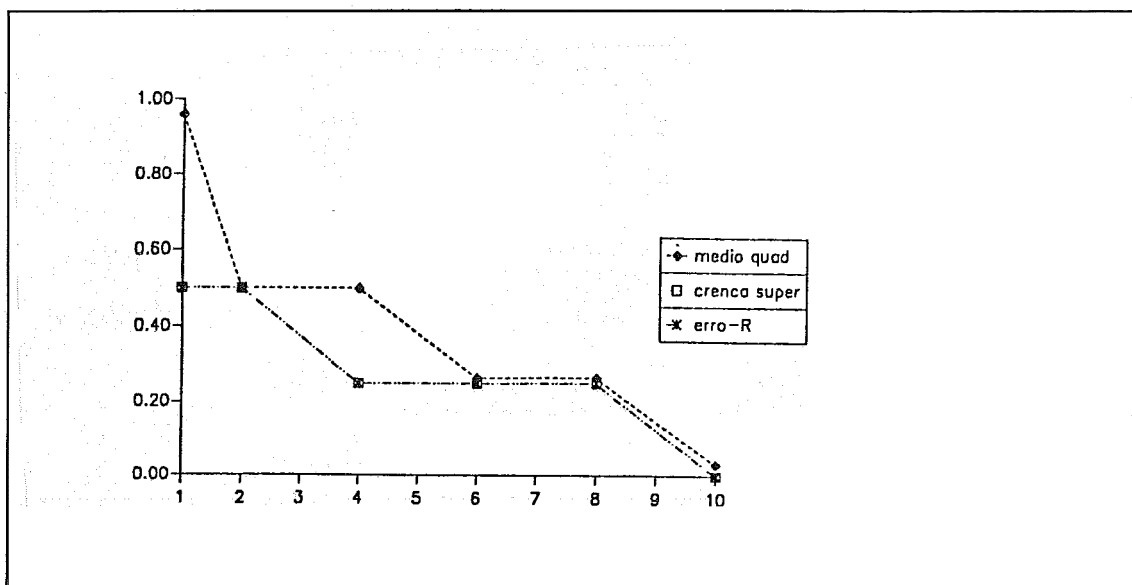


Figura 26. Gráfico da evolução das taxas de erro para o problema João e Maria. (ordem 3 com eliminação do gérmen de conhecimento).

VII.2.3 - Rede Treinada ordem 1 com ligações invertidas

Este experimento tem por objetivo mostrar que mesmo nas condições mais adversas a rede evolui para o padrão desejado. Para tanto criamos uma rede de ordem 1 com ligações invertidas mostrada na figura 27. Em princípio, esta rede não passaria pelo processo de treinamento do MNC, desaparecendo em função do número de punições muito maior que o número de recompensas.

