**Otimização de Redes Neurais RBF
Usando Algoritmos Genéticos e
sua Aplicação na Área Financeira**

Estéfane George Macedo de Lacerda

Orientação:

Prof^o Dr^o André C. P. de L. F. de Carvalho

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Área de Ciências de Computação e Matemática Computacional.

USP – São Carlos
Dezembro de 19

ABSTRACT

The choice of the topology of a RBF Neural Network is usually carried out by trial and error based on the designer experience. The most common training algorithms that define the network topology use local methods which have a large possibility of being trapped at a local minima, producing sub-optima solutions. Genetic Algorithms represent a global search method appropriate to find good solutions in complex search spaces, like the space of Neural Networks topologies. This work proposes a Genetic Algorithm for RBF networks optimisation limiting the search space through a clustering technique. The results achieved suggest that this optimisation improves the performance of RBF networks in finance applications.

RESUMO

A escolha da topologia de uma Rede Neural RBF é geralmente realizada por tentativa e erro baseado na experiência do projetista. Os algoritmos de treinamento existentes que determinam a topologia da rede utilizam métodos locais, que apresentam uma grande possibilidade de cair em mínimos locais gerando soluções sub-ótimas. Algoritmos Genéticos representam um método de busca global apropriado para encontrar boas soluções em espaços de busca complexos, como o espaço de busca das topologias das Redes Neurais. Este trabalho propõe um Algoritmo Genético para otimizar a topologia de redes RBF limitando o espaço de busca através de uma técnica de aglomeração. Os resultados obtidos sugerem que esta otimização melhora o desempenho de redes RBF em aplicações financeiras.

Pérolas esquecidas

Dá o que passas, como possas e quanto possas, em benefício dos outros, mas recorda sempre essas pérolas esquecidas...
O timbre de voz fraterna com quem ainda não simpatizas...
O sorriso acalhear para a visita inesperada...
O minuto de boa vontade no esclarecimento amigo...
A simples conversa reconfortante com a pessoa cuja presença te desagrada...
O silêncio generoso ante a provocação daqueles que ainda não te compreendem...
A insignificante gentileza na via pública...
A referência construtiva em favor dos ausentes...
O serviço singelo aos desconhecidos...
A oração pelos adversários...
A consideração para com as mais velhas...
O amparo à criança...
A ligeira visita aos doentes...
O bilhete afetuoso ao irmão necessitado de bom ânimo...
O carinho em casa...
O socorro aos desalentados...
A palavra otimista para quem te ouve...
A leitura edificante...
O respeito às situações que não conheces...
O auxílio à Natureza...
A cooperação desinteressada na bem...
Não te afastes do abençoado serviço a todas.
Os pequeninos gestos espontâneos da verdadeira fraternidade são alicerces seguros na construção do Reino de Luz e Amor.

Scheilla/Francisco Cândido Xavier.
In: Preces e Mensagens Espirituais.

Progresso e amor

Grande - é o avanço do progresso
Maior - será sempre a amar que ilumina.
Grande - é a inteligência dos que fabricam as pássaros metálicos que povoam os céus do mundo.
Maior - é a inteligência de quantos se utilizam deles para levantar a fraternidade entre os povos.
Grande - é a eficiência dos que engenharam máquinas que eliminam as distâncias.
Maior - é o espírito de responsabilidade e entendimento daqueles que as dirigem favorecendo o trabalho.
Grande - é o raciocínio de quantas se dedicam à radiaotelevisão, sustentando a informação rápida na vida comunitária.
Maior - é a bondade de quantos lhe manejam os recursos em auxílio da educação entre criaturas.
Grande - é a força de quantos organizam as maravilhas da imprensa.
Maior - é o poder de todos aqueles que escrevem para instruir e reconfortar os irmãos em humanidade.
Grande - é a segurança dos que formaram o trator destinado a revalver facilmente o solo.
Maior - é o mérito de quantos semeiam com humildade e devotamento, formulando os prodígios do pão na mesa.
Grande - é a técnica
Maior - é a compreensão.
Grande - é a cultura que ensina.
Maior - é a caridade que socorre.
Onde estiveres e seja com quem for, Ama sempre.
O progresso faz estruturas.
O amor acende a luz do caminho.
Por isto mesmo aprendemos a trabalhar e servir sempre, a fim de conquistarmos a felicidade maior.
Em verdade perante Deus por mais amplo o surto de evolução que nos caracterize a existência. não haverá progresso real sem a benção do amor.

Xavier, Francisco Cândido. In: Companheiro.
Ditado pelo Espírito Emmanuel.

- À memória de meu pai Eulício Farias de Lacerda;
- À minha mãe e irmãos pelo apoio recebido;
- Ao Instituto de Ciências Matemáticas e de Computação da USP de São Carlos/SP, juntamente com seus professores e funcionários;
- Ao Professor Doutor André C. P. de L. F. de Carvalho pela orientação e estímulo neste trabalho.

ÍNDICE

INTRODUÇÃO.....	1
1.1 APRESENTAÇÃO.....	1
1.2 ORGANIZAÇÃO DA DISSERTAÇÃO.....	2
1.3 CONVENÇÕES NOTACIONAIS.....	3
REDES DE FUNÇÕES BASE RADIAL.....	5
2.1 INTRODUÇÃO.....	5
2.2 FUNÇÕES RADIAIS.....	7
2.3 TREINAMENTO HÍBRIDO: FASE NÃO-SUPERVISIONADA.....	9
2.3.1 K-MÉDIAS.....	11
2.3.2 MAPAS AUTO-ORGANIZÁVEIS.....	13
2.3.3 HEURÍSTICAS PARA DETERMINAR A LARGURA.....	14
2.4 TREINAMENTO HÍBRIDO: FASE SUPERVISIONADA.....	15
2.4.1 MÍNIMOS QUADRADOS.....	15
2.4.2 LMS.....	19
2.4.3 PSEUDOINVERSA.....	23
2.4.4 WEIGHT DECAY.....	26
2.5 CROSSVALIDATION.....	26
2.6 FUNÇÕES DE ERRO.....	29
2.7 COMPARAÇÃO ENTRE AS REDES RBF E MLP.....	30
2.8 OLS.....	32
2.9 OUTROS MODELOS.....	37
2.10 RESUMO.....	37
ALGORITMOS GENÉTICOS.....	39
3.1 INTRODUÇÃO.....	39
3.1.1 OTIMIZAÇÃO DE FUNÇÃO.....	41
3.1.2 O CROMOSSOMO.....	42
3.1.3 A FUNÇÃO DE AVALIAÇÃO.....	43
3.1.4 SELEÇÃO.....	43
3.1.5 CROSSEOVER E MUTAÇÃO.....	45
3.1.6 ELITISMO.....	46
3.1.7 TERMINOLOGIA.....	46
3.2 COMPARAÇÃO COM OUTROS MÉTODOS.....	47
3.2.1 GERAR-E-TESTAR.....	47
3.2.2 SUBIDA DO MONTE.....	48
3.2.3 EXPLORATION E EXPLOITATION.....	48
3.2.4 RECOZIMENTO SIMULADO.....	49

3.3	SELEÇÃO	51
3.3.1	CONVERGÊNCIA PREMATURA	51
3.3.2	PROBLEMAS NO INTERVALO DA APTIDÃO	52
3.3.3	ESCALAMENTO DA APTIDÃO E RANKING	53
3.3.4	SELEÇÃO POR TORNEIO	55
3.3.5	AS SUBSTITUIÇÕES GERACIONAL E STEADY-STATE	56
3.3.6	AMOSTRAGEM ESTOCÁSTICA UNIVERSAL	56
3.4	COMO OS ALGORITMOS GENÉTICOS FUNCIONAM	58
3.4.1	O TEOREMA DOS ESQUEMAS	58
3.4.2	A HIPÓTESE DOS BLOCOS DE CONSTRUÇÃO E OUTROS TÓPICOS	62
3.5	OPERADORES GENÉTICOS	63
3.5.1	REPRESENTAÇÃO BINÁRIA	63
3.5.2	REPRESENTAÇÃO COM NÚMEROS REAIS	65
3.6	EPÍLOGO	66
ALGORITMOS GENÉTICOS PARA REDES NEURAIS		67
4.1	INTRODUÇÃO	67
4.1.1	UM EXEMPLO	68
4.1.2	POR QUE USAR OS ALGORITMOS GENÉTICOS	70
4.2	OTIMIZAÇÃO DA TOPOLOGIA	71
4.2.1	CODIFICAÇÃO DIRETA	71
4.2.2	CODIFICAÇÃO INDIRETA	75
4.2.3	COMENTÁRIOS	82
4.2.4	O PROBLEMA DO MAPEAMENTO ESTRUTURAL-FUNCIONAL	82
4.3	TREINAMENTO	83
4.3.1	INTRODUÇÃO	83
4.3.2	COMENTÁRIOS	85
4.4	EVOLUÇÃO DA REGRA DE APRENDIZADO	86
4.5	OTIMIZAÇÃO DE REDES DE FUNÇÕES BASE RADIAL	89
4.5.1	INTRODUÇÃO	89
4.5.2	OTIMIZAÇÃO	90
4.5.3	OUTROS MODELOS	99
4.6	COMENTÁRIOS FINAIS	100
UM NOVO ALGORITMO GENÉTICO		102
5.1	DESCRIÇÃO	102
5.1.1	REPRESENTAÇÃO	103
5.1.2	OPERADORES GENÉTICOS	104
5.1.3	APTIDÃO	106
5.1.4	IMPLEMENTAÇÃO	108
5.2	EXPERIMENTOS	109
5.2.1	EXPERIMENTO I: REGRESSÃO	109
5.2.2	EXPERIMENTO II: CLASSIFICAÇÃO	113
5.2.3	EXPERIMENTO III: REGRESSÃO	116
5.3	COMENTÁRIOS FINAIS	117
CONCLUSÃO		119
BIBLIOGRAFIA		121
REDES NEURAIS PERCEPTRON DE MULTICAMADAS		126
A.1	INTRODUÇÃO	126
A.2	FORWARD PROPAGATION	127
A.3	GRADIENTE DESCENDENTE	129
A.4	CÁLCULO DAS DERIVADAS	131

A.5	MOMENTO.....	134
A.6	BACKPROPAGATION.....	136
A.7	EARLY STOPPING.....	138
AVALIAÇÃO DE RISCO DE CRÉDITO		140
B.1	INTRODUÇÃO.....	141
B.2	OS C'S DO CRÉDITO.....	142
B.2.1	CARÁTER.....	142
B.2.2	CAPACIDADE.....	142
B.2.3	CAPITAL.....	143
B.2.4	CONDIÇÃO.....	143
B.2.5	COLATERAL.....	144
B.3	MÉTODOS PARA AVALIAÇÃO DE RISCO DE CRÉDITO.....	144
B.3.1	CREDIT SCORING.....	146
B.3.2	ANÁLISE DISCRIMINANTE.....	147
B.3.4	REDES NEURAIS ARTIFICIAIS.....	152
B.3.5	ÁRVORE DE DECISÃO.....	153
B.3.6	EPÍLOGO.....	159

1

INTRODUÇÃO

"Não vale contemplar sem agir, nem sonhar sem fazer"
(Espírito Emmanuel)

"Todo empreendimento exige o tributo da luta"
(Espírito Marco Prisco)

*"Sempre que se te faça possível, pede aos Céus te fortaleça com a
Paciência para que não se te dificulte o caminho para a frente"*
(Espírito Emmanuel)

1.1 APRESENTAÇÃO

Esta Dissertação diz respeito à utilização de Algoritmos Genéticos para a otimização de parâmetros de Redes Neurais do tipo *Radial Basis Function*, RBF. Para isto, é proposta uma nova forma de otimização genética destas redes. Com o objetivo de avaliar esta otimização, serão utilizadas, entre outras, bases de dados financeira da área de análise de crédito.

Nas última décadas, a área de Inteligência Artificial alcançou significativo desenvolvimento. Uma de suas subáreas mais promissoras é a de Redes Neurais. Em todo mundo, uma grande quantidade de recursos humanos e financeiros tem sido confiado a esse campo de trabalho. Áreas como reconhecimento de padrões, sistemas de controle e processamento de sinais possuem uma forte interação com as Redes Neurais. Aplicações de Redes Neurais espalham-se em áreas diversas. Principalmente onde regressão, classificação e previsão de séries temporais são requeridas. Exemplos vão desde previsão financeira, passando por aplicações industriais, até reconhecimento da voz humana.

Apesar do crescente número de pesquisas, não existe ainda um método prático e automático para determinar a topologia de uma Rede Neural. O desempenho das redes depende crucialmente da topologia utilizada. Nem grande demais. Nem pequena demais. O ponto ótimo é difícil de ser encontrado. A busca por tentativa e erro é o método mais usado. Tal processo dificulta a produção em larga escala de Redes Neurais. Têm sido propostos vários métodos. Estes, porém, baseiam-se em heurísticas que restringem o espaço de busca. Neste contexto, surge a proposta de utilização dos Algoritmos Genéticos como uma alternativa para determinar a topologia de uma Rede Neural.

Os Algoritmos Genéticos também representam uma promissora área de pesquisa. Esta técnica tem atraído muita atenção por sua capacidade de resolver problemas de otimização complexos com um tempo de processamento aceitável. Servindo, particularmente, para aqueles problemas que necessitam uma solução globalmente ótima, que pode estar escondida entre soluções localmente ótimas. Exemplos bem conhecidos são: otimização de função numéricas complexas (com descontinuidades e vários pontos de máximo) e otimização de problemas combinatoriais (e.g. problema do caixeiro viajante e problemas de agendamento de tarefas) .

Esta dissertação também pode ser vista como uma aplicação dos Algoritmos Genéticos em um problema de difícil otimização por métodos convencionais (a otimização da topologia da Rede Neural). E como tal poderá trazer contribuições e esclarecimentos da aplicação prática dos Algoritmos Genéticos em problemas de otimização em geral. Este trabalho está organizado como segue.

1.2 ORGANIZAÇÃO DA DISSERTAÇÃO

Redes RBF e Algoritmos Genéticos são técnicas recentes, importantes e de interesse geral. Porém, ambas são ainda pouco utilizadas e conhecidas pelo seu público alvo no Brasil. Em vista disto, torna-se imperativo a elaboração de tutoriais sobre as mesmas. Assim, os capítulos 2 e 3 cobrem os principais aspectos de redes RBF e Algoritmo Genéticos, respectivamente.

Existem muitas variações de Algoritmos Genéticos. Em geral, cada autor cria seu próprio Algoritmo Genético para resolver um problema específico. São muitas as idéias que beneficiam aqueles que querem projetar um Algoritmo Genético. O capítulo 4 reúne vários trabalhos, apresentando diversos modelos de Algoritmos Genéticos para redes RBF e Redes Perceptron de Multicamadas ou rede MLP (do inglês, *MultiLayer Perceptron*). A inclusão de modelos para este último tipo de rede é justificado pela sua semelhança estrutural com as redes RBF. Esta semelhança permite que sugestões de uma seja aproveitada pela outra.

No capítulo 5 é apresentado um novo modelo de Algoritmos Genéticos para otimização de redes RBF. Uma técnica de aglomeração é usada para manter as topologias em áreas promissoras do espaço de busca. Um operador genético diferente dos tradicionais é também proposto e desenvolvido.

Por fim, o capítulo 6 conclui e discute os resultados desta dissertação. Comparações da abordagem proposta com outras abordagens de otimização de redes RBF serão feitas. Para isto, são utilizados base de dados na área financeira e de outras áreas.

Além disso, dois apêndices foram incluídos. No apêndice A, é descrito a Rede Neural MLP, pois grande parte do capítulo 4 refere-se a este tipo de Rede Neural. O apêndice B descreve como é realizada atualmente a Avaliação de Crédito e como este pode se beneficiar das técnicas propostas nesta Dissertação. Antes de iniciar o capítulo 2, alguns convenções adotadas na notação serão apresentadas.

1.3 CONVENÇÕES NOTACIONAIS

Números escalares serão representados por letras em itálico, por exemplo: y , x e a . Matrizes serão representadas por letras maiúsculas em negrito, por exemplo:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

A transposta de uma matriz $\mathbf{A}$ será representada por $\mathbf{A}^T$. Se $\mathbf{B} = \mathbf{A}^T$, então $b_{ji} = a_{ij}$. A matriz $\mathbf{A}^{-1}$ representará a inversa da matriz $\mathbf{A}$. A matriz $\mathbf{I}_m$ representará a matriz identidade $m \times m$ (i. e. aquela matriz em que $a_{ii} = 1$ e $a_{ij} = 0$). Vetores serão representados por letras minúsculas em negrito, por exemplo, $\mathbf{x}$, $\mathbf{w}$ e $\boldsymbol{\mu}$. Os vetores serão representados por uma matriz-coluna (i.e. matriz com uma única coluna) por exemplo:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Na maioria das vezes os vetores serão escritos na forma mais prática (ocupando apenas uma linha). Por exemplo,

$$\mathbf{x} = [x_1, x_2, \dots, x_n]^T$$

Por que a transposta de um matriz-linha é uma matriz-coluna. Por fim, o produto escalar entre dois vetores $\mathbf{x}$ e $\mathbf{y}$ será escrito como:

$$\mathbf{x}^T \mathbf{y}$$

A seguir será apresentado o capítulo 2.