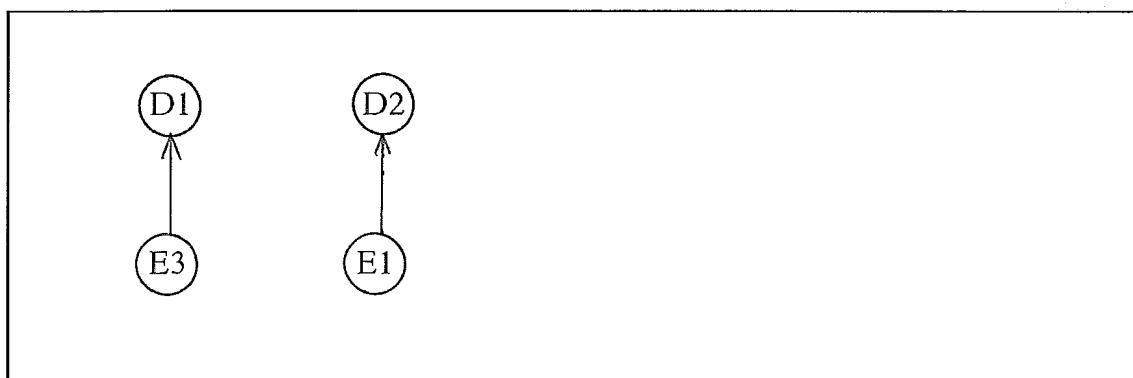


Figura 27. Rede inicial com ligações invertidas

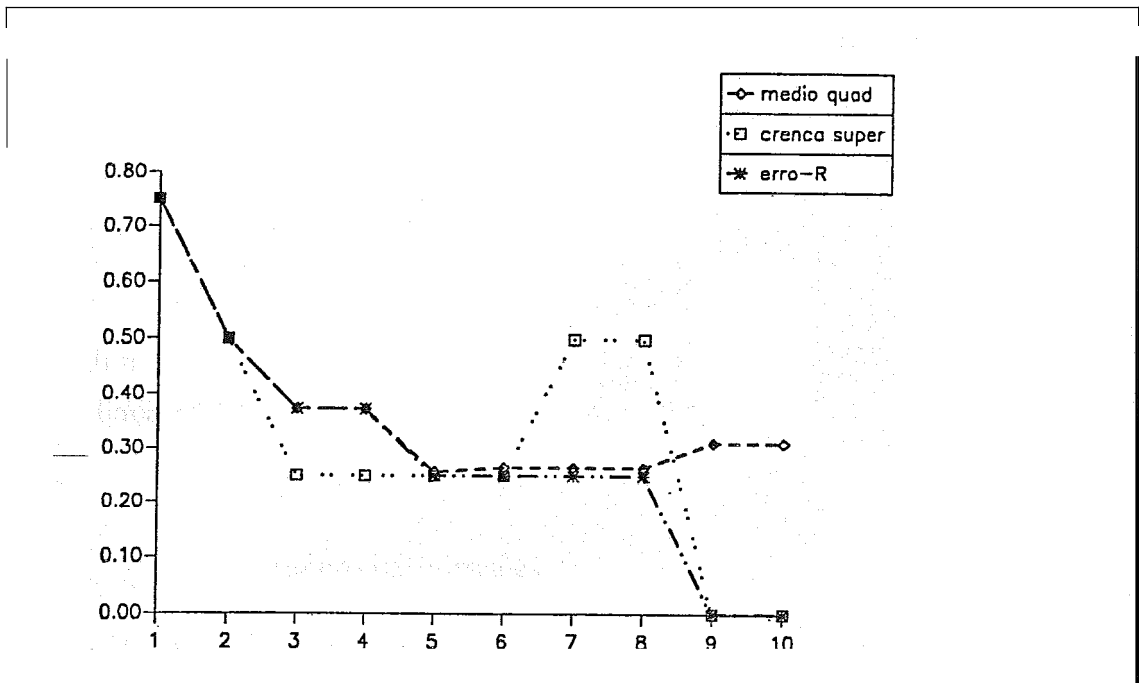


Figura 28. Gráfico da evolução das taxas de erro para o João e Maria. (ordem 1 com ligações invertidas).

VII.3 - O PROBLEMA DO 2-3 AGLOMERADOS

O problema do 2-3 aglomerados consiste em identificar numa cadeia de comprimento 10 formada de zeros e uns, quantas sub-cadeias 11 existem. Assume-se que as cadeias possam apresentar apenas 2 ou 3 sub-cadeias 11. Apresentamos a seguir duas cadeias que exemplificam este problema:

Padrão com 2 aglomerados: 0110011001;

Padrão com 3 aglomerados: 1100110110.

O conjunto usado para treinamento e teste apresenta 19 cadeias com 2 aglomerados e 4 cadeias com 3 aglomerados. O problema dos 2-3 aglomerados apresenta-se para o MNC como um problema de ordem 3, que não é resolvido integralmente pela versão de arranque do algoritmo de aprendizado. Apenas a

versão de refinamento do algoritmo de aprendizado do MNC consegue resolver na íntegra este problema.

O problema de 2-3 aglomerados exige o uso tanto de sinapses excitatórias como de inibitórias. Neste experimento, partimos de uma rede treinada de ordem 2, que apresenta taxas de erro-R igual a 34%, grau de erro relativo à crença superior igual a 10%, e taxa de erro médio quadrático igual a 30%.

A população máxima de vias foi estabelecida em 100 vias, reservando-se uma área de 10% para o "buffer". Rodamos o MCE durante 10 iterações a partir da rede inicial, sendo as taxas de erro obtidas ao longo dessas iterações mostradas na tabela 14, e na figura 29 sob a forma de gráfico.

Conforme pode se observar, obtivemos melhoras significativas de desempenho ao longo destas 10 iterações em todas as taxas de erro, a exceção da taxa de erro relativa a crença superior, onde não houve alteração. Nota-se que em função da versão GAREFINE do MCE estar atrelado à versão de arranque do MNC, não é de se esperar que as taxas de erro venham a descer até zero.

Tabela 16. Resumo dos resultados para o problema 2-3 aglomerados ordem 2.							
ITER.	ERRO-R	PESSIM.	OTIMIS.	CRENÇA SUPER.	CASAM.	CASAM. EXPUR.	MÉDIO QUAD.
1	0,3421	0,3421	0,3421	0,3684	0,6842	0,6842	0,3271
2	0,2895	0,2895	0,2895	0,1579	0,5789	0,5789	0,3158
3	0,2895	0,2895	0,2895	0,1579	0,5789	0,5789	0,2105
4	0,2895	0,2895	0,2895	0,1579	0,5789	0,5789	0,2105
5	0,2895	0,2895	0,2595	Q1579	0,5789	0,5789	0,2105
6	0,3158	0,3158	0,3158	0,1579	0,6316	0,6316	0,2632
7	0,3158	0,3158	0,3158	Q1579	0,6316	0,6316	0,2632
8	0,3158	0,3158	0,3158	0,1579	0,6316	0,6316	0,2632
9	0,0789	0,0789	0,0789	0,1053	0,1579	0,1579	0,2105
10	0,0526	0,0526	0,0526	0,1053	0,1053	Q1053	0,2105

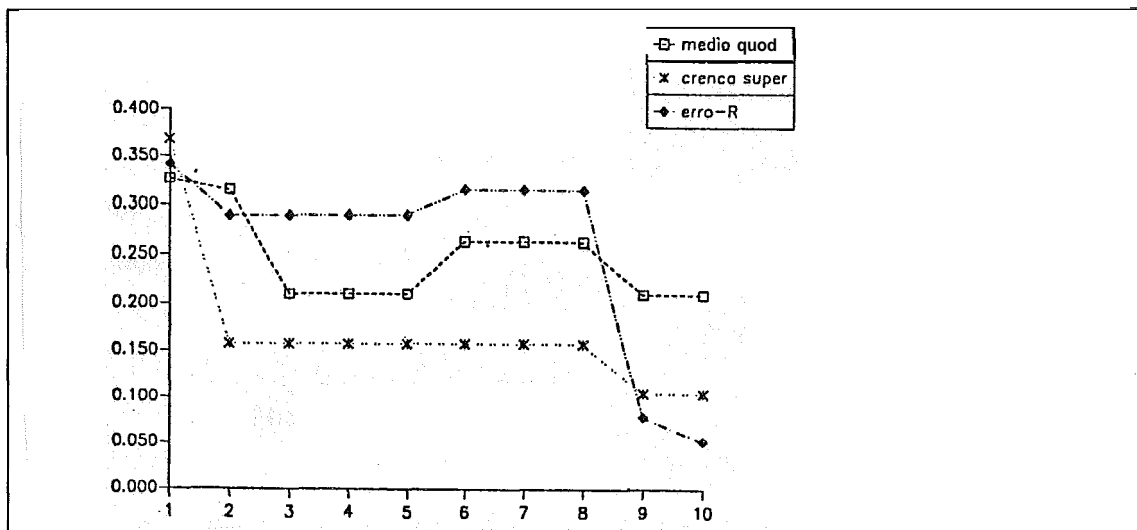


Figura 29. Gráfico da evolução das taxas de erro para o problema 2-3 aglomerados.

CAPÍTULO VIII

CONCLUSÕES

O aprendizado em sistemas especialistas (SEs) tem sido apontado como uma alternativa promissora para solucionar os seguintes problemas:

- *fragilidade dos SEs* - Os SEs tradicionais são considerados frágeis por responderem apropriadamente apenas em domínios restritos;
- *gargalo de aquisição de conhecimento* - A eliciação de conhecimento a partir de especialistas humanos é uma tarefa difícil e custosa. Além disso, os SEs tradicionais não possuem a capacidade de adquirir conhecimento automaticamente (aprendizado).

Como forma de superar estas limitações surge o conceito de Sistemas Inteligentes Híbridos [KAND91], considerados a próxima geração de SEs, que combinam técnicas de processamento simbólico com outras técnicas, tais como: lógica nebulosa, modelos conexionistas, algoritmos genéticos, redes bayesianas, etc.

Adotou-se nesta pesquisa o modelo neural combinatório (MNC), que consideramos uma ponte entre o simbolismo e o conexionismo. Achamos o MNC adequado e o adotamos neste trabalho em função das seguintes vantagens:

- Representação do conhecimento dos especialistas;
- Aprendizado heurístico a partir de exemplos;
- Refinamento incremental do conhecimento;

- Tratamento de informação incerta, vaga e imprecisa;
- Capacidade de explanação;
- Capacidade de aprendizado usando modelos conexionistas; e
- Redução da dimensão da entrada através da eliminação das evidências irrelevantes.

No entanto, o MNC exige uma topologia inicial adequada para a rede, pois ele só é capaz de ajustar os pesos sinápticos e eliminar as vias de raciocínio irrelevantes ou deletérias. Para domínios pequenos, podemos proporcionar essa topologia através de um modelo com ordem elevada. Em domínios grandes, o problema da explosão combinatória impede a geração de um modelo com ordem elevada.