2

REDES DE FUNÇÕES BASE RADIAL

*“Trabalho, ação, aprendizado, melhoria!...
Não te ponhas a espera deles sob a imaginária incapacidade de
procura-los, a vista de imperfeições e defeitos que te
marcaram ontem. Realização pede apoio da fé. Mãos a obra.
Tudo o que serve para corrigir, elevar, educar e construir nasce
primeiramente no esforço da vontade unida a decisão”*
(Espírito Emmanuel)

2.1 INTRODUÇÃO

A Rede Neural de Funções Base Radial ou rede RBF (do inglês *Radial Basis Function Network*) tem sido usada em regressão, classificação de padrões e previsão de séries temporais. A formulação original da rede RBF, no contexto das Redes Neurais, foi proposta por Broomhead e Lowe (1988), como uma Rede Neural de três camadas: uma camada de entrada, uma intermediária e uma de saída. O problema da rede RBF é aprender o mapeamento presente no **conjunto de treinamento**, que é um conjunto de pares entrada-saída representado por:

$$\{(\mathbf{x}_i, t_i); i = 1 \dots p\} \quad (2.1.1)$$

Onde, $\mathbf{x}_i$ é denominado vetor de entrada:

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{in} \end{bmatrix} \quad (2.1.2)$$

E t_i é a saída desejada. O par $(\mathbf{x}_i, t_i)$ que é denominado de padrão, exemplo ou observação. Este tipo de aprendizado ou treinamento é denominado **aprendizado supervisionado**. O aluno (i.e. a rede) aprende através de exemplos com entrada e resposta desejada fornecidos por um “professor”. O **aprendizado não-supervisionado** é realizado com exemplos que não têm resposta ou saída desejada. Ou seja: não há “professor”. Quando apropriadamente treinada, a rede RBF é capaz realizar o mapeamento do conjunto de treinamento do seguinte modo:

$$y(\mathbf{x}_i) = \sum_{j=1}^m w_j \phi_j(\|\mathbf{x}_i - \mu_j\|) + w_0 \quad (2.1.3)$$

Onde ϕ_j é a função de ativação da unidade j , também conhecida como função base radial, que será descrita mais tarde. O vetor:

$$\mu_j = \begin{bmatrix} \mu_{j1} \\ \mu_{j2} \\ \vdots \\ \mu_{jn} \end{bmatrix} \quad (2.1.4)$$

é denominado de vetor centro. Cada unidade intermediária j possui um vetor centro.

$$\|\mathbf{x}_i - \mu_j\| = \sum_{k=1}^n (x_{ik} - \mu_{jk})^2 \quad (2.1.5)$$

é a distância (euclidiana) entre os vetores $\mathbf{x}_i$ e μ_j . w_k é peso associado à conexão entre a unidade intermediária j e a unidade de saída. w_0 é denominada *bias*. Finalmente, $y(\mathbf{x}_i)$ é denominado de saída real da rede.

Vale frisar que a rede RBF pode ter múltiplas saídas. No entanto, para os tipos de treinamento apresentados neste texto, uma rede com múltiplas saídas pode ser dividida em um conjunto de redes com uma única saída. Portanto, para simplificar a notação, todas as redes serão consideradas apresentando uma única saída. Na seção seguinte serão descritos alguns dos parâmetros mencionados anteriormente.

Figura 2.1. Rede RBF com uma unidade de saída

2.2 FUNÇÕES RADIAIS

Funções Radiais são uma classe especial de funções. Elas possuem a propriedade de decrescer ou crescer monotonicamente a medida que o vetor de entrada $\mathbf{x}$ se distancia do vetor centro μ (Orr, 1996). A função radial mais utilizada é a Gaussiana¹:

$$\phi(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mu\|^2}{2\sigma^2}\right) \quad (2.2.1)$$

O parâmetro σ é denominado largura e serve como um fator de escala. Vários tipos de funções radiais são mostrados na Tabela 2.1. Algumas funções radiais são do tipo localizadas (ex. Gaussiana e multiquadrática inversa): quanto mais o vetor $\mathbf{x}$ se distancia do vetor centro μ , menor é a sua resposta. Tais funções criam um campo receptivo em torno do seu centro, onde a abertura do campo depende da largura σ . Funções não localizadas (ex. *thin-plate-spline* e multiquadrática) não possuem este comportamento: quando $\|\mathbf{x} - \mu\| \rightarrow \infty$, tem-se que $\phi(\mathbf{x}) \rightarrow \infty$.

¹ - Funções Radiais podem assumir formas mais gerais como a Gaussiana multivariável $\phi(\mathbf{x}) = \exp\left(-(\mathbf{x} - \mu)^T \mathbf{K}^{-1}(\mathbf{x} - \mu)\right)$, onde $\mathbf{K}$ é uma matriz de covariância.

Um gráfico, ilustrado na Figura 2.2, contém várias funções radiais com vetor de entrada unidimensional, centro $\mu = 0$ e largura $\sigma = 1$. A seguir será apresentado o treinamento híbrido de redes RBF.

Figura 2.2. Gráfico de vários tipos de funções radiais

Tabela 2.1. Funções Radiais

FUNÇÃO RADIAL	FÓRMULA, onde $v = \mathbf{x}-\mu $
Gaussiana	$\phi(v) = \exp(-\frac{v^2}{2\sigma^2})$
Linear	$\phi(v) = v$
Cúbica	$\phi(v) = v^3$
<i>Thin-Plate-Spline</i>	$\phi(v) = v^2 \log(v)$
Multiquadrática	$\phi(v) = (v^2 + \sigma^2)^{1/2}$
Multiquadrática Inversa	$\phi(v) = \frac{1}{(v^2 + \sigma^2)^{1/2}}$

2.3 TREINAMENTO HÍBRIDO: FASE NÃO-SUPERVISIONADA

O treinamento híbrido combina um treinamento supervisionado com um não-supervisionado. A fase não-supervisionada define os centros das funções base radial. A fase supervisionada define os pesos $\{w_j, j=1...m\}$ das conexões entre as unidades da camada intermediária e os unidades da camada de saída. Para isto, é utilizado um método linear rápido de otimização, como, por exemplo, o método dos mínimos quadrados.

O treinamento supervisionado só é possível quando os vetores centros $\{\mu_j, j=1...m\}$ e as larguras não se modificam durante o treinamento, pois de outro modo a otimização seria não-linear. As larguras são pré-determinadas por uma heurística (ver seção 2.3.3). Os vetores centros podem ser pré-determinados de três modos:

- A) Selecionando aleatoriamente os centros dos padrões de treinamento.
- B) Distribuindo os centros sobre uma grade regular.
- C) Utilizando uma técnica de aglomeração ou agrupamento (do inglês, *clustering*).

A seguir será estudado treinamento não-supervisionado. O treinamento supervisionado será visto na seção 2.4.

A) Centros aleatoriamente selecionados

Os centros são fixados sobre vetores de entrada selecionados aleatoriamente do conjunto de treinamento. Esta técnica é simples, mas é necessário que conjunto de treinamento seja uma amostra bem representativa do problema abordado. Quando isto não ocorre, a rede pode apresentar mal desempenho ou exigir um grande número de unidades intermediárias. Além disso, este método pode causar problemas numéricos (mal-condicionamento, seção 2.4.3), como exemplo; devido, pôr exemplo, a centros escolhidos muitos próximos um do outro (Chen et al, 1991).

B) Centros fixados em uma grade regular

Os centros são fixados em uma grade regular, Figura 2.3, cobrindo todo o espaço de entrada. Este método exige grande número de unidades intermediárias quanto o vetor de entrada tem muitas dimensões por causa do **curso da dimensionalidade** (do inglês, *curse of dimensionality*) descrito como segue:

Considere que o vetor de entrada é n -dimensional e que cada uma das n dimensões da grade é dividida em a intervalos. Então, o número de unidades intermediárias é igual a a^n . Consequentemente, aumentando-se a dimensionalidade n do vetor de entrada acarreta um crescimento exponencial do número de unidades intermediárias. O curso da dimensionalidade é também um problema que afeta muitos métodos de Estatística e Redes Neurais. Para mais detalhes, ver (Bishop, 1995).

Figura 2.3. Distribuição de centros sobre uma grade regular

C) Técnicas de aglomeração

Os centros são definidos por um aprendizado não-supervisionado. Os centros são distribuídos, através de uma técnica de aglomeração, formando agrupamentos nas regiões onde há maior povoamento de padrões. Ver Figura 2.4. Experimentos mostram que este método gera um número de unidades intermediárias menor do que aquele gerado pelos métodos anteriores (Moody e Darken, 1989). As técnicas de aglomeração mais utilizadas são:

- Algoritmo k-médias (seção 2.3.1)
- Os mapas auto-organizáveis de Kohonen (seção 2.3.2)

A seguir estes dois métodos serão apresentados.

Figura 2.4. Aglomerados obtidos pelo algoritmo k-médias

2.3.1 K-MÉDIAS

O algoritmo k-médias (Anderberg, 1973; MacQueen, 1967) é uma das mais populares técnicas de aglomeração. Ele divide o conjunto de vetores de entrada em m grupos ou aglomerados $\{S_j, j=1...m\}$ (ver Figura 2.4) e determina os centros pelo ponto central de cada aglomerado, ou seja:

$$\mu_j = \frac{1}{n_j} \sum_{\mathbf{x}_i \in S_j} \mathbf{x}_i \quad (2.3.1.1)$$

Onde, n_j é o número de vetores de entrada que pertencem ao aglomerado S_j . O critério para determinar os aglomerados é a minimização da função (2.3.1.2) que representa a soma dos quadrados das distâncias de cada vetor de entrada ao centro do seu aglomerado.

$$J = \sum_{j=1}^m \sum_{\mathbf{x}_i \in S_j} \|\mathbf{x}_i - \mu_j\|^2 \quad (2.3.1.2)$$

O algoritmo k-médias começa inicializando os centros arbitrariamente. Em seguida, através de procedimento iterativo, os vetores de entrada são transferidos de um aglomerado para outro até atingir uma situação estável. O algoritmo k-médias é apresentado no Algoritmo 2.2. Outra versão do k-média, que é adaptativa, é mostrada no Algoritmo 2.3. Nesta versão não há necessidade de armazenar a informação sobre a qual aglomerado pertence cada vetor de entrada. Esta versão é útil para sistemas de aprendizado adaptativo em tempo real (Moddy e Darken, 1989).

O número m de centros (i.e. o número de aglomerados) é normalmente determinado pelo método *cross-validation* ou *hold-out* que serão vistos mais adiante. A seguir é mostrada outra técnica de aglomeração.

Algoritmo 2.2. K-médias

Definir o número de aglomerados m

/* Inicializar os m centros dos aglomerados com os m primeiros vetores de entrada */

para $i = 1$ **até** m **faça** $\mu_i \leftarrow \mathbf{x}_i, S_i \leftarrow \{\mathbf{x}_i\}, c_i \leftarrow i$

/* Atribuir o restante dos vetores de entrada ao aglomerado mais próximo */

para $i = m+1$ **até** p **faça**

μ_q é tal que minimiza $\|\mathbf{x}_i - \mu_j\|$ /* Determinar o μ_q mais próximo de $\mathbf{x}_i$ */

$S_q \leftarrow S_q \cup \{\mathbf{x}_i\}, c_i \leftarrow q$ /* Inserir $\mathbf{x}_i$ no aglomerado S_q */

fim para

para $i = 1$ **até** m **faça** $\mu_i \leftarrow \frac{1}{|S_i|} \sum_{\mathbf{x}_j \in S_i} \mathbf{x}_j$ /* Calcular o ponto médio de todos os aglomerado */

repita

para $i = 1$ **até** p **faça** /* Apresentar todos os padrões */

μ_q é tal que minimiza $\|\mathbf{x}_i - \mu_j\|$ /* Determinar o μ_q mais próximo de $\mathbf{x}_i$ */

se ($q \neq c_i$) **então** /* o aglomerado c_i não é o mais próximo de $\mathbf{x}_i$ */

$S_{c_i} \leftarrow S_{c_i} \setminus \{\mathbf{x}_i\}$ /* Deletar $\mathbf{x}_i$ do aglomerado S_{c_i} */

$S_q \leftarrow S_q \cup \{\mathbf{x}_i\}$ /* Inserir $\mathbf{x}_i$ no aglomerado S_q */

$\mu_q \leftarrow \frac{1}{|S_q|} \sum_{\mathbf{x}_j \in S_q} \mathbf{x}_j, \mu_{c_i} \leftarrow \frac{1}{|S_{c_i}|} \sum_{\mathbf{x}_j \in S_{c_i}} \mathbf{x}_j$ /* Atualizar o ponto médio dos aglomer. S_{c_i} e S_q */

$c_i \leftarrow q$

fim se

fim para

até (não ocorrer mais mudança nos aglomerados)

Algoritmo 2.3. K-média (versão adaptativa)

Defina um limite thr de similaridade de um mesmo centro entre o passo n e o passo $n-1$

para $j = 1$ **até** m **faça** $\mu_j \leftarrow \mathbf{x}_j$ /* Inicializar os m centros com os m primeiros vetores de entrada */

$n \leftarrow 0$

repita

para $i = 1$ **até** p **faça** /* Apresentar todos os padrões */

μ_q é tal que minimiza $\|\mathbf{x}_i - \mu_j\|$ /* Determinar o μ_q mais próximo de $\mathbf{x}_i$ */

$\mu_q \leftarrow \mu_q + \eta(\mathbf{x}_i - \mu_q)$ /* η é uma constante (taxa de aprendizado). ex. $\eta = 0,03$ */

fim para

$n \leftarrow n+1$

até (atingir uma situação estável, isto é, $\|\mu_j^{(n)} - \mu_j^{(n-1)}\| < thr$ para $j = 1 \dots m$)

2.3.2 MAPAS AUTO-ORGANIZÁVEIS

Os mapas auto-organizáveis de Kohonen ou SOM (do inglês, *Self-Organizing Map*; Kohonen, 1989) são conhecidos algoritmos de aprendizado não-supervisionado para treinamento de Redes Neurais. A versão do SOM implementada no simulador de Redes Neurais SNNS (Zell, 1995) para treinar centros de redes RBF é apresentada no Algoritmo 2.4 que é uma das mais simples.

A pré-condição de usar este algoritmo é normalizar todos os vetores de entrada para que tenham tamanho 1. Ao iniciar, o algoritmo atribui os centros aleatoriamente sobre os vetores de entrada. Em seguida, todos os vetores de entrada são apresentados um a um ao algoritmo. A cada apresentação, determina-se qual centro μ_q possui o maior produto escalar com o vetor de entrada $\mathbf{x}_i$ apresentado, isto é,

$$\mu_q \text{ é tal que maximiza } \mathbf{x}_i^T \mu_j$$

Como os vetores estão normalizado, o produto escalar fornece uma medida de distância entre um vetor de entrada e um centro. Portanto, o centro selecionado é aquele que está mais próximo do vetor de entrada. Em seguida tal centro é movido em direção ao vetor de entrada:

$$\mu_q \leftarrow \mu_q + \eta(\mathbf{x}_i - \mu_q) \quad (2.3.2.1)$$

Onde, η é uma constante denominada taxa de aprendizado. A seguir será mostrado como determinar o parâmetro largura das funções base.

Algoritmo 2.4. Mapa auto-organizável

Normalizar todos os vetores de entrada para que tenham comprimento 1

para $j = 1$ **até** m **faça** $\mu_j \leftarrow \mathbf{x}_j$, /* Inicializar os m centros com os m primeiros vetores de entrada */

para $n = 1$ **até** $n_{\max}$ **faça**

para $i = 1$ **até** p **padrões faça** /* Apresentar todos os padrões */

μ_q é tal que maximiza $\mathbf{x}_i^T \mu_j$ /* Determinar o μ_q mais próximo de $\mathbf{x}_i$ usando produto escalar*/

$\mu_q \leftarrow \mu_q + \eta(\mathbf{x}_i - \mu_q)$ /* taxa de aprendizado: $0 < \eta < 1$ */

fim para

fim para

2.3.3 HEURÍSTICAS PARA DETERMINAR A LARGURA

As heurísticas utilizadas para a definição da largura das funções base radial são soluções sub-ótimas que visam melhorar a generalização da rede. Algumas usam um único valor (global) de σ para todas as unidades intermediárias. Outras usam um valor diferente (local) de σ_j para cada unidade intermediária j . Uma lista de heurísticas encontradas na literatura é mostrada a seguir.

Primeira heurística: A largura σ para todas as unidades intermediárias é dada por (Moody e Darken, 1989):

$$\sigma = \frac{1}{m} \sum_{j=1}^m \left\| \mu_j - \mu_{\text{proximo}}^{(j)} \right\| \quad (2.3.3.1)$$

Onde, $\mu_{\text{proximo}}^{(j)}$ é o centro mais próximo de μ_j em termos de distância euclidiana.

Segunda heurística: a largura σ_j da unidade j é dada por (Saha e Keller, 1990):

$$\sigma_j^2 = \frac{1}{N} \sum_{\mathbf{x}_i \in \Psi_j} \left\| \mu_j - \mathbf{x}_i \right\|^2 \quad (2.3.3.2)$$

Onde, Ψ_j é o conjunto dos N vetores de entrada mais próximos do centro μ_j em termos de distância euclidiana.

Terceira heurística: a largura σ_j da unidade j é dada por (Hassoun, 1995):

$$\sigma_j = \alpha \left\| \mu_j - \mu_{\text{proximo}}^{(j)} \right\| \quad (2.3.3.3)$$

Onde, $\mu_{\text{proximo}}^{(j)}$ é o centro mais próximo de μ_j em termos de distância euclidiana;
 α é usualmente entre 1,0 e 1,5.

Quarta heurística: considere S_j um aglomerado de vetores de entrada obtido pelo algoritmo k-médias. A largura σ_j da unidade j é dada por:

$$\sigma_j^2 = \frac{1}{n_j} \sum_{\mathbf{x}_i \in S_j} \left\| \mu_j - \mathbf{x}_i \right\|^2 \quad (2.3.3.4)$$

Onde, S_j é o aglomerado de vetores de entrada obtido pelo algoritmo k-médias;
 n_j é o número de vetores de entrada pertencentes a S_j .

Pode ser interessante multiplicar a largura por uma constante visando melhor adaptação ao conjunto de treinamento.