Nesses casos, podemos tentar obter a topologia adequada eliciando o conhecimento de especialistas sob a forma de grafos de conhecimento, transformando-os em redes neuronais artificiais (RNAs). Entretanto, sempre existe o risco do conhecimento do especialista ser falho ou incompleto, ou de o mesmo não conseguir expressá-lo adequadamente nos grafos de conhecimento. Nesse caso, o MNC seria incapaz de criar as vias de raciocínio faltantes.

Como forma de superar esta limitação do MNC, propusemos o modelo conexionista evolutivo (MCE) que, baseado em técnicas de algoritmos genéticos (AGs), é capaz de realizar uma busca no espaço de topologias, visando a encontrar a configuração ideal para a RNA.

O MCE, ao contrário de outros trabalhos integrando AGs e RNAs, não se preocupa com o cálculo dos pesos da rede, que é deixado a cargo do MNC. Portanto existe uma distribuição mais equilibrada das tarefas entre o AG e a RNA.

O MCE também se distingue por considerar a rede como sendo a população a evoluir, com as vias representando os indivíduos, ao contrário dos demais

trabalhos, onde cada elemento da população é uma rede e portanto uma solução candidata.

A incorporação de um AG de segundo nível (AMG) destinado a controlar a evolução dos operadores genéticos, aplicáveis à RNA, pode aumentar consideravelmente a flexibilidade e o poder do MCE.

O MCE foi implementado com êxito em APL2, como uma extensão do ambiente de desenvolvimento de SEs conexionistas: NEXT.

Dentre as características do MCE destacamos:

- Capacidade de refinar um conhecimento anteriormente armazenado na RNA;
- A busca de uma topologia adequada para a RNA em relação a um dado domínio;
- Permite a multiplicidade de operadores genéticos;
- Incorporação de heurísticas gerais visando realizar operações genéticas mais efetivas;
- Possibilita a evolução dos operadores genéticos determinando a população ideal de operadores e a frequência com que devem ser usados através do algoritmo meta-genético (AMG); e
- Em problemas epistáticos, por exemplo XOR, é possível desativar temporariamente a função de aptidão para realizar uma busca cega.

Na experimentação realizada com diversos sistemas classificadores, foi possível constatar que o MCE é efetivo, mostrando-se capaz de encontrar a topologia adequada para a RNA e produzindo melhorias significativas do desempenho classificatório.

Como perspectivas de pesquisas futuras sugerimos:

- Continuação da investigação da versão GAREFINE do MCE, testando alternativas de diferentes tamanhos de população, populações estanques por hipótese, diferentes tamanhos de "buffer", conjuntos diversos de operadores genéticos, novos operadores (por exemplo: fusão de vias), diferentes probabilidades de seleção de algoritmos genéticos, confrontando seu desempenho com a versão clássica do algoritmo genético;
- Acionamento automático de procedimentos de busca aleatória quando se constata a inexistência de subredes para uma ou mais hipóteses, ou quando nenhuma via da subrede correspondente à hipótese-solução de um exemplo, é ativada. Neste caso vias aleatórias serão geradas automaticamente dentro dos limites do exemplo em questão visando a tentar resolvê-lo. Esta restrição atua como uma heurística de limitação da busca no espaço de topologias;
- Construção de uma versão atrelada ao algoritmo de refinamento do MNC, capaz de superar a limitação do algoritmo de arranque do MNC, que em certos domínios não consegue atingir o desempenho ótimo;
- Explorar o espaço de busca apresentado pelo AMG para determinar os melhores operadores genéticos existentes na população de operadores. O AMG é potencialmente um componente importante para ajustar os parâmetros do MCE;
- Teste e utilização do MCE em outros domínios de aplicação práticos como, por exemplo: processamento de imagens, reconhecimento de padrões, processamento de sinais, processamento de fala, diagnóstico de equipamento, predição, etc;
- Ampliação da atuação do AG permitindo que os operadores genéticos de adição e substituição tenham acesso a novas informações fora do escopo do modelo (deteção e incorporação automática de novidade); e
- Criação de operadores genéticos com acesso a bases de conhecimento teóricas sobre o domínio, capazes de realizar mutações mais efetivas.

REFERÊNCIAS BIBLIOGRÁFICAS

- [ANTO87] Antonisse, H.J., and Keller, K.S., "Genetic Operators for High-Level Knowledge Representations", The MITRE Corporation, McLean, VA, 1987.
- [AUST90A] Austin, S., "An Introduction to Genetic Algorithms", AI Expert, pgs. 48-53, 1990.
- [AUST90B] Austin, S., "Genetic Solutions to XOR Problems", AI Expert, pgs. 52-57, 1990.
- [BARR81] Barr, A. and Feigenbaum, E., "The Handbook of Artificial Intelligence", vol. I, William Kaufmann Ed., Los Altos, CA, 1981.
- [BOOK87] Booker, L.B., "Improving Search in Genetic Algorithms", Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, Los Altos, CA, pgs. 61-73, 1987.
- [BOOK88] Booker, L.B., "Classifier Systems that Learn Internal World Models", Machine Learning 3, pgs. 161-192, 1988.
- [BOOK89] Booker, L.B., Goldberg, D.E. and Holland, J.H., "Classifier Systems and Genetic Algorithms", Artificial Intelligence, 40, pgs. 235-282, 1989.