2.4 TREINAMENTO HÍBRIDO: FASE SUPERVISIONADA

Após a definição dos centros e larguras, vem a fase supervisionada. Na fase supervisionada, os pesos são determinados por um método linear que resolve o problema dos mínimos quadrados. Os principais métodos utilizados são:

- O métodos dos mínimos quadrados usando equações normais (seção 2.4.1)
- O algoritmo LMS (seção 2.4.2)
- A matrix pseudo-inversa (seção 2.4.3)

2.4.1 MÍNIMOS QUADRADOS

Conforme já visto, a rede RBF realiza o mapeamento dos padrões de treinamento, $\{(\mathbf{x}_i, t_i), i = 1 \dots p\}$, do seguinte modo:

$$y(\mathbf{x}_i) = \sum_{j=1}^m w_j \phi_j(\mathbf{x}_i) \quad (2.4.1.1)$$

Para incluir o *bias* w_0 na Equação (2.4.1.1), basta criar uma unidade intermediária extra cujo valor da função de ativação é sempre 1, isto é, $\phi_0(\mathbf{x})_i = 1$. Considere o vetor coluna formado por todas as saídas reais da rede:

$$\mathbf{y} = \begin{bmatrix} y(\mathbf{x}_1) \\ y(\mathbf{x}_2) \\ \vdots \\ y(\mathbf{x}_p) \end{bmatrix} \quad (2.4.1.2)$$

que também pode ser escrita na forma mais prática (ocupa apenas uma linha):

$$\mathbf{y} = [y(\mathbf{x}_1), y(\mathbf{x}_2), \dots, y(\mathbf{x}_p)]^T \quad (2.4.1.3)$$

Considere também o vetor formado por todas as saídas desejadas da rede:

$$\mathbf{t} = [t_1, t_2, \dots, t_p]^T \quad (2.4.1.4)$$

Para que a rede realize o mapeamento dos padrões de treinamento, uma “boa aproximação” entre a saída real $y(\mathbf{x}_i)$ e a saída desejada t_i deve ser encontrada. Uma medida de aproximação poderia ser a distância entre os vetores $\mathbf{y}$ e $\mathbf{t}$:

$$E = \|\mathbf{y} - \mathbf{t}\|^2 = \sum_{i=1}^p [y(\mathbf{x}_i) - t_i]^2 \quad (2.4.1.5)$$

Pois quando uma soma de determinados números positivos é pequena conclui-se que tais números também são pequenos. O método dos mínimos quadrados obtém os pesos da rede minimizando o valor de E dado pela Equação (2.4.1.5), que é denominada de **soma dos erros quadráticos** ou **SSE** (do inglês, *sum of squared errors*). O procedimento utilizado na minimização é encontrado em livros de Cálculo (Leithold, 1977). Este procedimento consiste em diferenciar a função dada em (2.4.1.5) com relação a todos os pesos $\{w_j, j=1\dots m\}$:

$$\frac{\partial E}{\partial w_j} = 2 \sum_{i=1}^p [y(\mathbf{x}_i) - t_i] \frac{\partial y(\mathbf{x}_i)}{\partial w_j}, \quad \text{para } j = 1\dots m \quad (2.4.1.6)$$

Onde,

$$\frac{\partial y(\mathbf{x}_i)}{\partial w_j} = \phi_j(\mathbf{x}_i) \quad (2.4.1.7)$$

Em seguida, iguala-se o resultado a zero. Tem-se:

$$2 \sum_{i=1}^p [y(\mathbf{x}_i) - t_i] \phi_j(\mathbf{x}_i) = 0, \quad \text{para } j = 1\dots m \quad (2.4.1.8)$$

$$\sum_{i=1}^p y(\mathbf{x}_i) \phi_j(\mathbf{x}_i) = \sum_{i=1}^p t_i \phi_j(\mathbf{x}_i), \quad \text{para } j = 1\dots m \quad (2.4.1.9)$$

Considere agora os vetores:

$$\boldsymbol{\phi}_j = [\phi_j(\mathbf{x}_1), \phi_j(\mathbf{x}_2), \dots, \phi_j(\mathbf{x}_m)]^T \quad (2.4.1.10)$$

$$\mathbf{w} = [w_1, w_2, \dots, w_m]^T \quad (2.4.1.11)$$

Escrevendo (2.4.1.9) na notação de vetores obtém-se:

$$\boldsymbol{\phi}_j^T \mathbf{y} = \boldsymbol{\phi}_j^T \mathbf{t} \quad (2.4.1.12)$$

Para as m funções radiais ϕ_j tem-se:

$$\begin{bmatrix} \phi_1^T \mathbf{y} \\ \phi_2^T \mathbf{y} \\ \vdots \\ \phi_m^T \mathbf{y} \end{bmatrix} = \begin{bmatrix} \phi_1^T \mathbf{t} \\ \phi_2^T \mathbf{t} \\ \vdots \\ \phi_m^T \mathbf{t} \end{bmatrix} \quad (2.4.1.13)$$

Seja a matriz:

$$\Phi = \begin{bmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_m(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_p) & \phi_2(\mathbf{x}_p) & \cdots & \phi_m(\mathbf{x}_p) \end{bmatrix} \quad (2.4.1.14)$$

Então (2.4.1.13) pode ser escrita da seguinte forma:

$$\Phi^T \mathbf{y} = \Phi^T \mathbf{t} \quad (2.4.1.15)$$

Escrevendo (2.4.1.1) em notação matricial tem-se:

$$\mathbf{y} = \Phi \mathbf{w} \quad (2.4.1.16)$$

Substituindo (2.4.1.16) em (2.4.1.15) tem-se:

$$(\Phi^T \Phi) \mathbf{w} = \Phi^T \mathbf{t} \quad (2.4.1.17)$$

que é um simples sistema de equações lineares (note que se $\mathbf{A} = \Phi^T \Phi$ e $\mathbf{b} = \Phi^T \mathbf{t}$, então o sistema tem a forma conhecida $\mathbf{A} \mathbf{w} = \mathbf{b}$). As equações deste sistema são conhecidas por **equações normais**. Este sistema pode ser resolvido por métodos tradicionais, tais como Eliminação de Gauss (Press et al, 1988).

Frequentemente, aplicações de Redes Neurais envolvem muitos padrões com ruídos tornando quase impossível resolver o sistema devido a problemas numéricos conhecidos como mal-condicionamento. Em (Press et al, 1988) é recomendado usar **decomposição em valores singulares** para resolver tais problemas numéricos. Este método e o problema de mal-condicionamento serão descritos na seção (2.4.3). Antes, é instrutivo considerar a interpretação geométrica do problema dos mínimos quadrados.

A) Interpretação Geométrica dos Mínimos Quadrados

Sem perda de generalidade, esta discussão utilizará uma rede RBF com apenas duas unidades intermediárias e 3 exemplos de treinamento. Considere os dois vetores base ϕ_1 e ϕ_2 (2.4.1.10) correspondentes às duas unidade intermediária. Tais vetores expandem um sub-espço S (um plano), Figura 2.5, em um espaço de 3 dimensões através da combinação linear:

$$\mathbf{y} = \sum_{j=1}^2 w_j \phi_j \quad (2.4.1.18)$$

Variando w_1 e w_2 qualquer ponto $\mathbf{y}$ do plano pode ser obtido, porém $\mathbf{y}$ está restrito ao plano S . Seja $\mathbf{t} = [t_1, t_2, t_3]^T$ o vetor das saídas desejadas. Seja:

$$\mathbf{t} = \mathbf{t}_{\text{proj}} + \mathbf{t}_{\text{ort}} \quad (2.4.1.19)$$

Onde $\mathbf{t}_{\text{proj}}$ é a projeção de $\mathbf{t}$ sobre S e $\mathbf{t}_{\text{ort}}$ é a projeção de $\mathbf{t}$ ortogonal ao plano S . Intuitivamente, nota-se que o ponto do plano S mais próximo de $\mathbf{t}$ é igual à projeção de $\mathbf{t}$ sobre S . Portanto, a solução do problema dos mínimos quadrados (isto é aquela que minimiza a distância $\|\mathbf{y} - \mathbf{t}\|$) é obtida quando $\mathbf{y} = \mathbf{t}_{\text{proj}}$. Desta observação (que pode ser provada rigorosamente (Boldrini et al, 1980)), tem-se que: como $\mathbf{t}_{\text{ort}}$ é ortogonal a S , por definição:

$$\phi_j^T \mathbf{t}_{\text{ort}} = 0, \quad \text{para } j = 1..2 \quad (2.4.1.20)$$

Colocando na forma matricial,

$$\Phi^T \mathbf{t}_{\text{ort}} = 0 \quad (2.4.1.21)$$

Usando (2.4.1.19) em (2.4.1.21) obtém-se:

$$\Phi^T (\mathbf{t} - \mathbf{t}_{\text{proj}}) = 0 \quad (2.4.1.22)$$

Por fim, esta Equação resulta na mesma Equação encontrada em (2.4.1.15):

$$\Phi^T (\mathbf{t} - \mathbf{y}) = 0 \quad (2.4.1.23)$$

Observa-se também que vetor erro $\mathbf{e} = \mathbf{y} - \mathbf{t}$ é igual a $-\mathbf{t}_{\text{ort}}$ quando a distância $\|\mathbf{y} - \mathbf{t}\|$ é minimizada. Pois, $\mathbf{e} = \mathbf{t}_{\text{proj}} - \mathbf{t} = -\mathbf{t}_{\text{ort}}$. A seguir será descrito um outro modo de resolver o problema dos mínimos quadrados.

Figura 2.5. Interpretação geométrica dos mínimos quadrados.

2.4.2 LMS

O algoritmo LMS (do inglês *Least-Mean-Square*), também conhecido por outros nomes (regra Delta, regra do Adaline ou algoritmo Widrow-Holf), foi inicialmente utilizado por Widrow e Holf (1960) para treinar a Rede Neural Adaline que é um tipo de rede sem camada intermediária. Este algoritmo resolve o problema dos mínimos quadrados usando o método do gradiente encontrado em livros de Cálculos e descrito no seguinte exemplo:

Considere uma rede RBF com duas unidades intermediárias e um padrão de treinamento. Os pesos são determinado pelo mesmo critério usado na seção anterior, que é minimizar a Equação (2.4.1.5), que neste caso se resume a:

$$E = (y - t)^2 \quad (2.4.2.1)$$

Onde t é a saída desejada para o único padrão deste exemplo e y é a correspondente saída real da rede, que é dada por:

$$y = w_1 \phi_1 + w_2 \phi_2 \quad (2.4.2.2)$$

O gráfico de E em função dos pesos w_1 e w_2 é um parabolóide (i.e. uma parábola tridimensional). Ver Figura 2.6. O vetor $\mathbf{w}^{(*)} = [w_1^{(*)}, w_2^{(*)}]^T$ representa os pesos ótimos que minimizam o valor de E . Dos livros de Cálculo, sabe-se que a direção que leva ao ponto ótimo $(w_1^{(*)}, w_2^{(*)})$ a partir de um ponto arbitrário $(w_1^{(0)}, w_2^{(0)})$ é igual a direção oposta ao vetor gradiente. Tal direção aponta para o maior declive na superfície do parabolóide no ponto $(w_1^{(0)}, w_2^{(0)})$. O vetor gradiente ∇E é dado por:

$$\nabla E = \begin{bmatrix} \frac{\partial E}{\partial w_1} \\ \frac{\partial E}{\partial w_2} \end{bmatrix} \quad (2.4.2.3)$$

Com base neste resultado, obtém-se a primeira aproximação do vetor $\mathbf{w}^{(*)}$ deslocando $\mathbf{w}^{(0)}$ em direção oposta ao vetor gradiente, isto é, $-\nabla E$. Este deslocamento é dado por:

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} - \eta \nabla E^{(n)} \quad (2.4.2.4)$$

Onde η é uma constante denominada **taxa de aprendizado** que determina o tamanho do deslocamento. Efetuando sucessivos deslocamentos $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \mathbf{w}^{(3)}, \dots$ fatalmente o ponto $\mathbf{w}^{(*)}$ será encontrado (ou uma boa aproximação dele). Esta é a idéia do método do gradiente. As derivadas do vetor gradiente são calculadas a seguir.

Figura 2.6. Superfície do erro quadrático em função dos pesos

A) O cálculo das derivadas

Utilizando a regra da cadeia, os componentes do vetor gradiente são calculados como segue:

$$\frac{\partial E}{\partial w_1} = \frac{\partial [(y-t)^2]}{\partial y} \frac{\partial y}{\partial w_1} = 2(y-t) \frac{\partial y}{\partial w_1} \quad (2.4.2.5)$$

$$\frac{\partial y}{\partial w_1} = \frac{\partial (\phi_1(\mathbf{x})w_1 + \phi_2(\mathbf{x})w_2)}{\partial w_1} = \phi_1(\mathbf{x}) \quad (2.4.2.6)$$

Então obtém-se:

$$\frac{\partial E}{\partial w_1} = 2(y-t)\phi_1(\mathbf{x}) \quad (2.4.2.7)$$

Do mesmo modo obtém-se:

$$\frac{\partial E}{\partial w_2} = 2(y-t)\phi_2(\mathbf{x}) \quad (2.4.2.8)$$

Com base nestes resultados, a Equação (2.4.2.4) é desenvolvida a seguir.

B) Algoritmo LMS

Substituindo (2.4.2.7) e (2.4.2.8) em (2.4.2.4) obtém a regra de ajuste de pesos em cada interação n :

$$w_j^{(n+1)} = w_j^{(n)} - \eta 2(y^{(n)} - t)\phi_j(\mathbf{x}), \quad \text{para } j = 1...2 \quad (2.4.2.9)$$

Absorvendo o fator 2 na constante η e substituindo $(y^{(n)} - t)$ por $-(t - y^{(n)})$, obtém-se a regra de aprendizado do algoritmo LMS para redes RBF:

$$w_j^{(n+1)} = w_j^{(n)} + \Delta w_j^{(n)} \quad (2.4.2.10)$$

$$\Delta w_j^{(n)} = \eta (t - y^{(n)}) \phi_j(\mathbf{x}) \quad (2.4.2.11)$$

Onde $\Delta w_j^{(n)}$ é o ajuste do peso w_j na interação n . Ver Algoritmo 2.1. Este resultado² pode ser facilmente generalizado para uma rede RBF com m unidades

² É devido a diferença $(t-y)$ entre a saída desejada e a saída real que este algoritmo é conhecido como regra Delta.

intermediárias. Para mais de um padrão, a mudança de pesos segue na média a direção oposta do gradiente (Anderson, 1995). Contudo, o desempenho do LMS depende da taxa de aprendizado η .

Algoritmo 2.1. LMS para redes RBF.

```

para  $j = 1$  até  $m$  faça  $w_j \leftarrow 0$       /* Inicializar os pesos com o valor zero */
para  $n = 1$  até MaxCiclos faça      /* MaxCiclos é o número de iterações */
    para  $i = 1$  até  $p$  faça      /*  $p$  é o número de padrões */
         $y_i \leftarrow \sum_{j=1}^m w_j \phi_j(\mathbf{x}_i)$ 
        para  $j = 1$  até  $m$  faça  $w_j \leftarrow w_j + \eta(t_i - y_i)\phi_j(\mathbf{x}_i)$ 
    fim para
fim para

```

C) Taxa de aprendizado.

O processo de convergência do algoritmo tem o comportamento ilustrado na Figura A.6. Quando a taxa de aprendizado η é muito baixa, acarreta ajustes de pesos $\Delta w_j^{(n)}$ muito baixos, levando o algoritmo a convergir lentamente. Quando a taxa de aprendizado η é muito alta, o algoritmo não convergirá, apresentando um comportamento divergente e oscilatório, que é mostrado na Figura A.7.

Existem vários métodos para definir a taxa de aprendizado. Manter η constante durante o treinamento é um dos mais simples e não garante a convergência (Bishop, 1995). Um método muito usado é fazer η variar com o tempo (Robbins e Monro, 1951), isto é:

$$\eta(n) = \frac{c}{n} \quad (2.4.2.12)$$

Onde c é uma constante. Outros métodos alternativos tem sido propostos em (Darken e Moody, 1992). O algoritmo LMS é útil em sistemas adaptativos em tempo real em que os padrões chegam continuamente e são descartados. Mais informações sobre o LMS encontram-se em (Haykin, 1994; Widrow e Lehr, 1990). A seguir mais uma maneira de resolver o problema dos mínimos quadrados será apresentada.

2.4.3 PSEUDOINVERSA

Conforme discutido na seção 2.4.1, o vetor peso $\mathbf{w}$ da rede RBF determinado pelo sistema (2.4.1.17) é dado por:

$$\mathbf{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t} \quad (2.4.3.1)$$

Em muitos problemas práticos, a matriz $\Phi^T \Phi$ é mal-condicionada. O que significa dizer que calcular a sua inversa $(\Phi^T \Phi)^{-1}$ é quase impossível. O método de decomposição em valores singulares é normalmente empregado para resolver tal problema. Para o treinamento, o que mais interessa neste método, é uma matriz denominada **pseudo-inversa** que é equivalente a $(\Phi^T \Phi)^{-1} \Phi^T$. Então, os pesos são obtidos por:

$$\mathbf{w} = \Phi^+ \mathbf{t} \quad (2.4.3.2)$$

Onde Φ^+ é a matriz pseudo-inversa. A seguir alguns esclarecimentos serão dados sobre este método e problemas de mal-condicionamento.

A) Mal-condicionamento

Considere dois sistemas lineares: o sistema I que é dado por:

$$\begin{bmatrix} 1,000 & 1,000 \\ 1,000 & 1,010 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 2,000 \\ 2,010 \end{bmatrix}$$

Cuja solução é $(w_1^{(0)}, w_2^{(0)}) = (1,000; 1,000)$ e o sistema II, dado por:

$$\begin{bmatrix} 1,000 & 1,000 \\ 1,001 & 1,000 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 2,000 \\ 2,010 \end{bmatrix}$$

Os dois sistemas são praticamente idênticos. A única diferença é que a primeira matriz do sistema II sofreu uma pequena perturbação em relação a primeira matriz do sistema I. Apesar disso, é possível imaginar que a solução do sistema I seja uma boa aproximação da solução do sistema II. De fato, substituindo a solução do sistema I, $(w_1^{(0)}, w_2^{(0)}) = (1,000; 1,000)$, no sistema II tem-se:

$$\begin{bmatrix} 1,000 & 1,000 \\ 1,001 & 1,000 \end{bmatrix} \begin{bmatrix} 1,000 \\ 1,000 \end{bmatrix} = \begin{bmatrix} 2,000 \\ 2,001 \end{bmatrix}$$

Que gera um erro dado pela vetor $\mathbf{r}$ (resíduo) de:

$$\mathbf{r} = \begin{bmatrix} 2,000 \\ 2,010 \end{bmatrix} - \begin{bmatrix} 2,000 \\ 2,001 \end{bmatrix} = \begin{bmatrix} 0,000 \\ 0,009 \end{bmatrix}$$

O surpreendente é que, apesar do resíduo ser pequeno, a solução do sistema I **não** é uma solução aproximada do sistema II. A solução exata do sistema II é $(w_1^{(II)}, w_2^{(II)}) = (10,000; -8,000)$ que difere largamente da solução do sistema I. Matrizes que apresentam este comportamento são denominadas de matrizes mal-condicionadas.

Assim, quando o sistema é mal-condicionado pequenas perturbações na matriz fazem a solução variar muito. Do ponto de vista geométrico, tais problemas numéricos são descritos como segue: os vetores ϕ_1 e ϕ_2 da Figura 2.7(a) exigem pesos relativamente pequenos para gerar um combinação linear de y . Por outro lado, problemas surgem quando ϕ_1 e ϕ_2 são aproximadamente paralelos, Figura 2.7(b), uma vez que, além de requerer pesos grandes, pequenas perturbações em ϕ_1 e ϕ_2 geram grandes variações de pesos.

Fontes de perturbação são erros de arredondamento e ruídos (erros de medidas físicas). Estes erros não podem ser eliminados pelo método Eliminação de Gauss e similares. Press et al (1988) recomenda usar Decomposição em Valores Singulares.

Figura 2.7. Combinação linear de vetores

B) Decomposição em Valores Singulares

Este método é baseado no seguinte teorema da álgebra: qualquer matriz $\Phi_{p \times m}$ pode ser decomposta no produto de uma matriz ortogonal³ $U_{p \times p}$, uma matriz diagonal $\Sigma_{p \times m}$ e uma matriz ortogonal $V_{m \times m}$. Ou seja:

$$\Phi = U \cdot \Sigma \cdot V^T \quad (2.4.3.3)$$

Onde,

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_s \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad (2.4.3.4)$$

Os valores σ_j são denominados de valores singulares. Considere o sistema linear:

$$Ax = b \quad (2.4.3.5)$$

Quando o número de linhas p é maior que o número de colunas m o sistema não tem solução, mas pode ser provado (Press et al, 1988) que o vetor x dado por:

$$x = A^+b \quad (2.4.3.6)$$

Onde,

$$A^+ = V \Sigma U^T \quad (2.4.3.7)$$

fornece a solução do sistema (2.4.3.5) no sentido dos mínimos quadrados. A matriz A^+ é denominada de matriz pseudo-inversa. Quando o sistema tem solução, a pseudo-inversa é igual a matriz inversa A^{-1} . A Decomposição em Valores Singulares também informa sobre o mal-condicionamento de uma matriz através do número de condição (razão entre o maior e o menor dos valores singulares σ_j). Altos números de condição indicam que a matriz A é mal-condicionada. Press et al (1988) recomenda zerar valores singulares que são muito pequenos. Os autores argumentam que isto elimina os vetores-coluna de A corrompidos por erros de arredondamento e ruídos.

Press et al (1988) também traz uma introdução à Decomposição em Valores Singulares e um programa de computador que implementa este método. Tal programa utiliza métodos matemáticos que estão além do escopo deste texto. Estes métodos são descritos em (Golub e Van Loan, 1989).

³ A inversa de uma matriz ortogonal A é a sua própria transposta: $A^T A = I$ onde I é a matriz identidade. Se A é uma matriz quadrada então $A^T A = A A^T = I$. Os vetores coluna de uma matriz ortogonal forma uma base de vetores ortonormais que são vetores ortogonais de comprimento unitário.

Na próxima seção, um método para melhorar a generalização da rede RBF será apresentado.

2.4.4 WEIGHT DECAY

A técnica *Weight decay* melhora a capacidade de generalização da rede pela adição de um termo à função de erro. Este termo tem por finalidade penalizar pesos com valores altos. Deste modo, a nova função a ser minimizada para determinar os pesos é dada por:

$$E = \sum_{i=1}^p \|\mathbf{t}^{(i)} - \mathbf{y}^{(i)}\|^2 + \lambda \sum_{j=1}^m w_j^2 \quad (2.4.4.1)$$

Onde $\lambda > 0$ é o parâmetro de regularização. O parâmetro λ controla o termo de penalidade. Tal termo encoraja os pesos a permanecerem com valores baixos. Quando os pesos podem assumir valores elevados, a rede RBF torna-se muito flexível tendendo a ajustar-se demasiadamente (*overfitting*) aos padrões de treinamento. O vetor peso $\mathbf{w}$ é obtido através do sistema linear:

$$(\Phi^T \Phi + \lambda \mathbf{I}_p) \mathbf{w} = \Phi^T \mathbf{t} \quad (2.4.4.2)$$

Onde $\mathbf{I}_p$ é uma matriz identidade $p \times p$. Este sistema foi obtido com a mesma metodologia utilizada na seção 2.4.1. Quando $\lambda = 0$, o sistema (2.4.4.2) torna-se igual ao sistema (2.4.1.17) obtido pelo método dos mínimos quadrados tradicional. *Weight decay* é conhecido na estatística como *ridge regression*. É também uma das formas mais simples de um método denominado **Regularização**. Formas de regularização mais complexas encontram-se em (Bishop, 1995).

Por aqui termina o treinamento híbrido da rede RBF. A topologia da rede que ainda não foi determinada será abordada na próxima seção.

2.5 CROSSVALIDATION

O treinamento híbrido não determina a topologia da rede (i.e. o número de unidades intermediárias). Por outro lado, o número de unidades intermediárias ou de centros influencia na capacidade de generalização da rede, isto é, influencia o desempenho sobre padrões novos não utilizados no treinamento. Portanto, é necessário um método para determinar o modelo de rede. Dois métodos muito utilizados em Redes Neurais são *hold out* e *crossvalidation*.

A) Hold out

Neste método o conjunto total de padrões disponíveis é dividido em três subconjuntos de padrões:

- Conjunto de treinamento
- Conjunto de validação
- Conjunto de teste

Em seguida, um conjunto de topologias são treinadas com o conjunto de treinamento. A topologia que apresenta melhor generalização é aquela que apresenta o melhor desempenho sobre um conjunto de padrões novos não utilizados no treinamento. Para isto utiliza-se o conjunto de validação. Porém, uma topologia pode ter sido selecionada por que ficou “por sorte” demasiadamente ajustada (i.e. *overffited*) ao conjunto de validação. Então, é necessário um terceiro conjunto de novos padrões para confirmar o desempenho da topologia selecionada. Para isto utiliza-se o conjunto de teste (Bishop, 1995).

B) Crossvalidation

Nos casos onde o número de padrões disponíveis é pouco e não é possível deixar padrões de fora do treinamento para criar o conjunto de validação, pode-se optar pelo método *crossvalidation* em que todos os padrões participam do treinamento (i.e. aqueles que no métodos *hold out* formariam os conjuntos de treinamento e validação).

O método *crossvalidation* divide o conjunto de treinamento em n subconjuntos de padrões disjuntos. Treina a rede usando $n-1$ dos subconjuntos e testa o desempenho com o subconjunto que ficou de fora. O processo se repete n vezes, sempre com um dos n subconjuntos sendo utilizado para testar o desempenho. O desempenho final é medido pela média. Por exemplo, considere o conjunto de treinamento S que é particionado em $n = 4$ subconjuntos disjuntos S_1, S_2, S_3 e S_4 . Então, 4 treinamentos e 4 testes são realizados usando as seguintes combinações:

- O 1º treinamento usa $S_2 \cup S_3 \cup S_4$ e o erro E_1 é medido sobre S_1
- O 2º treinamento usa $S_1 \cup S_3 \cup S_4$ e o erro E_2 é medido sobre S_2
- O 3º treinamento usa $S_1 \cup S_2 \cup S_4$ e o erro E_3 é medido sobre S_3
- O 4º treinamento usa $S_1 \cup S_2 \cup S_3$ e o erro E_4 é medido sobre S_4

A estimativa do erro é dado pela média:

$$E = \frac{1}{4} \sum_{i=1}^4 E_i$$

Em geral, na prática, o conjunto de treinamento é dividido em $n = 10$ subconjuntos. O caso extremo ocorre quando o conjunto de treinamento é dividido em $n = p$ partes onde p é o número de padrões. Neste caso, o método é chamado de *leave-one-out*. O *crossvalidation*⁴ requer longo tempo de processamento (principalmente, no caso do *leave-one-out*) devido ao lento treinamento de algumas Redes Neurais.

É importante frisar que o conjunto de teste não deve ser utilizado para selecionar o modelo da rede. No método *crossvalidation* (que não usa conjunto de validação), o modelo é selecionado somente utilizando o conjunto de treinamento. No método *hold out*, a seleção do modelo é feita exclusivamente pelo conjunto de validação. A conjunto de teste é usado uma única vez para fazer a estimativa “imparcial” do erro sobre o modelo selecionado. Mesmo que modelos com pior desempenho sobre o conjunto de validação tenham bom desempenho sobre o conjunto de teste, a estimativa final do erro deve ser baseado unicamente no modelo selecionado pelo conjunto de validação.

C) Métodos para modelos lineares

Os métodos *crossvalidation* e *hold out* são utilizados para estimar o erro de generalização para modelos não-lineares (e. g. redes MLP de várias camadas, apêndice A). Quando a rede RBF já possui os centros e larguras pré-determinados, o treinamento pode ser realizado por um método linear (e.g. algoritmo LMS e mínimos quadrados). Para tais redes RBF, que são modelos lineares, existem várias fórmulas para estimar o erro de generalização. Um exemplo é o *Generalized Crossvalidation* (GCV) que é dado por (Golub, Heath e Wahba, 1979; Zell et al, 1995)⁵:

$$\text{GCV} = \frac{m \text{ SSE}}{(p - m)^2} \quad (2.5.1)$$

Outros exemplos são o *Final Prediction Error* (FPE), o *Akaike's Information Criterion* (AIC) e o *Schwarz's Bayesian Information Criterion* (BIC). Existem também fórmulas para estimar o erro de generalização pelo *leave-one-out*, apresentado na seção anterior em modelos lineares (ver Orr, 1996). Um estudo sobre este tópico pode ser encontrado em (Moody, 1994).

A seção seguinte apresenta vários tipos de medidas de erro.

⁴ No contexto da Estatística, o método *hold out* é diferente do *crossvalidation*, pois no primeiro os padrões não são reutilizados (i.e. não há “crossing”) (Sarle, 1997). Contudo, na literatura das Redes Neurais os dois métodos têm sido confundidos.

⁵ A formula (2.5.1) é válida para redes RBF sem utilizar o método *weight decay*. Modificações em (2.5.1) podem ser feitas para incluir o *weight decay* (Orr, 1996).

2.6 FUNÇÕES DE ERRO

Várias funções de erro podem ser utilizadas para medir o desempenho de uma Rede Neural. Algumas são relacionados a seguir.

1) Soma dos erros quadráticos ou SSE (do inglês, *Sum of Squared Error*):

$$SSE = \sum_{i=1}^p \| \mathbf{t}^{(i)} - \mathbf{y}^{(i)} \|^2 \quad (2.6.1)$$

Onde, $\mathbf{y}^{(i)} = [y_1^{(i)}, y_2^{(i)}, \dots, y_c^{(i)}]$ é i -ésimo o vetor de saída,
 $\mathbf{t}^{(i)} = [t_1^{(i)}, t_2^{(i)}, \dots, t_c^{(i)}]$ é i -ésimo o vetor saída desejada,
 c é o número de unidades de saída,
 p é o número de padrões.

2) Erro quadrático médio ou MSE (do inglês, *Mean Squared Error*):

$$MSE = \frac{1}{p} \sum_{i=1}^p \| \mathbf{t}^{(i)} - \mathbf{y}^{(i)} \|^2 \quad (2.6.2)$$

3) Erro relativo médio ou MRE (do inglês, *Mean Relative Error*):

$$MRE = \frac{1}{p} \sum_{i=1}^p \left\| \frac{\mathbf{t}^{(i)} - \mathbf{y}^{(i)}}{\mathbf{t}^{(i)}} \right\| \quad (2.6.3)$$

4) Raiz do erro quadrático médio ou RMS (do inglês, *Root Mean Squared error*):

$$RMS = \sqrt{\frac{1}{p} \sum_{i=1}^p \| \mathbf{t}^{(i)} - \mathbf{y}^{(i)} \|^2} \quad (2.6.4)$$

5) RMS normalizado, também conhecido como erro de predição normalizada (do inglês, *Normalized Prediction Error*).

$$RMS \text{ normalizado} = \sqrt{\frac{\sum_{i=1}^p \| \mathbf{t}^{(i)} - \mathbf{y}^{(i)} \|^2}{\sum_{i=1}^p \| \bar{\mathbf{t}} - \mathbf{t}^{(i)} \|^2}} \quad (2.6.5)$$

Onde:

$$\bar{\mathbf{t}} = \frac{1}{p} \sum_{i=1}^p \mathbf{t}^{(i)} \quad (2.6.6)$$

O RMS normalizado é o RMS dividido pelo desvio padrão das saídas desejadas. Ele assume o valor 1 quando a rede prediz o conjunto de padrões “na média” (i.e. $y = \bar{t}$). E assume o valor zero quando prediz perfeitamente o conjunto de padrões. Em (Bishop, 1995, p.263) é recomendado medir o desempenho da rede sobre o conjunto de teste através do RMS normalizado por que tem a vantagem de ser independente do tamanho do conjunto de padrões.

Para problemas de classificação, não é adequado utilizar funções de erro para medir o desempenho da rede. O desempenho é medido em termos de percentagem de padrões classificados corretamente (ou incorretamente). Nestes casos a saída da rede assume valores lógicos (1 e 0) e há uma unidade de saída para cada classe. Por exemplo, uma rede com 3 unidades de saída terá para a primeira classe o vetor de saída (1,0,0), para a segunda classe (0,1,0) e para a terceira (0,0,1). Deve ser mencionado que o vetor de saída real y da rede não assume necessariamente valores binários, mas valores contínuos. Portanto, é necessário interpretar a saída da rede.

Existem vários métodos para esta interpretação. O mais simples método é o **winner-takes-all**. Aqui a saída da unidade com maior ativação designa a classe (i.e. será interpretado como 1 e unidades restantes como zero). Um método muito utilizado é a **regra 402040**. Aqui todas as unidades com saída menor que 0,4 serão interpretadas como zero. Todas as unidades com saída maior que 0,6 serão interpretadas como 1. E as unidades com saída no intervalo central [0,4, 0,6] serão interpretadas como "em dúvida".

2.7 COMPARAÇÃO ENTRE AS REDES RBF E MLP

Esta seção descreve uma comparação entre os modelos clássicos de redes RBF e redes MLP. Mais especificamente refere-se a redes RBF com função de ativação gaussiana usando treinamento híbrido e de redes MLP treinadas com o *backpropagation* (ver apêndice A). A seguir serão mostradas as semelhanças e diferenças entre as dois modelos.

- A principal diferença entre os dois modelos está no cálculo do valor de ativação da unidade intermediária. Na rede MLP, tal ativação é função do **produto escalar** entre vetor de entrada e o vetor peso das conexões que chegam a unidade intermediária. Na rede RBF, é função da **distância euclidiana** entre o mesmo vetor de entrada e o vetor peso das conexões que chegam a unidade intermediária da rede RBF (considerando que o componente μ_{ji} do vetor centro μ_j é o peso da conexão entre a unidade de entrada i e a unidade de intermediária j). Ver Tabela 2.2.
- A rede RBF possui apenas uma camada intermediária. A rede MLP possui uma ou mais camadas intermediárias.
- Na rede RBF, a camada intermediária é não-linear no sentido que realiza um mapeamento não linear do espaço de entrada para o espaço da camada

intermediária. Enquanto a camada de saída é linear, isto é, realiza um mapeamento linear do espaço da camada intermediária para o espaço de saída. Na rede MLP, ambas as camadas intermediárias e de saída são não-lineares.

- Na rede RBF, o valor de ativação da camada intermediária é constante sobre hipersferas no espaço de entrada. Na rede MLP, o valor de ativação da camada intermediária é constante sobre hiperplanos no espaço de entrada.
- A rede RBF possui representação local do vetor de entrada no espaço das unidades intermediárias (apenas algumas unidades intermediárias contribuem para o valor de saída. A rede MLP possui representação distribuída do vetor de entrada no espaço das unidades intermediárias (todas as unidades intermediárias contribuem para o valor de saída).
- Na rede RBF, os parâmetros da rede são determinados em dois estágios de treinamento: um não-supervisionado e um supervisionado. Na rede MLP, os parâmetros da rede são determinados ao mesmo tempo em um único treinamento supervisionado.
- A rede RBF divide o treinamento em dois rápidos estágios de treinamento. Evitando assim a lenta repropagação do erro que ocorre no treinamento das redes MLP.
- Existem justificativas teóricas que a rede MLP com suficiente número de unidades e parâmetros treinados apropriadamente é possível aproximar qualquer função continua com precisão arbitrária (Haykin, 1994; Hornik et al, 1989). As redes RBF também possui esta propriedade (Park e Sandberg, 1991) na qual é denominada de aproximação universal de função. Assim, sempre existe teoricamente uma rede RBF que equivale a uma rede MLP e vice-versa.
- A rede RBF constrói aproximadores locais de funções. Não possuem resposta significativa em regiões onde não há padrões de treinamento (é um pobre extrapolador). A rede MLP constrói aproximadores globais de funções. São capazes de generalizar em regiões onde não há padrões de treinamento.