- [CARB83] Carbonell, J.G., Michalski, R.S., Mitchel, T.M., "Machine Learning Part I: A Historical and Methodological Analysis", Technical Report, Carnegie-Mellon University, CMS-CS-83-135, 1983.
- [CARB89] Carbonell, "Introduction: Paradigms for Machine Learning", Artificial Intelligence, vol. 40, pgs. 1-9, 1989.
- [CARV88] Carvalho, L.A.V., "Redes Neurais e a Tradição Conexionista da Inteligência Artificial", Relatório Interno, Programa de Eng. Mecânica, COPPE/UFRJ, 1988.
- [CARV89] Carvalho, L.A.V., "Síntese de Redes Neurais com Aplicações à Representação do Conhecimento e à Otimização", Tese de Doutorado, COPPE - Sistemas, UFRJ, 1987.
- [CHAN88] Chandrasekaran, B. and Goel, A., "From Numbers to Symbols to Knowledge Structures: Artificial Intelligence Perspectives on the Classification Task", IEEE Transactions on Systems, Man and Cybernetics, vol. 18, num. 3, pgs. 415-424, 1988.
- [COHE83] Cohen, M. and Grossberg, S., "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks", IEEE Transactions on Systems, Man, and Cybernetics, SMC-13, pgs. 815-825, 1983.
- [COTT85] Cottrell, G.W., "A Connectionist Approach to Word Sense Disambiguation", Ph.D. Thesis, University of Rochester, Computer Science, 1985.
- [DAVI87] Davis, L.D. and Steenstrup, M., "Genetic Algorithms and Simulated Annealing: An Overview", Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, Los Altos CA, pgs. 1-11, 1987.

- [DENI89] Denis, F.A.R.M., Machado, R.J., "Introdução aos Sistemas Classificadores e Algoritmos Genéticos", II Jornada de Atualização do Grupo de Inteligência Artificial, UFRJ, 1989.
- [FEIG82] Feigenbaum, E.A., "The Handbook of Artificial Intelligence", vol. 3, William Kaufmann Ed., Los Altos, CA, 1982.
- [GALL88] Gallant, S. I., "Connectionist Expert Systems", Communications of the ACM, February 1988, vol. 31 num. 2, pgs. 152-168.
- [GOLD89] Goldberg, D.E., "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley Publishing Company Inc., 1989.
- [GOTT90] Gottgroy, M.P.B., "O Processo de Aquisição do Conhecimento na Construção de Sistemas Especialistas", Tese de Mestrado - COPPE Sistemas, UFRJ, 1990.
- [GREF86] Grefenstette, J.J., "Optimization of Control Parameters for Genetic Algorithms", IEEE Transactions on Systems, Man, and Cybernetics, SMC-16, vol. 1, pgs. 122-128, 1986.
- [GREF87] Grefenstette, J.J., "Incorporating Problem Specific Knowledge into Genetic Algorithms", Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, Los Altos, CA, pgs. 42-60, 1987.
- [GREF88] Grefenstette, J.J., "Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms", Machine Learning 3, pgs. 225-245, 1988.
- [GROS86] Grossberg, S., "The Adaptive Brain I: Cognition, Learning, Reinforcement, and Rhythm, and The Adaptive Brain II: Vision, Speech, Language, and Motor Control", Elsevier/North-Holland, Amsterdam, 1986.
- [HAYE83] Hayes-Roth, F., Waterman, D.A., Lenat, D.B., "Building Expert Systems", Addison-Wesley Publishing, 1983.