Tabela 2.2. Função de ativação das redes MLP e RBF

Rede	$a_j =$ Ativação da unidade intermediária j
MLP	$a_j = f(\mathbf{x}^T \mathbf{w}_j)$ Onde, $\mathbf{w}_j = [w_{j1}, w_{j2}, \dots, w_{jm}]^T$ e w_{ji} é o peso da conexão entre unidade de entrada i e a unidade intermediária j
RBF	$a_j = \phi(\ \mathbf{x} - \mu_j\)$ Onde, $\mu_j = [\mu_{j1}, \mu_{j2}, \dots, \mu_{jm}]^T$ e μ_{ji} é interpretado como o peso da conexão entre unidade de entrada i e a unidade intermediária j

2.8 OLS

O algoritmo OLS (do inglês, *Orthogonal Least Square*) (Chen et al, 1991) determina automaticamente o número de centros ao mesmo tempo que treina os pesos da rede RBF. Os centros são selecionados de um conjunto de centros candidatos extraídos do conjunto de treinamento⁶. O algoritmo OLS é baseado no seguinte procedimento:

- Passo 1: Iniciar com uma rede com zero unidades intermediárias.
- Passo 2: Adicionar um novo centro a rede correspondente ao centro candidato que mais contribui na redução do SSE.
- Passo 3: Repetir o passo 2 até satisfazer o critério de parada.

Vale notar que tal procedimento é pouco eficiente: identificar em cada passo qual o centro que mais reduz o SSE de um conjunto de p centros candidatos requer a computação da pseudo-inversa p vezes. O OLS toma este procedimento mais eficiente empregando um método chamado de **Mínimos Quadrados Ortogonal**.

A) Mínimos quadrados ortogonal

O treinamento da rede RBF pode ser interpretado pelo seguinte modelo:

$$\mathbf{t} = \Phi \mathbf{w} + \mathbf{e} \quad (2.8.1)$$

O vetor peso $\mathbf{w}$ é obtido minimizando a soma dos erros quadráticos $SSE = \mathbf{e}^T \mathbf{e}$ (*sum of squared errors*) pelo método dos mínimos quadrados (ver seção 2.4.1). A matriz Φ pode ser decomposta em:

$$\Phi = \tilde{\Phi} \mathbf{U} \quad (2.8.2)$$

Onde, $\mathbf{U}$ é uma matriz triangular superior $m \times m$ dada por:

$$\mathbf{U} = \begin{bmatrix} 1 & a_{12} & a_{13} & \cdots & a_{1m} \\ 0 & 1 & a_{23} & \cdots & a_{2m} \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 & a_{m-1,m} \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \quad (2.8.3)$$

⁶ Para largos conjuntos de padrões é mais eficiente extrair um subconjunto dos padrões para ser centros candidatos. Este subconjunto pode ser extraído aleatoriamente ou utilizando uma técnica de aglomeração, como o k-médias, e escolhendo o ponto médio do aglomerado como centros candidatos.

$\tilde{\Phi}$ é uma matriz $p \times m$ com colunas ortogonais ($\tilde{\phi}_i^T \tilde{\phi}_j = 0$; $i \neq j$) que expandem o mesmo subespaço expandido pelos vetores não ortogonais (ϕ_j ; $j = 1 \dots m$). Portanto, (2.8.1) pode ser escrita como:

$$\mathbf{t} = \tilde{\Phi} \tilde{\mathbf{w}} + \mathbf{e} \quad (2.8.4)$$

Onde,

$$\tilde{\mathbf{w}} = \mathbf{U} \mathbf{w} \quad (2.8.5)$$

O vetor peso ortogonal $\tilde{\mathbf{w}}$, obtido pelo método dos mínimos quadrados, é dado por:

$$\tilde{\mathbf{w}} = (\tilde{\Phi}^T \tilde{\Phi})^{-1} \tilde{\Phi}^T \mathbf{t} \quad (2.8.6)$$

Vale observar que $\tilde{\Phi}^T \tilde{\Phi}$ é uma matriz diagonal, pois $\tilde{\phi}_i^T \tilde{\phi}_j = 0$ para $i \neq j$. A sua inversa é dada por:

$$(\tilde{\Phi}^T \tilde{\Phi})^{-1} = \begin{bmatrix} \frac{1}{\tilde{\phi}_1^T \tilde{\phi}_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\tilde{\phi}_2^T \tilde{\phi}_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\tilde{\phi}_m^T \tilde{\phi}_m} \end{bmatrix} \quad (2.8.7)$$

Os componentes do vetor $\tilde{\mathbf{w}}$ são dados por:

$$\tilde{w}_j = \frac{\mathbf{t}^T \tilde{\phi}_j}{\tilde{\phi}_j^T \tilde{\phi}_j}, \quad 1 \leq j \leq m. \quad (2.8.8)$$

Finalmente, o vetor peso $\mathbf{w}$ é determinado facilmente através do sistema triangular (2.8.4). Uma propriedade importante dos vetores $\tilde{\phi}_j$ ortogonais é que:

$$\|\mathbf{t}\|^2 = \sum_{j=1}^m \|\tilde{w}_j \tilde{\phi}_j\|^2 + \|\mathbf{e}\|^2 \quad (2.8.9)$$

Onde:

$$\|\tilde{w}_j \tilde{\phi}_j\|^2 = \frac{(\mathbf{t}^T \tilde{\phi}_j)^2}{\tilde{\phi}_j^T \tilde{\phi}_j}, \quad (2.8.10)$$

A Equação (2.8.10) representa a redução do SSE devido a uma única unidade intermediária j . Este resultados será útil mais tarde. A seguir será apresentado o clássico

Processo de Ortogonalização de Gram-Schmidt que é utilizado na decomposição de (2.8.2).

B) Processo de Ortogonalização de Gram-Schmidt

Considere o seguinte problema: obter uma base de vetores ortogonais $(\tilde{\phi}_1, \tilde{\phi}_2)$ a partir de uma base de vetores não-ortogonais. (ϕ_1, ϕ_2) . Tal problema é resolvido como segue: o primeiro vetor ortogonal é igual ao primeiro vetor não ortogonal:

$$\tilde{\phi}_1 = \phi_1 \quad (2.8.11)$$

O segundo vetor ortogonal é dado por:

$$\tilde{\phi}_2 = \phi_2 - a\tilde{\phi}_1 \quad (2.8.12)$$

Onde a é ajustado de tal modo que $\tilde{\phi}_1$ e $\tilde{\phi}_2$ sejam ortogonais. Ou seja:

$$\tilde{\phi}_2^T \tilde{\phi}_1 = 0 \quad (2.8.13)$$

Substituindo (2.8.12) em (2.8.13) obtém-se:

$$(\phi_2 - a\tilde{\phi}_1)^T \tilde{\phi}_1 = 0 \quad (2.8.14)$$

Então:

$$a = \frac{\phi_2^T \tilde{\phi}_1}{\tilde{\phi}_1^T \tilde{\phi}_1} \quad (2.8.15)$$

Este procedimento é ilustrado na Figura 2.8. Para o caso geral, com uma base de m vetores, tem-se:

$$\begin{aligned} \tilde{\phi}_1 &= \phi_1 \\ \tilde{\phi}_2 &= \phi_2 - a_{12}\tilde{\phi}_1 \\ \tilde{\phi}_3 &= \phi_3 - a_{13}\tilde{\phi}_1 - a_{23}\tilde{\phi}_2 \\ &\vdots \\ \tilde{\phi}_m &= \phi_m - \sum_{i=1}^{m-1} a_{mi}\tilde{\phi}_i \end{aligned} \quad (2.8.16)$$

Onde,

$$a_{ij} = \frac{\phi_j^T \tilde{\phi}_i}{\tilde{\phi}_i^T \tilde{\phi}_i} \quad (2.8.17)$$

O procedimento (2.8.16) chama-se Processo de Ortogonalização de Gram-Schmidt (que pode ser encontrado em livros de Álgebra Linear). Os coeficientes a_{ij} são usados na matriz U . Existem também outros procedimentos de ortogonalização mais eficientes como por exemplo a **Transformação de Householder** (Press et al, 1988). A seguir será apresentado o algoritmo OLS.

Figura 2.8. Ortogonalização de vetores

C) Treinamento

Para representar o conjunto de centros candidatos do algoritmo OLS é utilizado uma matriz F dada por:

$$F = [\mathbf{f}_1, \dots, \mathbf{f}_p]^T \quad (2.8.18)$$

Onde $\mathbf{f}_i$ o vetor coluna que representa o centro candidato fixado sobre o vetor de entrada $\mathbf{x}_i$:

$$\mathbf{f}_i = [\phi_i(\|\mathbf{x}_1 - \mathbf{x}_i\|), \phi_i(\|\mathbf{x}_2 - \mathbf{x}_i\|), \dots, \phi_i(\|\mathbf{x}_p - \mathbf{x}_i\|)]^T \quad (2.8.19)$$

Para adicionar o centro $m+1$ a uma rede RBF com m centros, o algoritmo ortogonaliza todos os vetores $\mathbf{f}_i$ em relação aos vetores $\{\tilde{\phi}_j; i = 1..m\}$ usando o Processo de Ortogonalização de Gram-Schmidt:

$$\tilde{\mathbf{f}}_i = \mathbf{f}_i - \sum_{j=1}^m \frac{\mathbf{f}_i^T \tilde{\phi}_j}{\tilde{\phi}_j^T \tilde{\phi}_j} \tilde{\phi}_j \quad (2.8.20)$$

Posteriormente, é selecionado o centro candidato q que mais reduz o SSE. A redução do SSE é calculado pela Equação (2.8.10). Portanto:

$$\tilde{\phi}_{m+1} = \tilde{\mathbf{f}}_q \text{ é tal que maximiza: } \frac{(\mathbf{t}^T \tilde{\mathbf{f}}_i)^2}{\tilde{\mathbf{f}}_i^T \tilde{\mathbf{f}}_i} \quad (2.8.20)$$

A utilização dos vetores ortogonais permite calcular a redução do SSE com pouco esforço computacional. Esta é a chave da eficiência do algoritmo OLS. O OLS é apresentado no Algoritmo 2.5. Contudo, se centros forem adicionados indefinidamente, a rede será levada ao *overfitting*. Logo, um critério de parada deve ser adotado.

Algoritmo 2.5. OLS

```

m ← 0
repita
  m ← m+1
  ΔEk ← 0
  para k=1 até p faça          /* apresentar todos os centros candidatos */
    se k ∉ {j1, j2, ..., jm-1} então /* exceto os previamente escolhidos */
       $\tilde{\mathbf{f}}_k \leftarrow \mathbf{f}_k - \sum_{i=1}^{m-1} \tilde{\phi}_i (\mathbf{f}_k^T \tilde{\phi}_i) / (\tilde{\phi}_i^T \tilde{\phi}_i)$  /* ortogonalização de Gram-Schmidt */
      e ←  $(\mathbf{t}^T \tilde{\mathbf{f}}_k)^2 / (\tilde{\mathbf{f}}_k^T \tilde{\mathbf{f}}_k)$ 
      se (e > ΔEk) então ΔEk ← e, jm ← k /* maximizar ΔEk */
    fim se
  fim para
   $\tilde{\phi}_m \leftarrow \tilde{\mathbf{f}}_{j_m}$  /* adicionar novo centro a rede */
até satisfazer o critério de parada

```

D) Critério de parada

Em (Chen et al, 1991) o critério de parada adotado foi quando o SSE atinge um valor limite mínimo pré-fixado. Melhores resultados têm sido obtidos com critérios de seleção de modelos, tais como o GCV mostrado na seção 2.5.

Nas primeiras etapas do algoritmo, tanto o SSE como o GCV tendem a diminuir. Quando a rede RBF atinge um certo número de centros, inicia-se a perda de generalização da rede e o GCV começa a aumentar, apesar do SSE continuar diminuindo. Neste ponto o algoritmo deve ser interrompido. O GCV é dado por:

$$\text{GCV} = \frac{p}{(p-m)^2} \left[\mathbf{t}^T \mathbf{t} - \sum_{j=1}^m \frac{(\mathbf{t}^T \tilde{\phi}_j)^2}{\tilde{\phi}_j^T \tilde{\phi}_j} \right] \quad (2.8.21)$$

Orr (1995) melhorou o algoritmo OLS usando *weight decay*. Um texto sobre o OLS e similares é encontrado em (Orr, 1996). Além do OLS e do treinamento híbrido, existem muitos outros algoritmos alternativos de treinamento para redes RBF.

2.9 OUTROS MODELOS

Este texto cobriu técnicas básicas para treinamento de redes neurais através do treinamento híbrido e do algoritmo OLS. Contudo, existem outros modelos alternativos de redes RBF. Por exemplo, o algoritmo RAN (do inglês, *Resource Allocating Network*) (Platt, 1991) realiza tipo um treinamento denominado treinamento sequencial na qual os padrões de treinamento são apresentados uma única vez a rede e descartados. Os centros são alocados automaticamente na medida que os padrões são apresentados. Se a rede tem uma resposta incorreta para um novo padrão apresentado então um novo centro é alocado para corrigir a resposta. Se a rede responde corretamente ao novo padrão então os pesos são ajustados pelo algoritmo LMS.

O algoritmo RAN-EKF (Kadirkamanathan e Niranjan, 1993) é um aperfeiçoamento do RAN na qual substitui o LMS pelo *Extended Kalmon Filter* (EKF). Outras melhorias no RAN é encontrado em (Yingwei et al, 1997) na qual inclui uma estratégia de poda (*pruning*) para diminuir o tamanho da rede.

Um recente Algoritmo denominado DDA (do inglês, *Dynamic-Decay Adjustment*) (Berthold e Diamond, 1995) tem obtido para problemas de classificação desempenho semelhante as redes MLP porém com tempo de treinamento significativamente menor (Zell et al, 1995). Vogt (1993) usou *Learning Vector Quantization* (LVQ) para determinar os centros da rede. Outros modelos de rede RBF podem ser encontrados em (Hartman e Keeler, 1991; Musavi et al, 1992; Wettschereck e Dietterich, 1992; Roy et al, 1995).

2.10 RESUMO

O treinamento híbrido da rede RBF combina um treinamento não-supervisionado com um supervisionado. Tal treinamento é muito rápido em termos computacionais. O treinamento não-supervisionado (e.g. k-médias) seleciona os centros. O treinamento supervisionado determina os pesos por um método dos mínimos quadrados (e.g. o algoritmo LMS ou a matriz pseudoinversa). A capacidade de generalização da rede pode ser melhorada usando técnicas de regularização, na qual o método *Weight Decay* é uma das mais simples e utilizadas.

A topologia da rede não é determinada pelo método híbrido. Em geral, a topologia é obtida por tentativa e erro com auxílio de técnicas como *crossvalidation* e *hold out*. O método OLS determina a topologia da rede automaticamente adicionando, passo a passo,

unidades intermediárias a rede (com centros extraídos do conjunto de treinamento). O OLS é um método de busca local que utiliza uma heurística de selecionar, em cada passo, o centro candidato que mais reduz a soma dos erros quadráticos da rede.

No próximo capítulo serão investigados os conceitos básicos dos Algoritmos Genéticos. Com ele completa a introdução inicial para que se possa chegar ao principal objetivo desta Dissertação, que é a otimização de Redes Neurais utilizando Algoritmos Genéticos.

3

ALGORITMOS GENÉTICOS

"Quem despreza o alfabeto não atinge a ciência"
(Espírito Emmanuel)

*"Quando a lição oferecer dificuldades à tua mente,
compelindo-te à desistência do progresso individual,
aplica-te ao problema ou ao ensinamento mais um pouco
e a solução será clara resposta à nossa expectativa"*
(Espírito André Luiz)

"Aceita o fracasso por base do recomeço"
(Espírito Emmanuel)

3.1 INTRODUÇÃO

Os Algoritmos Genéticos são métodos de busca e otimização. Foram introduzidos por John Holland da década de 70 (Holland, 1975). Estes algoritmos foram inspirados nos mecanismos de evolução de populações de seres vivos, baseados no princípio da seleção natural e sobrevivência do mais adaptado declarado em 1859 pelo naturalista e fisiologista inglês Charles Darwin em seu livro "A Origem das Espécies".

Sinteticamente, a teoria da evolução diz que o meio ambiente seleciona, em cada geração, os seres vivos mais aptos de uma população. Em geral, somente os mais aptos conseguem reproduzir, uma vez que os menos adaptados geralmente são eliminados antes de se tornarem reprodutivos. Durante a reprodução, ocorrem fenômenos como mutação e crossover (recombinação), entre outros, que atuam sobre o material genético armazenado nos cromossomos. Estes fenômenos mantêm grande variabilidade de seres vivos na população. Sobre esta população diversificada age a seleção natural só permitindo a sobrevivência dos seres mais adaptados.

Os Algoritmos Genéticos processa uma população de **cadeias de bits** (ou alguma outra estrutura de dados). As cadeias de bits codificam em notação binária o conjunto de **parâmetros** do problema que será otimizado. Portanto, cada cadeia de bits representa uma solução para o problema. O conjunto de todas as configurações que a cadeia de bits pode assumir é denominado **espaço de busca**. Por exemplo, se a cadeia de bits codifica n parâmetros a ser otimizados, então a espaço de busca é um espaço com n dimensões.

Um Algoritmo Genético é mostrado no Algoritmo 3.1. Tipicamente, Algoritmo Genético inicia com uma população $S(0)$ constituído de N cadeias de bits aleatórias. Em seguida, cada cadeia de bits é avaliada. O resultado da avaliação é um valor denominado **aptidão** que mede a utilidade da solução representada na cadeia de bits de resolver um dado problema.

Através de um processo de **seleção**, as melhores cadeias de bits (i.e. as boas soluções) são selecionadas de $S(0)$ e modificadas através de operadores genéticos tais como **crossover** e **mutação**. A nova geração de N cadeias de bits modificadas irão constituir nova população $S(1)$. O processo é repetido, e uma sequência de populações $S(2), S(3), \dots, S(t-1), S(t)$ são geradas.

A combinação do processo de seleção com os operadores crossover e mutação guiam o algoritmo ao encontro das melhores cadeias de bits, ao mesmo tempo que explora novas áreas do espaço de busca. Finalmente, o algoritmo converge para uma população $S(t)$ na qual contém cadeias de bits que representam o **máximo global** (solução ótima) do problema. O máximo global representa o maior ponto de máximo de uma função. Ver Figura 1.2. Os demais pontos de máximo são chamados **máximos locais**. Deve ser enfatizado que no Algoritmo Genético não é garantido encontrar a solução ótima.