- [HEBB49] Hebb, D.O., "The Organization of Behavior; A Neuropsychological Theory", Science Editions Inc., 1949.
- [HECH88] Hecht-Nielsen, R., "Applications of Counterpropagation Networks", Neural Networks, vol. 1, pgs. 131-140, 1988.
- [HINT89] Hinton, G.E., "Connectionist Learning Procedures", Artificial Intelligence, vol.40, pgs.185-234, 1989.
- [HOLL75] Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, MI, 1975.
- [HOLL78] Holland, J.H., Reitman, J.S., "Cognitive Systems Based on Adaptive Algorithms", Pattern-Directed Inference Systems, Academic Press, NY, pgs. 313-329, 1978.
- [HOLL86A] Holland, J.H., "Escaping Brittleness: the Possibilities of General Purpose Machine Learning Algorithms Applied to Paralell Rule-Based Systems, Machine Learning: An Artificial Intelligence Approach", R.S. Michalski, J.G. Carbonell e T.M. Mitchell Eds., vol.2, pgs. 593-623, Los Altos, California, Kaufmann, 1986.
- [HOLL86B] Holland, J.H., Holyoak, K.J., Nisbett, R.E. and Thagard, P.R., "Induction: Processes of Inference, Learning and Discovery", MIT Press, Cambridge, MA, 1986.
- [HOLL87] Holland, J.H., Holyoak, K.J., Nisbett, R.E. and Thagard, P.R., "Classifier Systems, Q-Morphisms and Induction", Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, Los Altos, CA, pgs. 116-129, 1987.
- [HOPF82] Hopfield, J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proc. Nat. Acad. Science USA, vol. 79, pgs. 2554-2558, 1982.

- [JACK86] Jackson, P., "Introduction to Expert Systems", Addison-Wesley Publishing, 1986.
- [KAND91] Kandel, A. and Langholz, G., "Intelligent Hybrid Systems", Department of Computer Science of Florida State University, Tallahassee, outline do livro, 1991.
- [KIRK83] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P., "Optimization by Simulated Annealing", Science, 220, pgs. 671-680, 1983.
- [KOH082] Kohonen, T., "Clustering Taxonomy and Topological Maps of Patterns", Proc. of the Sixth International Conference on Pattern Recognition", pgs. 114-128, 1982.
- [KOH084] Kohonen, T., "Self-Organization and Associative Memory", Springer-Verlag, Berlin, 1984.
- [KOH088] Kohonen, T., "The Neural Fonetec Typewriter", IEEE Computer, vol. 21, num. 3, pgs. 11-22, 1988.
- [KOSK88] Kosko, B., "Feedback Stability and Unsupervised Learning", Proceedings of the IEEE International Conference on Neural Networks: vol. 1, pgs. 141-152, San Diego, 1988.
- [LAIR86] Lair, J.E., Rosenbloom, P.S., Newell, A., "Chunking in SOAR: The Anatomy of a General Learning Mechanism", Machine Learning I, pgs. 11-46, 1986.
- [LEAO88] Leão, B.L., "Construção da Base de Conhecimento de um Sistema Especialista de Apoio ao Diagnóstico de Cardiopatias Congênitas", Tese de Doutorado - Escola Paulista de Medicina, 1988.
- [LEAO90] Leão, B.L., Rocha, A.F., "Proposed Methodology for Knowledge Acquisition: A Study on Congenital Heart Diseases Diagnosis", Methods of Inf. in Medicine, vol. 29, pgs. 30-40, 1990.

- [LIPP87] Lippmann, R.P., "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pgs. 4-22, 1987.
- [MACH89] Machado, R.J. and Rocha, A.F., "Redes Neurais Combinatórias - Um Modelo Conexcionista para Sistemas Baseados em Conhecimento", Anais do VI Simpósio Brasileiro de Inteligência Artificial, pgs. 344-358, 1989.
- [MACH90A] Machado, R.J., Rocha, A.F., Laginha, M.O.R., Rizzo, I., "Inference and Inquiry in Fuzzy Connectionist Systems", COGNITIVA90, Madrid, 1990.
- [MACH90B] Machado, R.J., Rocha, A.F., Leão, B.F., "Calculating the Mean Knowledge Representation from Multiple Experts", Multiperson Decision Making Models Using Fuzzy Sets and Possibility Theory, Eds.: Fedrezzi, M., and Kacprzyk, J., Kluwer Academic Publishers, 1990.
- [MACH90C] Machado, R.J., Duarte, V.H.A., Denis, F.A.R.M. e Rocha, A.F., "NEPHREX - Um Sistema Especialista Conexcionista para o Diagnóstico de Doenças Renais", anais do XII Congresso Brasileiro de Engenharia Biomédica, Botucatu, 1990.
- [MACH91] Machado, R.J., Duarte, V.H.A., Denis, F.A.R.M. e Rocha, A.F., "NEXT - The Neural EXpert Tool", Rio Scientific Center, Technical Report CCR-120, 1991.
- [MICH87] Michalsky, R.S., "Learning Strategies and Automated Knowledge Acquisition: An Overview", Computational Models of Learning, Leonard Bolc Ed., Springer-Verlag, pgs. 1-20, 1987.
- [MILL56] Miller, G.A., "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information", Psychological Review, 63, pgs. 81-97, 1956.