Segundo Beasley et al (1993a), a área de maior interesse para os Algoritmo Genéticos é aquela que envolve problemas de difícil otimização, que se caracterizam pela inexistência de técnicas para resolvê-los. Quando já existe tal técnica, geralmente ela é mais rápida e mais acurada que os Algoritmos Genéticos. Algumas aplicações bem sucedidas de Algoritmos Genéticos em problemas de difícil otimização são:

- Problema do caixeiro viajante
- Processamento de Imagens
- Layout de circuitos integrados VLSI
- Estruturas de Pontes
- Topologias de Redes Neurais

Na literatura existem bons livros sobre Algoritmos Genéticos tais como (Davis, 1991; Goldberg, 1989; Michalewicz, 1994; Mitchell, 1996) e vários textos introdutórios

(Beasley et al, 1993a, 1993b; Heitkoetter e Beasley, 1998; Whitley, 1993). Nas próximas seções, será mostrado uma aplicação dos Algoritmos Genéticos para otimização de função.

Algoritmo 3.1. Algoritmo Genético

```

 $t \leftarrow 0$ 
inicializar  $S(t)$ 
avaliar  $S(t)$ 
enquanto não satisfazer o critério de parada faça
     $t \leftarrow t+1$ 
     $S(t) \leftarrow$  selecionar  $S(t-1)$ 
    crossover sobre  $S(t)$ 
    mutação sobre  $S(t)$ 
    avaliar  $S(t)$ 
fim para

```

Figura 3.1. Máximos global e local de uma função

3.1.1 OTIMIZAÇÃO DE FUNÇÃO

Considere o seguinte problema de otimização de função:

Problema 3.1 Encontrar o máximo global da função mostrada na Figura 3.1 dada pela Equação (3.1.1.1) no intervalo $-100 \leq x \leq 100$ e $-100 \leq y \leq 100$ (Davis, 1991).

$$f(x, y) = 0,5 - \frac{\left(\sin \sqrt{x^2 + y^2} \right)^2 - 0,5}{\left(1,0 + 0,001(x^2 + y^2) \right)^2} \quad (3.1.1.1)$$

Um das características dessa função é o grande número de pontos de máximo cujo máximo global ocorre em $x = 0$ e $y = 0$. O Problema 3.1 pode não ser resolvido satisfatoriamente por métodos de busca local como, por exemplo, o método do gradiente, como será mostrado na seção 3.2. Tais métodos não conseguem distinguir qual é máximo global entre os diversos máximos locais. A seguir será descrito o cromossomo utilizado neste problema.

Figura 3.2 Função com vários pontos de máximo local.

3.1.2 O CROMOSSOMO

Uma possível solução para um problema 3.1 é representada por dois parâmetros x e y . Estes parâmetros são agrupados para formar uma cadeia de valores conhecida como cromossomo. Holland (1975) utilizou originalmente números binários para representar a cadeia de valores. Um exemplo de cromossomo é a seguinte cadeia de bits:

$$s_1 = 01101001001001101000001000111000100001110010$$

Na cadeia s_1 de 44 bits⁷ está codificada uma possível solução para o Problema 3.1. A decodificação desta solução se dá como segue:

- 1) Dividir a cadeia de bits s_1 em duas partes:

$$0110100100100110100000 \text{ e } 1000111000100001110010$$

- 2) Mudar a notação das duas partes da base 2 para a base 10:

⁷ Aumentando o tamanho da cadeia de bits aumenta-se a precisão da solução.

1722784 e 2328690.

3) Converter estes resultados para o intervalo $[-100, 100]$:

$$x_1 = \frac{1722784}{2^{22} - 1} [100 - (-100)] + 100 = -17,851$$

$$y_1 = \frac{2328690}{2^{22} - 1} [100 - (-100)] + 100 = 11,041$$

4) s_1 representa a solução $(x_1, y_1) = (-17,851; 11,041)$

Para medir se a solução da cadeia s_1 é boa ou não, utiliza-se a **função de avaliação**.

3.1.3 A FUNÇÃO DE AVALIAÇÃO

A Aptidão de um cromossomo é baseada da função de avaliação ou **função objetivo**. A função de avaliação do Problema 3.1 pode ser a própria função fornecida pela Equação (3.1.1.1). Neste problema, o valor da aptidão pode ser igual ao valor da função de avaliação (esta igualdade nem sempre é interessante como visto mais tarde). Assim, a aptidão f_1 da cadeia de bits s_1 da seção anterior é dada por:

$$f_1 = 0,5 - \frac{\left(\sin \sqrt{-17,851^2 + 11,041^2}\right)^2 - 0,5}{\left(1,0 + 0,001(-17,851^2 + 11,041^2)\right)^2} = 0,580$$

O Algoritmo Genético, pela sua natureza, sempre procura maximizar a função de avaliação (busca cromossomos de alta aptidão). Quando o problema é de minimização de uma função $f(x)$ basta multiplicar a função por -1 , isto é, $g(x) = -f(x)$ para convertê-lo em um problema de maximização. As vezes, é necessário adicionar uma constante c , $g(x) = c - f(x)$, como por exemplo $c = \max(g(x))$ para garantir aptidão sempre positiva na qual é requerida no algoritmo Roda da Roleta mostrado na próxima seção.

3.1.4 SELEÇÃO

Um Algoritmo Genético começa com uma população inicial de N cadeias de bits geradas aleatoriamente (assumindo que não se tem nenhum conhecimento prévio sobre a região do espaço de busca onde se encontra a solução do problema). Em cada geração, pares de cromossomos são selecionados para gerar pares de cromossomos filhos que são variantes dos pais através dos operadores genéticos crossover e mutação.

Os cromossomos são selecionados para crossover e mutação com probabilidade proporcional a sua aptidão. Um modo simples e intuitivo de fazer isto é através de uma Roda de Roleta. Como exemplo, considere a população de 5 cromossomos ilustrada na Figura 3.3 e suas respectivas aptidões e probabilidades de seleção apresentados na Tabela 3.1. A probabilidade de seleção de um cromossomo i com aptidão f_i é dada por:

$$\frac{f_i}{\sum_{i=1}^N f_i} \quad (3.1.4.1)$$

Onde N é o tamanho da população (no caso $N=5$). Os cromossomos são selecionados usando a roleta mostrada na Figura 3.4, onde o tamanho de cada fatia é proporcional à aptidão do cromossomo. Quanto maior a fatia, tanto maior a chance do cromossomo ser selecionado. O número esperado de cópias de um cromossomo i que vai para crossover e mutação é dado por:

$$\frac{1}{N} \sum_{i=1}^N f_i \quad (3.1.4.2)$$

O algoritmo da Roda da Roleta é descrito no Algoritmo 3.2. A seguir os operadores crossover e mutação serão apresentados.

A	100000000000010001100011111111111001101110
B	011100000011101111101011101110000001100100
C	100010110110010111100010001001000111010000
D	1001101011011110101011011111111110110100100
E	10001011011111111010000111011111100100001011

Figura 3.3. População de 5 cromossomos

Tabela 3.1. Aptidões dos cromossomos

Cromossomo	Aptidão	Nº esperado de filhos	Probabilidade de seleção
A	0,999588	1,7	33,5 %
B	0,826877	1,39	27,7 %
C	0,655533	1,10	22,0 %
D	0,400148	0,67	13,4 %
E	0,102002	0,17	3,4 %

Figura 3.4. Roda da Roleta

Algoritmo 3.2. Roda da Roleta

```

total ←  $\sum_{i=1}^N f_i$       /* a soma das aptidões de todos os cromossomos da população */
rand ← randômico(0, total)
totalparcial ← 0
i ← 0
repetir
    i ← i+1
    totalparcial ← totalparcial +  $f_i$ 
até totalparcial ≥ rand
retornar o cromossomo  $s_i$ 

```

3.1.5 CROSSOVER E MUTAÇÃO

A) Crossover

No operador de crossover tradicional, os cromossomos dos indivíduos selecionados terão suas cadeias de bits cortadas em uma posição aleatória produzindo duas cabeças e duas caudas. As caudas são trocadas gerando dois novos cromossomos. Um exemplo é mostrado na Figura 3.5.

O crossover é aplicado com uma dada probabilidade a cada par de indivíduos selecionados. Na prática, esta probabilidade, denominada de **taxa de crossover**, varia entre 0.6 e 0.99. Isto pode ser implementado utilizando um gerador de números pseudo-aleatórios para produzir valores entre 0 e 1. O crossover só é aplicado se o valor aleatório gerado for menor que a taxa de crossover.

PAI 1	=	(100010010110010110100101 01110100010101110000),	Aptidão = 0,654459
PAI 2	=	(111000100000111110001100 10010111001100100100),	Aptidão = 0,502389
FILHO 1	=	(100010010110010110100101 10010111001100100100),	Aptidão = 0,703362
FILHO 2	=	(111000100000111110001100 01110100010101110000),	Aptidão = 0,498025

Figura 3.5. Operador genético crossover.

B) Mutação

O operador de mutação é geralmente aplicado após crossover em a cada bit dos dois cromossomos filhos. A mutação inverte os valores dos bits. Cada bit tem uma probabilidade pequena (por exemplo, **taxa de mutação** igual 0,001) de ser invertido. Na Figura 3.6 é mostrado um exemplo no qual a mutação inverte o 9º e o 41º bit (que passaram no teste de probabilidade) de um cromossomo.

ANTES	(11001100 0 01100000000010001000000000000 1 001),	Aptidão = 0,498670
DEPOIS	(11001100 1 01100000000010001000000000000 0 001),	Aptidão = 0,500114

Figura 3.6. Operador genético mutação

3.1.6 ELITISMO

Após aplicar o crossover e a mutação nos cromossomos selecionados pela Roda da Roleta, surge a uma nova população de filhos que substituirá completamente a antiga população de pais na geração subsequente. Com esta substituição geralmente perde-se do melhor cromossomo da população antiga. Este problema é evitado inserindo um cópia do melhor cromossomo na população de filhos. Esta técnica simples é denominada **Elitismo** na qual visa conservar a melhor solução, encontrada até então, na população (DeJong, 1975).

A evolução das populações é repetida geralmente até um número fixo de gerações, onde se espera encontrar a solução desejada. A seguir será apresentada a terminologia utiliza na literatura de Algoritmos Genéticos.

3.1.7 TERMINOLOGIA

A lista abaixo descreve os principais termos utilizados na literatura dos Algoritmos Genéticos.

- **Genoma, Genótipo e Cromossomo:** têm significados diferentes na biologia, mas no Algoritmo Genético são frequentemente usados para designar a mesma coisa: a estrutura de dados que codifica a solução de um problema.

- **Fenótipo:** representa o objeto, estrutura ou organismo construído a partir das informações do genótipo. É o cromossomo decodificado. Por exemplo, considere que o cromossomo codifica parâmetros como as dimensões das vigas em um projeto de construção de um edifício, ou as conexões e pesos de uma topologia de Rede Neural. O Fenótipo seria o edifício construído ou a Rede Neural.
- **Gene:** representa uma subseção do cromossomo. Geralmente um gene codifica um parâmetro do problema.
- **Alelos:** são os valores que o gene pode assumir. Como exemplo, o alelo de um gene representando a cor de um objeto pode ser azul, preto, verde, etc..
- **Função de avaliação:** é também conhecida como função objetivo. Mede a utilidade da solução representada no cromossomo. Em problemas práticos, a função de avaliação pode ser complexa. No exemplo do edifício do item fenótipo, a função de avaliação pode envolver muitos fatores, como a carga máxima suportada pelas estruturas do edifício, o custo e o tempo de construção.
- **Função de aptidão:** serve para transformar o valor da função de avaliação a fim de obter a aptidão de um cromossomo. Às vezes esta transformação é um escalamento da função envolvendo a aptidão de todos os indivíduos da população. Contudo, em alguns textos encontrados na literatura a função de aptidão é usada com o mesmo significado de função de avaliação.
- **Epistasia:** interação entre genes do cromossomo, isto é, quando um valor de gene influencia sobre o valor de outro gene.

A seguir é realizada uma comparação dos Algoritmos Genéticos com outras técnicas de busca e otimização.

3.2 COMPARAÇÃO COM OUTROS MÉTODOS

Nesta seção será esboçado uma comparação com alguns métodos gerais de busca e otimização.

3.2.1 GERAR-E-TESTAR

O algoritmo Gerar-e-Testar é um método de força bruta. Consiste em gerar uma solução aleatoriamente ou sistematicamente e então testar a solução. Se a solução for gerada

sistematicamente, ele é certo encontrar a solução. Porém se o espaço de busca for muito grande a busca poderá tomar um tempo proibitivamente longo.

3.2.2 SUBIDA DO MONTE

O algoritmo de Subida do Monte (*hill climbing*) investiga os pontos adjacentes do espaço de busca e move-se na direção que mais aumenta o valor da função objetivo. Como exemplo, considere o problema de encontrar o máximo global da função ilustrada na Figura 3.7.

A Subida do Monte geralmente começa em um ponto arbitrário. Se iniciar no ponto A ele moverá na direção da encosta ascendente mais inclinada que é na direção de B. Chegando no máximo local B não haverá mais subida e o algoritmo para. Já se o início for no ponto C então o algoritmo terminará no máximo global D.

Portanto, quando a função tem muitos máximos locais, o método de Subida do Monte terá dificuldades em localizar qual “monte” possui o máximo global. Contudo, a Subida do Monte é geralmente rápido e é um bom método para a funções com um único ponto de máximo.

Os conhecidos métodos de gradiente utilizam este algoritmo. Nesses métodos a direção da encosta mais inclinada é informada pelo gradiente que, por sua vez, é obtida da derivada da função.

Figura 3.7. A Subida do Monte

3.2.3 EXPLORATION E EXPLOITATION

Qualquer algoritmo de otimização eficiente deve usar estas duas técnicas para encontrar o máximo global da função (Beasley, 1993a). *Exploration* é a exploração de pontos inteiramente novos no espaço de busca. A algoritmo Gerar-e-testar utiliza esta técnica durante a busca. *Exploitation* consiste em utilizar a informação de pontos anteriormente visitados para encontrar melhores pontos. Esta técnica é utilizada pelo algoritmo Subida do Monte.

O Algoritmo Genético combina estas duas técnicas para encontrar máximo global da função. O crossover realiza um tipo de *Exploitation* pois combina as informações de dois cromossomos para tentar gerar um cromossomo com maior aptidão. Deste modo, o operador crossover trabalha semelhante ao método de Subida do Monte.

Se a taxa de mutação for muito baixa, poucas soluções novas serão exploradas e a *Exploration* é reduzida uma vez que o crossover não gera novos genes, apenas combina os que já existem. E o algoritmo genético torna-se um simples método de Subida do Monte. O aumento da taxa de mutação tem o efeito de aumentar a *Exploration*. É incentivada a criação de novos genes, melhorando a diversidade de genes e permitindo a exploração de novas áreas do espaço de busca (*Exploration*).

A **pressão da seleção** (razão entre aptidão máxima da população e a aptidão média) também influencia na quantidade de *Exploitation* e *Exploration*. Quando a pressão da seleção está alta favorece o *Exploitation*. Quando a pressão da seleção está baixa favorece o *Exploration*. Quando a pressão de seleção é extremamente baixa (todas as aptidão iguais) e não há crossover, o busca torna-se aleatória (muito *Exploration* sem nenhum *Exploitation*). A pressão de seleção pode ser controlada de várias maneiras, como será visto mais tarde. Contudo, na prática é difícil conhecer o equilíbrio ideal entre *Exploitation* e *Exploration*.

A seguir será descrito o algoritmo Recozimento Simulado que, assim como os Algoritmos Genéticos, permite encontrar o máximo global de uma função.

3.2.4 RECOZIMENTO SIMULADO

O **Recozimento Simulado** (do inglês *Simulated Annealing*) (Kirkpatrick et al, 1983) é uma técnica também inspirada na natureza. A inspiração vem do processo físico do recozimento de metais. Metais são fundidos a uma alta temperatura assumindo um estado de alta energia. Após fundidos, os metais são resfriados lentamente para que o sistema atinja o nível mínimo ótimo (mínimo global) de energia correspondente a cristais perfeitos. Se o resfriamento for rápido não há formação de cristais perfeitos por que o sistema cai em níveis mínimos sub-ótimos (mínimo local) de energia.

Durante o resfriamento, o sistema nem sempre move-se para um estado de menor energia. Existe uma probabilidade do sistema mover para um estado de maior energia. Esta probabilidade é dada por:

$$p = \exp\left(-\frac{\Delta E}{kT}\right) \quad (3.2.4.1)$$

Onde, ΔE é a diferença de energia entre o estado de maior e o de menor energia, T é a temperatura do sistema e k é a constante de Boltzmann. Este movimento para um estado de maior energia permite ao sistema escapar de um mínimo local para buscar melhores estados de energia. Quando a temperatura está baixa é improvável que o sistema mova para

um estado de maior energia. Porém, nesta fase é esperado que o sistema já tenha encontrado a “descida no vale” do mínimo ótimo de energia.

O algoritmo Recozimento Simulado utiliza esta idéia para minimizar a função objetivo $E(x)$ que representa, em analogia, o nível de energia do sistema. O algoritmo inicia em um ponto arbitrário x_i . Posteriormente, é feito um movimento aleatório para um outro ponto x_{i+1} . Se $E(x_{i+1}) < E(x_i)$ então o movimento é aceito. Caso contrário, o movimento é aceito com probabilidade:

$$p = \exp\left(-\frac{E(x_{i+1}) - E(x_i)}{T}\right) \quad (3.2.4.2)$$

A Equação (3.2.4.2) é semelhante à Equação (3.2.4.1). Contudo, a constante k de Boltzmann é absorvida pelo parâmetro T . Achar o mínimo local depende **Cronograma do Recozimento** (do inglês *annealing schedule*). A temperatura T deve começar alta e depois ir diminuindo aos poucos. Quando a temperatura está alta, grandes áreas do espaço de busca são exploradas, uma vez que quase todo movimento é aceito.