- [MINS69] Minsky, M. and Papert, S., "Perceptrons - An Introduction to Computacional Geometry", The MIT Press, 1969.
- [MINS75] Minsky, M., "A Framework for representing knowledge", Winston Ed., 1975.
- [MINS88] Minsky, M. and Papert, S., "Perceptrons - An Introduction to Computacional Geometry - Expanded Edition", The MIT Press, 1988.
- [MONA88] Monat, A.S., "Métodos de Raciocínio Impreciso para Sistemas Especialistas Baseados em Regras", Tese de Mestrado - COPPE Sistemas, UFRJ, 1988.
- [MONT89] Montana, D.J. and Davis, L., "Training Feedforward Neural Networks Using Genetic Algorithms", Proceedings of International Joint Conference of Artificial Intelligence (IJCAI-89), pgs. 762-767, 1989.
- [MUHL90] Muhlenbein, H., "Limitations of Multi-layer Perceptron Networks - Steps Towards Genetic Neural Networks", Parallel Computing 14, pgs. 249-260, North-Holland, 1990.
- [NEWE63] Newell, A. and Simon, H.A., "GPS, a Program that Simulates Human Thought", Computers and Thought, New York, McGraw-Hill, Feigenbaum and Feldman Eds., pgs. 279-293, 1963.
- [PAO89] Pao, Y.H., "Adaptive Pattern Recognition and Neural Networks", Addison-Wesley Publishing Company, 1989.
- [PESS90] Pao, Y.H., "Aprendizado Não-Supervisionado em Redes Neurais", Tese de Mestrado - COPPE Sistemas, UFRJ, 1990.
- [PINH88] Pinho, A.A., "Inteligência Artificial - Notas de Aula", COPPE Sistemas, 1988.

- [QUIL68] Quillian, M.R., "Semantic Memory", *Semantic Information Processing*, Cambridge Mass., MIT Press, Minsky Ed., pgs. 227-270, 1968.
- [ROCH90] Rocha, A.F., Laginha, M.P.R., Machado, R.J., Sigulem, D., Ancao, M., "Declarative and Procedural Knowledge - Two Complementary Tools for Expertise", *Approximate Reasoning Tools for Artificial Intelligence*, Eds.: Verdegay, J.L. and Delgado, M., Verlag Tuv-Interdisciplinary Systems Research Series, Rheinland, 1990.
- [ROMA89] Romaniuk, S.G. and Hall, L.O., "FUZZNET, A Fuzzy Connectionist Expert System", Technical Report CSE-89-07, Dept. of Computer Science and Engineering, University of South Florida, FL, 1989.
- [ROSE59] Rosenblatt, R., "Principles of Neurodynamics", New York, Spartan Books, 1959.
- [RUME85] Rumelhart, D.E. and Zipser, D., "Feature discovery by competitive learning", *Cognitive Science*, 9, pgs. 75-112, 1985.
- [RUME86A] Rumelhart, D.E., McClelland, J.L. and the PDP Research Group, "Parallel Distributed Processing", vol. 1, MIT Press, 1986.
- [RUME86B] Rumelhart, D.E., Hinton, G.E. and Williams, R.J., "Learning Representations by Backpropagating Errors", *Nature* 323, pgs. 533-536, 1986.
- [RUME86C] Rumelhart, D.E., Hinton, G.E. and Williams, R.J., "Learning Internal Representations by Error Propagation", *Parallel Distributed Processing vol. I* (MIT Press Cambridge, MA), D. Rumelhart and J. McClelland Eds., pgs. 318-362, 1986.
- [SAIT88] Saito, K. and Nakano, R., "Medical Diagnostic Expert System Based on PDP Model", *NTT Communications and Information*

Processing Laboratories, Yokosuka-shi, Kanagana, Japão, pgs. 255-262, 1988.

[SAMA88] Samad, T., "Towards Connectionist Rule-Based Systems", Proceedings of the 1988 Conference on Neural Information Processing Systems, vol II, pgs. 525-531, 1988.

[SEJN87] Sejnowski, T. and Rosenberg, C. "Parallel Networks that Learn to Pronounce English Text", Complex Systems, vol. 1, pgs. 145-168, 1987.

[SIMP90] Simpson, P., "Artificial Neural Systems: Foundations, Paradigms, Applications and Implementations", Pergamon Press Inc., 1990.

[THEO87] Theoto, M.T. and Koizumi, M.S., "The Expert Environment - A Case Study", Preprints of the Second International Fuzzy Systems Association Congress, vol. 1, Tokio, pgs. 380-383, 1987.

[VARE91] Varejão, F.M., Machado, R.J., and Milidui, R.L., "ALBA - Um Ambiente para Classificadores Bayesianos por Aglomeração", Rio Scientific Center, Technical Report CCR-125, 1991.