Depois, quanto a temperatura T baixa, o algoritmo cai em algum vale. Se o resfriamento for lento o suficiente, o algoritmo cairá no vale do mínimo global. Se for rápido poderá cair em um vale de mínimo local. Saber quão lento deve ser o resfriamento vai depender do problema. O Algoritmo Recozimento Simulado é apresentado no Algoritmo 3.3.

É possível relacionar algumas semelhanças e diferenças entre Algoritmos Genéticos e Recozimento Simulado:

- Recozimento Simulado trabalha com um único ponto no espaço de busca
- Algoritmo Genético trabalha com muitos pontos no espaço de busca
- Ambos precisam de operadores específicos para gerar soluções a partir de soluções anteriores.
- Recozimento Simulado é mais simples que os Algoritmos Genéticos.
- Ambos são capazes de encontrar soluções globalmente ótimas em espaços de busca complexos (sem derivadas em todos os pontos, descontínuos etc..).
- Ambos têm sido aplicados com sucesso em problemas de difícil otimização (ex. o problema do caixeiro viajante).

Contudo, não é possível afirmar qual dos dois Algoritmos é o melhor. Os dois ainda são temas de pesquisa e estão sendo aperfeiçoados. A seguir serão apresentados diversos tópicos relacionados com o mecanismo de seleção utilizado pelos Algoritmos Genéticos.

Algoritmo 3.3. Recozimento Simulado (*Simulated Annealing*)

```
escolher um ponto arbitrário  $x_0$ 
inicializar  $T$  com uma temperatura suficientemente alta
 $i \leftarrow 0$ 
enquanto ( $T > 0$ ) faça
    realizar um movimento aleatório de  $x_i$  para  $x_{i+1}$ 
     $\Delta E \leftarrow E(x_{i+1}) - E(x_i)$ 
    se ( $\Delta E < 0$ ) então
         $i \leftarrow i+1$ 
    senão
         $p = \exp(-\Delta E/T)$ 
        se ( $p < \text{randomico}(0,1)$ ) então  $i \leftarrow i+1$ 
    fim se
    se ( $E$  não foi reduzido significativamente em várias interações) então
        abaixar a temperatura  $T$ 
    fim se
fim enquanto
```

3.3 SELEÇÃO

Nesta seção serão abordados diversos tópicos relacionados com o processo de seleção de cromossomos. Serão apresentados alguns problemas, entre eles o problema de convergência prematura, e como a seleção pode resolvê-los através de técnicas como escalamento da aptidão, *ranking* e seleção por torneio. Estas técnicas também permitem controlar a pressão da seleção. Os tipos de seleção e várias técnicas alternativas serão descritos.

3.3.1 CONVERGÊNCIA PREMATURA

A convergência prematura ocorre quando surgem indivíduos de alta aptidão (mas não com aptidão ótima) e os indivíduos realmente ótimos ainda não estão presentes na população. Tais indivíduos (denominados de **superindivíduos**) geram excessivo número de filhos que dominam a população uma vez que a mesma é finita. Consequentemente, o algoritmo converge para uma solução sub-ótima (máximo local). Ver Figura 3.9.

Combate-se a convergência prematura limitando o número de filhos por indivíduo através do escalamento da aptidão, *ranking* e outras técnicas descritas mais adiante. Manter

a diversidade dos indivíduos na população também combate a convergência prematura, visto que a perda de diversidade causa a convergência prematura. O aumento da taxa de mutação melhora a diversidade (mais genes são criados). Evitar inserir filhos duplicadas na população também melhora a diversidade. A seguir um outro conhecido grupo de problemas será descrito.

Figura 3.9. Convergência prematura

3.3.2 PROBLEMAS NO INTERVALO DA APTIDÃO

Uma pequena alteração na função de avaliação (3.1.1.1) gera a Equação (3.3.2.1) e o seguinte problema: a população da Figura 3.3 passa a ter as aptidões e probabilidades de seleção mostradas na Tabela 3.2. A probabilidade de seleção torna-se praticamente a mesma em todos indivíduos. Assim, a seleção torna-se aleatória, sem favorecer os melhores indivíduos. Problema semelhante ocorre nas gerações finais do algoritmo. Neste ponto grande parte da população convergiu e os indivíduos têm aptidão praticamente iguais. O algoritmo tem dificuldade de melhorar a solução e converge lentamente.

$$f(x, y) = 999.5 - \frac{(\sin \sqrt{x^2 + y^2})^2 - 0.5}{(1.0 + 0.001(x^2 + y^2))^2} \quad (3.3.2.1)$$

Este problema pode ser solucionado expandindo o intervalo de aptidão através das mesmas técnicas usadas para combater a convergência prematura: *ranking*, seleção por torneio ou escalamento da aptidão que serão apresentados na seções seguintes.

Tabela 3.2. Intervalo estreito de aptidões.

Cromossomo	Aptidão	Probabilidade de seleção
A	999,999588	20,008 %
B	999,826877	20,005 %
C	999,655533	20,001 %
D	999,400148	19,996 %
E	999,102002	19,990 %

3.3.3 ESCALAMENTO DA APTIDÃO E RANKING

Os métodos mais utilizados de remapeamento da aptidão (na qual combate a convergência prematura e resolve problemas de intervalo de aptidão) são:

- Escalamento da aptidão
- *Ranking*

A) Escalamento da aptidão

O escalamento de uma aptidão f é realizado por meio de uma função linear do tipo: $f' = af + b$ onde f' é aptidão convertida para a nova escala. Ver Figura 3.10. Os coeficientes a e b são geralmente determinados de modo que o número máximo de filhos por indivíduo seja fixado em C , onde tipicamente $C = 2$. Um problema no escalamento é a possibilidade de aparecerem aptidões negativas indesejadas.

O procedimento de escalamento encontrado em (Goldberg, 1989) é mostrado na Figura 3.11. Os coeficientes a e b foram obtidos pela condição descrita anteriormente e pela condição adicional que $\bar{f}' = \bar{f}$, onde $\bar{f}$ é a aptidão média da população. Quando não é possível escalar sem gerar aptidões negativas, então a e b são determinados impondo $f'_{min} = 0$.

Figura 3.10. Gráfico de escalamento da aptidão

```

 $\bar{f} = \frac{1}{N} \sum_{i=1}^N f_i$  /* aptidão média */
Se  $f_{\min} > \frac{C\bar{f} - f_{\max}}{C-1}$  então /* testa se ocorre aptidão negativa */
 $f' = \left[ \frac{(C-1)\bar{f}}{f_{\max} - \bar{f}} \right] f + \left[ \frac{\bar{f}(f_{\max} - C\bar{f})}{f_{\max} - \bar{f}} \right]$ 
senão
 $f = \left[ \frac{\bar{f}}{\bar{f} - f_{\min}} \right] f - \left[ \frac{\bar{f} f_{\min}}{\bar{f} - f_{\min}} \right]$ 
fim se

```

Figura 3.11. Procedimento para escalamento da aptidão

B) Ranking

No *ranking*, a população é ordenada segundo o valor da função de avaliação. A aptidão de um indivíduo é definida de acordo com sua posição no ordenamento e não pelo valor da função de avaliação. Na Tabela 3.3 é mostrado uma população 5 indivíduos com aptidão obtida por um **ranking linear** dado por:

$$f = s - \frac{2(r-1)(s-1)}{N-1} \quad (3.3.3.1)$$

Onde, r é o *rank* do indivíduo (i.e. sua posição na população ordenada), f é sua aptidão, N é o tamanho da população, e s é a pressão da seleção (aptidão_máxima / aptidão_média). Na Tabela 3.3. foi utilizado $s = 2$. A Figura 3.12 é mostrado o controle da pressão de seleção usando *ranking*. Na Figura 3.12.a, a alta pressão de seleção favorece fortemente aos melhores indivíduos, direcionando a busca ao encontro das melhores soluções encontradas até então (*Exploitation*). Na Figura 3.12.b, a baixa pressão de seleção favorece um pouco mais os indivíduos de baixa aptidão, direcionando a busca para explorar mais regiões do espaço de busca (*Exploration*).

Alternativamente, um **ranking exponencial** pode ser utilizado, na qual é dado por:

$$f = q(q-1)^{r-1} \quad (3.3.3.2)$$

Onde $q \in [0,1]$. A seguir será apresentado a seleção por torneio na qual não é proporcional a aptidão.

Tabela 3.3. Ranking linear

Cromossomo	Função objetivo	rank	aptidão
A	999,999588	1	2,0
B	999,826877	2	1,5
C	999,655533	3	1,0
D	999,400148	4	0,5
E	999,102002	5	0,0

Figura 3.12. Ranking e a pressão da seleção

3.3.4 SELEÇÃO POR TORNEIO

Na seleção por torneio cada indivíduo é selecionado para crossover e mutação do seguinte modo: são escolhidos aleatoriamente (com probabilidades iguais) n indivíduos da população e o melhor dentre estes indivíduos é selecionado. O valor de $n = 2$ é usual. A seleção por torneio dispensa escalamento da aptidão e *ranking*.

Em uma outra versão, a seleção por torneio usa probabilidades. Se o torneio é entre dois indivíduos o primeiro ganha o torneio com probabilidade q (onde $1 > q > 0,5$) e o segundo com probabilidade $1 - q$. Para um torneio entre n indivíduos, o primeiro indivíduo ganha o torneio com probabilidade q , o segundo com probabilidade $q(1 - q)$, o terceiro com $q(1 - q)^2$ e assim por diante...

Aumentando o número n de indivíduos do torneio ou a probabilidade q do primeiro indivíduo vencer, aumenta-se a pressão de seleção, isto é, indivíduos com aptidão acima da média terão mais chances de serem selecionados. A seguir, esquemas de substituição de indivíduos serão apresentados.

3.3.5 AS SUBSTITUIÇÕES GERACIONAL E STEADY-STATE

Há dois tipos básicos de substituição de indivíduos:

- Substituição Geracional
- Substituição Steady-State

A) Substituição Geracional

Na Substituição Geracional, toda a população é substituída em cada geração. Ou seja, são criados N filhos para substituir N pais. Alternativamente, são criados N filhos e os N melhores indivíduos da união de pais e filhos substituem a população.

Na Substituição Geracional com Elitismo, os k melhores pais nunca são substituídos por filhos piores. Tipicamente é utilizado um valor de $k = 1$. Aumentando o valor de k , aumenta-se o risco da convergência prematura.

B) Substituição Steady-State

Na Substituição *Steady-State*, são criados apenas dois (ou um) filhos em cada “geração” que substituem os dois piores indivíduos da população. Os dois filhos podem substituir os pais ou os dois indivíduos mais velhos da população. Na forma geral da Substituição *Steady-State* são criados n filhos que substituem os n piores pais.

Apesar desta abordagem permitir usar os filhos para crossover e mutação tão logo eles sejam criados, ela gera um custo computacional extra. Por exemplo, quando cada filho é criado é necessário recalcular estatísticas, a aptidão média, reordenar a população por *ranking*, etc.. Contudo, em muitos problemas práticos este custo extra não é significativo, pois o custo computacional está quase todo concentrado no cálculo da aptidão. Davis (1991) usou esta técnica evitando inserir filhos duplicatas na população, obtendo uma melhora no desempenho. Na seção seguinte serão apresentados métodos que podem substituir a Roda da Roleta com mais eficiência.

3.3.6 AMOSTRAGEM ESTOCÁSTICA UNIVERSAL

A Roda da Roleta possui o problema de gerar grande variância no número esperado de filhos de cada indivíduo que é dado por $f_i/\bar{f}$, onde $\bar{f}$ é a aptidão média da população (3.3.6.2). Os seguintes métodos podem substituir a Roda da Roleta e amenizar este problema:

- Amostragem Estocástica do Resto sem Substituição (do inglês, *Remainder Stochastic Sampling without Replacement*).
- Amostragem Universal Estocástica ou SUS (do inglês, *Stochastic Universal Sampling*).

A) Amostragem Estocástica do Resto sem Substituição

Neste método, primeiramente é criada uma população intermediária (*mating pool*) para alocar as cópias dos indivíduos selecionados para crossover e mutação. Posteriormente, a aptidão normalizada de cada indivíduo é calculada pela Equação (3.3.6.1).

$$f'_i = \frac{f_i}{\bar{f}} \quad (3.3.6.1)$$

Onde,

$$\bar{f} = \frac{1}{N} \sum_{i=1}^N f_i \quad (3.3.6.2)$$

Seja $\text{int}(f'_i)$ a parte inteira da aptidão normalizada. Para cada indivíduo i da população são alocadas $\text{int}(f'_i)$ cópias na população intermediária. A parte fracionária da aptidão normalizada serve como probabilidade para completar a população intermediária. Por exemplo, suponha que um indivíduo tenha $f' = 2,4$. São alocadas então 2 cópias deste indivíduo na população intermediária. Uma terceira cópia é alocada com probabilidade 0,4.

B) Amostragem Universal Estocástica

Amostragem Universal Estocástica ou **SUS** (do inglês, *Stochastic Universal Sampling*) (Baker, 1987) é considerado o melhor método de amostragem pelos pesquisadores, por ser simples e teoricamente perfeito. Neste método, primeiro é criada uma população intermediária onde são alocadas as cópias dos indivíduos que vão para crossover e mutação. A seguir, a população de pais é embaralhada e é criado um gráfico tipo “torta” com fatias proporcionais à aptidão de cada indivíduo. Em volta da parte externa da “torta” põem-se N ponteiros igualmente espaçados. Por fim, o indivíduo apontado por cada ponteiro é copiado na população intermediária. Ver Figura 3.13.

A seguir será descrito como os Algoritmos Genéticos funcionam.

Figura 3.13. *Stochastic Universal Sampling*

3.4 COMO OS ALGORITMOS GENÉTICOS FUNCIONAM

Nesta seção serão apresentados alguns aspectos matemáticos dos Algoritmos Genéticos através do **Teorema dos Esquemas**.

3.4.1 O TEOREMA DOS ESQUEMAS

Este teorema foi a primeira explicação formal sobre o funcionamento dos Algoritmos Genéticos (Holland, 1975).

A) Esquemas

Um **esquema** é formado pelos símbolos 0, 1 e *. O cromossomo 01001 pode ter 32 esquemas entre os quais os esquemas *1*01, *1001 e *1*0*. Quando o símbolo * ocupa uma posição no esquema, esta posição pode ser ocupado por 1 ou 0. Um esquema H representa um conjunto de cadeias de bits. Por exemplo, o esquema $H = *1*01$ representa o conjunto todas as cadeias de 5 bits que possui 1 no segundo e quinto bits e 0 no quarto bit, isto é:

$$H = \{11101, 11001, 01101, 01001\}.$$

O Algoritmo Genético manipula esquemas quando é executado. Nesta seção, o comportamento de um esquema H ao longo de várias gerações será analisado. As seguintes definições serão úteis para esta análise:

- **Comprimento de um esquema H , ou $\delta(H)$** , é a diferença entre a última posição que é ocupada por 1 ou 0 e a primeira posição que é ocupada por 1 ou 0. Por exemplo, os esquemas $H_1 = *1*01$, $H_2 = **10*$ e $H_3 = 1****$ possuem os comprimentos $\delta(H_1) = 3$, $\delta(H_2) = 1$ e $\delta(H_3) = 0$, respectivamente.

- **Ordem de um esquema** H , ou $O(H)$, é o número de símbolos 1 e 0 que ele contém. Por exemplo, os esquemas $H_1 = *1*01$, $H_2 = **10*$, e $H_3 = 1****$ possuem ordens $O(H_1) = 3$, $O(H_2) = 2$ e $O(H_3) = 1$, respectivamente.

B) Relação do número de esquemas entre duas gerações consecutivas

Considere $S(t) = \{s_1, s_2, s_3, \dots, s_N\}$ a população de cromossomos na geração t . Seja $m(H, t)$ o número de cromossomos da população $S(t)$ que contém o esquema H na geração t . Considere $f(H, t)$ a média das aptidões dos cromossomos que contém o esquema H na geração t , ou seja:

$$f(H, t) = \frac{1}{m(H, t)} \sum_{s_i \in S(t) \cap H} f_i(t) \quad (3.4.1.1)$$

Seja $\bar{f}(t)$ a média das aptidões dos cromossomos da população na geração t , isto é:

$$\bar{f}(t) = \frac{1}{N} \sum_{i=1}^N f_i(t) \quad (3.4.1.2)$$

Onde N é o tamanho da população. Caso não sejam aplicados os operadores crossover e mutação, o número esperado de cromossomos com esquema H na geração $t+1$ é dado por:

$$m(H, t+1) = \frac{f(H, t)}{\bar{f}(t)} m(H, t) \quad (3.4.1.3)$$

Pela Equação (3.4.1.3), o número de esquemas H aumentará na geração seguinte se $f(H, t) > \bar{f}(t)$. Isto é, o número de esquemas pertencentes a cromossomos com aptidão acima da média aumentará na próxima geração. Por outro lado, o número de esquemas pertencentes a cromossomos de baixa aptidão diminuirá. Para se ter uma idéia da velocidade de crescimento dos bons esquemas no decorrer das gerações, considere que $f(H) / \bar{f} = 1 + a$, onde a é uma constante. Assim, a Equação (3.4.1.3) pode ser escrita como:

$$m(H, t+1) = (1 + a)m(H, t) \quad (3.4.1.4)$$

Supondo que a seja constante durante todas as gerações. Calculando o valor de $m(H, t)$ desde da geração $t=0$ obtém-se:

$$m(H, t) = m(H, 0)(1 + a)^t \quad (3.4.1.5)$$

A Equação (3.4.1.5) mostra que a quantidade de bons esquemas, isto é, aqueles contidos em cromossomos de aptidão acima da média, cresce exponencialmente ao longo das gerações. Contudo, o crossover e a mutação causam a destruição dos esquemas, portanto seus efeitos devem ser incluídos nesta análise.