[WALT83] Waltz, D., Genesereth, M., Hart, P., Hendrix, G., Joshi, A., McDermott, J., Mitchel, T.M., Nilsson, N., Wilensky, R., "Artificial Intelligence: An Assessment of the State of the Art and Recommendation for the Future Directions", AI Magazine, vol. 4, num. 3, 1983.

[WATE86] Waterman, D., "A Guide to Expert Systems", Addison-Wesley Publishing, 1986.

[WHIT90] Whitley, D., Starkweather, T. and Bogart, C., "Genetic Algorithms and Neural Networks: Optimizing Connections and Connectivity", Parallel Computing 14, pgs. 347-361, North-Holland, 1990.

[WINO72] Winograd, T., "Understanding Natural Language", Cognitive Psychology, 1, pgs. 1-191, 1972.

- [WINS75] Winston, P., "Learning Structural Descriptions from Examples",
The Psychology of Computer Vision, McGraw-Hill, NY, 1975.
- [ZADE65] Zadeh, L.A., "Fuzzy Sets", Information and Control, vol. 8, pgs.
338-353, 1965.
- [ZADE78] Zadeh, L.A., "Fuzzy Sets as a Basis for a Theory of Possibility",
Fuzzy Sets and Systems, vol. 1, pgs. 3-28, 1978.

APÊNDICE I

Algoritmo de Punição e Recompensa: Versão de Arranque

Início

- Criar para cada arco da rede um acumulador para recompensas e um acumulador para punições, ambos com valor inicial 0
- Fazer todos os pesos sinápticos da rede iguais a 1.
- Para cada exemplo da base de dados de treinamento faça:
 - Propagar as crenças dos nós de entrada até a camada de hipóteses.
 - Para cada arco que alcança um nó-hipótese faça:
 - Se o nó-hipótese alcançado corresponde à classe correta do caso então
 - Propague retroativamente a partir deste nó até os nós de entrada, incrementando o acumulador de recompensas de cada arco percorrido pelo produto (Recompensa):

Fluxo evidencial . ativação do neurônio de destino . importância do exemplo

senão

- Propague retroativamente a partir deste nó até a camada de entrada, decrementando o acumulador de punições de cada arco percorrido pelo produto (Punição):

Fluxo evidencial . ativação do neurônio de destino . importância do exemplo

- Fim-se

- Fim-para

- Fim-para

Fim

Algoritmo de Punição e Recompensa: Versão de Refinamento

Início

- Para cada exemplo da base de dados de treinamento faça:
 - Propague as crenças nos nós de entrada até a camada de hipóteses
 - Para cada hipótese H que é solução desse caso faça:
 - Calcule $a = 1 - \text{ativação}(H)$
 - Para cada arco que atinge o nó da hipótese H faça:
 - Propagar retroativamente a partir de H até os nós de entrada, incrementando o acumulador de recompensas de cada arco percorrido pelo produto (Recompensa):

α . fluxo evidencial . ativação do neurônio de destino . importância do exemplo

- Fim-para

- Fim-para

- Para cada hipótese H que não é solução do exemplo, mas apresenta ativação $> L_{\text{aceit}}$ faça:

- Fazer $\alpha = \text{ativação}(H)$

- Para cada arco que atinge o nó da hipótese H faça:

- propagar retroativamente a partir de H até os nós de entrada, incrementando o acumulador de punições de cada arco percorrido pelo produto (Punição):

a . fluxo evidencial . ativação do neurônio de destino . importância do exemplo

- Fim-para

- Fim-para

- Fim-para

Fim

Algoritmo de Poda e Normalização da Rede

Início - Para cada arco da rede faça:

- Calcule o valor líquido dos acumuladores
 $ACUMLIQ = \text{acumulador de recompensas} - \text{acumulador de punições}$
- Se $ACUMLIQ \leq 0$
então
 - Remova o arco da rede
 - Fim-para
- Fim-se
- Se acumulador de punições do arco > 0
então
 - Compute o peso do arco como

$$ACUMLIQ / MAXLIQ$$

onde $MAXLIQ = \text{máximo } ACUMLIQ \text{ na subrede da hipótese a qual pertence o arco}$

senão

- Compute o peso do arco como

$$\sqrt{T_{\text{aceit}}} + (1 - \sqrt{T_{\text{aceit}}}) \cdot ACUMLIQ / MAXLIQ$$

- Fim-se
- Se peso do arco $<$ Limiar de poda
então
 - Remova o arco da rede
- Fim-se
- Fim-para
- Remover todos neurônios das camadas de entrada e combinatória que percam conectividade com os nós-hipótese.

Fim