C) Incluindo os efeitos do crossover e mutação

Considere um cromossomo de tamanho $l = 7$. Um esquema H deste cromossomo é mostrado na Figura 4.14. Dependendo do ponto de corte, o crossover pode destruir o esquema (Figura 4.14.b) ou não (Figura 4.14.c). Das $l-1 = 6$ posições possíveis de corte do crossover, apenas as $\delta(H) = 3$ posições entre o segundo bit e o quinto bit são capazes de destruir o esquema. Portanto, a probabilidade de destruição do esquema H é $3/6$. Considerando que o crossover ocorre com probabilidade p_c , então, a probabilidade de sobrevivência do esquema H ao crossover é dada por:

$$1 - p_c \frac{\delta(H)}{l-1} \quad (3.4.1.6)$$

Considere os esquemas $1****0*$ e $*10****$. Em ambos os esquemas existem 6 posições possíveis de corte pelo crossover. No primeiro esquema, as 5 primeiras posições de corte destroem o esquema, enquanto no segundo esquema apenas uma única posição (entre o 2º bit e o 3º bit) é capaz de destruir o esquema. Portanto, esquemas de grande comprimento $\delta(H)$ possuem poucas chances de sobrevivência ao crossover.

Há ainda outro fator que deve ser considerado. Se o mesmo esquema existe em ambos pais, este esquema sobreviverá mesmo que o crossover o tenha destruído. Por exemplo, considere o esquema H contido em Pai 1 e Pai 2 (ver Figura 3.15). Apesar do esquema H ter sido destruído pelo crossover ele ainda está presente em Filho 1 e Filho 2. Considere $P(H, t)$ a fração de cromossomos da população que contém o esquema H na geração t , que é dado por:

$$P(H, t) = \frac{m(H, t)}{N} \quad (3.4.1.7)$$

Considerando a possibilidade de dois pais terem o mesmo esquema H , a probabilidade de sobrevivência ao crossover passa a ser:

$$1 - p_c \frac{\delta(H)}{l-1} (1 - P(H, t)) \quad (3.4.1.8)$$

Desde que o número de bits que sofrem mutação em um esquema H é $O(H)$, a probabilidade do esquema sobreviver à mutação é dado por:

$$(1 - p_m)^{O(H)} \quad (3.4.1.9)$$

Onde p_m é a taxa de mutação. Incluindo o efeito do crossover e da mutação na Equação (3.4.1.3), obtém-se:

$$m(H, t+1) \geq \frac{f(H, t)}{\bar{f}(t)} m(H, t) \left[1 - p_c \frac{\delta(H)}{l-1} (1 - P(H, t)) \right] (1 - p_m)^{O(H)} \quad (3.4.1.10)$$

O sinal $\geq$ na Equação (3.4.1.10) é devido aos ganhos de esquemas H que são provenientes de pais que não possuem o esquema H mas cujos filhos o possuem. Por exemplo, na Figura 3.16 é mostrado que o esquema H não está contido em pai 1 e pai 2. Entretanto, o filho contém o esquema H . A seguir será apresentado o teorema dos esquemas.

Figura 3.14. O efeito do crossover sobre um esquema

Figura 3.15. Esquemas iguais em ambos os pais

Figura 3.16. Ganhos de esquemas devido ao crossover

C) Versão final

Se ambos os lados de (3.4.1.10) forem divididos pelo tamanho da população N obtém-se:

$$P(H, t+1) \geq \frac{f(H, t)}{\bar{f}(t)} P(H, t) \left[1 - p_c \frac{\delta(H)}{l-1} (1 - P(H, t)) \right] (1 - p_m)^{O(H)} \quad (3.4.1.11)$$

Esta expressão que relaciona a quantidade ou a porcentagem de esquemas entre duas gerações consecutivas é conhecida como o **Teorema dos Esquemas** e a partir dela é chega-se a seguinte conclusão:

Em um Algoritmo Genético tradicional usando seleção, crossover e mutação, o número de esquemas de pequeno tamanho, de baixa ordem e com aptidão acima da média da população crescerá exponencialmente nas gerações subsequentes (Goldberg, 1989).

Os esquemas de pequeno tamanho e de baixa ordem desempenham um papel importante nos Algoritmos Genéticos e são denominados de **blocos de construção**.

3.4.2 A HIPÓTESE DOS BLOCOS DE CONSTRUÇÃO E OUTROS TÓPICOS

A) A Hipótese dos blocos de construção

Os bons blocos de construção se combinarão com outros bons blocos de construção e, no decorrer das gerações, produzirão cromossomos de alta aptidão. Esta afirmação é conhecida como **hipótese dos blocos de construção**. Se esta hipótese está correta então, pelo Teorema dos Esquemas, um Algoritmo Genético encontrará a solução ótima. Porém surge uma pergunta: é absolutamente certo afirmar que a combinação de bons blocos de construção irá produzir bons cromossomos?

Na prática esta hipótese tem sido verdadeira em muitos problemas em áreas diversas. Os blocos de construção, quando incorporados em um indivíduo, melhoram sua aptidão. A codificação de cromossomos deve ser estruturada de modo a estimular a formação de blocos de construção.

B) Problemas deceptivos

Há problemas em que a hipótese descrita anteriormente falha. Nestes problemas, as cadeias de bits globalmente ótimas estão cercadas e próximas, em termos da distância de Hamming, a muitas cadeias de bits de aptidão muito baixa (é como se fosse uma “agulha em palheiro”). Tais cadeias “ruins” levam o Algoritmo Genéticos para longe das “boas” cadeias. Problemas com esta propriedade são denominados de **deceptivos**. Vale frisar este

tipo de problema é difícil também para qualquer procedimento de otimização (Goldberg,1989).

C) Paralelismo implícito

O Algoritmo Genético processa N cadeias de bits em cada geração. O número de esquemas processados está entre 2^l e $2^l N$ (se todas as cadeias de bits forem iguais o Algoritmo processa 2^l esquemas, se todas as cadeias foram diferentes o algoritmo processa $2^l N$ esquemas). Holland (1975) estimou que na ordem $O(N^3)$ esquemas são processados considerando que muitos tipos de esquemas não são processados devido a sua destruição pelo crossover e mutação (ver Golberg,1989). Apesar de somente N cadeias de bits serem processadas, um número muito maior de esquemas, $O(N^3)$, são processados em cada geração. Este resultado é denominado de **Paralelismo implícito**.

3.5 OPERADORES GENÉTICOS

Nesta seção serão apresentados vários tipos de operadores genéticos para representação binária e real.

3.5.1 REPRESENTAÇÃO BINÁRIA

É a representação tradicional. Os tipos de operadores crossover mais conhecidos para este tipo de representação são:

- Crossover de 1 ponto
- Crossover de 2 pontos
- Crossover multipontos
- Crossover uniforme

Uma descrição destes operadores será dada a seguir, com exceção do crossover de 1 ponto, já apresentado na seção 3.1.5.

A) Crossover de 2 pontos

O crossover de 2 pontos é apresentado na Figura 3.17. Os dois pontos de corte foram escolhidos aleatoriamente. Neste operador, o cromossomo pode ser interpretado como um anel formado pela união de suas extremidades, Figura 3.18. Observando o anel nota-se que

o crossover de 1-ponto é um caso particular do crossover de 2-pontos onde um dos pontos de corte é fixo sobre a junção do anel.

Portanto, o crossover 2-pontos desempenha também a função do crossover de 1 ponto, além de permitir a presença dos blocos de construção que existem sobre a junção do anel (o mesmo não ocorre no crossover de 1 ponto). Este é um dos motivos pela qual pesquisadores concordam que o crossover de 2 pontos é geralmente melhor do que o crossover de 1 ponto (Beasley, 1993b; Davis, 1991).

PAI 1	010	011000	101011
PAI 2	001	001110	001101
FILHO 1	010	001110	101011
FILHO 2	001	011000	001101

Figura 3.17. Crossover de 2 pontos

Figura 3.18. Crossover de 2 pontos visto como um anel

B) Crossover multipontos

O crossover multipontos é a extensão dos crossover's de 1 e 2 pontos para vários pontos. Um exemplo é o crossover de 4-pontos mostrado na Figura 3.19. Excessivos pontos de corte normalmente não dão bons resultados, uma vez que destroem com facilidade os blocos de construção.

PAI 1	101	010010	01010	01001
PAI 2	001	001110	00110	11100
FILHO 1	101	001110	01010	11001
FILHO 2	001	010010	00110	01100

Figura 3.19. Crossover de 4 pontos

C) Crossover uniforme

O crossover uniforme é apresentado da Figura 3.20. Neste caso, para cada par de pais é gerada uma máscara de bits aleatórios. Se o primeiro bit da máscara é 1, então o primeiro bit do pai 1 é copiado para o primeiro bit do filho 1. Caso contrário, o primeiro bit do pai 2 é copiado para filho 1. O processo se repete para os bits restantes do filho 1. Na criação do segundo filho o processo é invertido. Se o bit da máscara é 1, então será copiado o bit do pai 2. Se o bit for 0, então será copiado o bit do pai 1.

Em Eshelman et al (1989) é investigada a diferença de desempenho entre o crossover 1-ponto, 2-pontos, de múltiplos pontos e uniforme. A conclusão, conforme relatado em (Beasley, 1993), é que não há grandes diferenças de desempenho entre eles.

Figura 3.20. Crossover Uniforme

3.5.2 REPRESENTAÇÃO COM NÚMEROS REAIS

Neste texto, os cromossomos foram representados por cadeias de bits. É comum também representar cromossomos com números reais. Definir qual é a melhor das duas representações é ainda um tópico de pesquisa.

A representação binária é historicamente importante, foi utilizado nos trabalhos pioneiros de Holland (1975). É a representação tradicional. É fácil de usar e manipular além de simples de analisar teoricamente. Contudo, não há uniformidade nos operadores (mutação nos primeiros bits do gene afeta mais a aptidão que mutação nos últimos bits do gene).

A representação real é mais natural para o ser humano do que a cadeia de bits. Por exemplo, um cromossomo para o Exemplo 3.1 poderia ter simplesmente dois números reais (-50, 35). Além disso, alguns pesquisadores têm obtido bons resultados com números reais (Michalewicz, 1994). Outra vantagem é que além de poder utilizar os operadores da representação binária (crossover de 1 e 2 pontos, uniforme etc..) existe ainda a possibilidade de criar uma grande variedade de operadores (Beasley et al, 1993b). Ver Tabela 3.3.

Tabela 3.3. Operadores Genéticos para representação real.

Operador	Descrição
Crossover média	Média aritmética entre dois genes
Crossover média geométrica	Raiz do produto entre dois genes
Extensão	Calcula a diferença entre dois genes e adiciona a diferença ao maior gene ou subtrai do menor gene.
Mutação <i>creep</i>	Adiciona a ou subtrai de um gene uma pequena quantidade gerada com distribuição uniforme, normal ou por alguma outra distribuição. A idéia deste operador é a seguinte: se o gene está próximo do ponto máximo, esta pequena perturbação pode levá-lo ao topo mais rápido.
Mutação <i>creep</i> geométrico	Multiplica o gene por um valor próximo de um

3.6 EPILOGO

Este capítulo descreveu Algoritmos Genéticos, um tópico de pesquisa que tem despertado crescente interesse e que tem um papel importante nesta Dissertação. O capítulo iniciou-se com um aplicação em otimização de função na qual foram apresentados os conceitos básicos. Em seguida foram realizadas comparações com outras abordagens de busca e otimização tais como ao algoritmo de Recozimento Simulado.

Também foi descrito problemas como convergência prematura entre outros e várias técnicas como *ranking*, escalamento da aptidão e seleção por torneio que visam melhorar desempenho do Algoritmo Genético. Foi apresentado o Teorema dos Esquemas na qual tenta explicar matematicamente o funcionamento dos Algoritmos Genéticos.

No capítulo seguinte, serão apresentados mais informações sobre os Algoritmos Genéticos na sua aplicação para otimização de Redes Neurais.

4

ALGORITMOS GENÉTICOS PARA REDES NEURAIS

"O importante é colocar o amor em tudo que estamos realizando"

*"Amanhã será, sem dúvida, um belo dia, mas para trabalhar e servir,
renovar e aprender, hoje é melhor"*
(Espírito André Luiz)

*"Ensinar não é ferir. É orientar o próximo, amorosamente,
para o reino da compreensão e da paz"*
(Espírito André Luiz)

*"Minúscula é a formiga, mas edifica, a força de perseverança,
complicadas cidades subterrâneas"*
(Espírito Emmanuel)

4.1 INTRODUÇÃO

Este capítulo descreve várias maneiras de combinar Algoritmos Genéticos com Redes Neurais, a saber:

- Otimização da topologia de uma Rede Neural. Por exemplo, otimizar o número de unidades, o número de camadas e as conexões entre unidades.
- Treinamento (otimização dos pesos sobre as conexões) de uma Rede Neural, substituindo a regra de aprendizado (e.g. o *backpropagation*).
- Otimização da regra de aprendizado. Ou seja, dada uma topologia, encontrar uma regra de aprendizado eficiente.

Geralmente, as classificações encontradas na literatura dividem a combinação de Algoritmo Genéticos com Redes Neurais em dois grupos de acordo a representação do cromossomo:

- Codificação direta, forte ou de baixo nível.
- Codificação indireta, fraca ou de alto nível.

A codificação direta refere-se a métodos que codificam diretamente os parâmetros da Rede Neural (i.e conexões, pesos, número de unidades etc) no cromossomo. A codificação indireta, ao contrário, codifica regras ou estruturas que contém informações ou “receitas” de como construir a Rede Neural.

Um exemplo de codificação direta é mostrado a seguir.

4.1.1 UM EXEMPLO

Na Figura 4.1, é apresentado um Algoritmo Genético para Redes Neurais. Neste exemplo, é utilizado a representação de cromossomo encontrado em (Miller et al, 1989). Figura 4.2. Cada cromossomo codifica uma Rede Neural com n unidades na qual é representada por uma matriz $n \times (n+1)$. Cada elemento a_{ij} da matriz assume o valor 1 ou 0 dependendo se há ou não conexão da unidade j para a unidade i . A coluna $n+1$ da matriz indica a existência de *bias*. O cromossomo é formado pela concatenação das linhas da matriz.

Para avaliar o cromossomo, o mesmo é decodificado gerando uma Rede Neural (fenótipo). A rede é treinada por um número fixo de ciclos e testada sobre um conjunto de treinamento (e/ou validação). O erro quadrado médio ou MSE (Mean-Square-Error) sobre o conjunto de treinamento é utilizado para obter a aptidão.

Vale frisar que não há uma correspondência direta entre erro da rede e aptidão, pois baixos erros são associados a altos valores de aptidão. Pode-se transformar erro em aptidão usando fórmulas como: $1/\text{MSE}$, $1/(1+\text{MSE})$, $\text{MSE_Máximo} - \text{MSE}$ etc... ou utilizando *Ranking*. Além do erro da rede, podem ser utilizados vários critérios ou combinações deles para determinar a aptidão. Por exemplo:

- Erro de desempenho (sobre o conjunto de treinamento);
- Erro de generalização (sobre o conjunto de validação);
- Velocidade de treinamento;
- Complexidade da Rede Neural (número de unidades ou conexões);

- Número de camadas da Rede Neural;
- Percentagem de padrões corretamente classificados.

Tais critérios podem ser combinados da seguinte forma (Harp et al, 1991): Considere p_j um destes critérios, como por exemplo: o MSE de validação ou o número de ciclos de treinamento. A aptidão $f(i)$ do cromossomo i é calculada pela soma ponderada de tais critérios:

$$f(i) = \sum_j a_j \phi_j(p_j(i)) \quad (4.1.1.1)$$

Onde, a função ϕ_j transforma a medida p_j em uma escala adequada e o peso a_j serve para o usuário realçar qual critério da rede é mais importante para sua aplicação. A seguir, justificativas do uso de Algoritmos Genéticos para otimizar Redes Neurais serão discutidas.

Figura 4.1. Um Algoritmo Genético aplicado a Rede Neural

Figura 4.2. Representação do cromossomo por Miller et al (1989)

4.1.2 POR QUE USAR OS ALGORITMOS GENÉTICOS

Os Algoritmos Genéticos é uma técnica robusta que tem sido aplicado com sucesso nas mais variadas áreas:

- Otimização de funções numéricas não-diferenciável, ruidosas e multimodais.
- Processamento de imagem.
- Otimização combinatorial (e.g. o problema do caixeiro viajante, agendamento etc..).
- Design (e.g. determinar a estrutura de pontes)
- Aprendizado de máquina.

Em geral, os Algoritmo Genéticos, não é a melhor técnica de otimização para um problema quando este já possui uma técnica específica revolve-lo. O principal campo de aplicação dos Algoritmos Genéticos são problemas de otimização na qual não existe técnicas para soluçona-lo (Beasley, 1993a). Talvez os principais motivos de usar os Algoritmos Genéticos para otimizar a topologia da Rede Neural sejam:

- A falta de um método eficiente para determinar a topologia.
- O complicado espaço de busca das topologias que é o torna intratável para técnicas convencionais de otimização.

Alguns problemas encontrados no espaço de busca de topologias de Redes Neurais são:

- É muito grande por que não há limite para o número de unidades e conexões. Portanto não é possível uma busca exaustiva por todas as possíveis topologias em um tempo computacional aceitável.
- Não é diferenciável, por que parâmetros discretos (e.g. números de unidades) estão envolvidos. Logo, métodos baseados no gradiente não podem ser utilizados.
- É multimodal, isto é, existe vários pontos de máximo. Isto significa que duas topologias completamente distintas podem ter bom desempenho.
- É deceptivo, isto é, pequenas mudanças na topologia pode levar a uma grande mudança de desempenho.
- A função objetivo é ruidosa. Isto ocorre com Redes treinadas com o *backpropagation* por que inicialização dos pesos é aleatória.

Existem uma vasta bibliografia de combinação de Redes Neurais com Algoritmos Genéticos (Balakrishnan e Honavar, 1995). Um livro texto sobre o assunto é encontrado em (Rooij et al, 1997).

A seguir, na seção 4.2, serão apresentadas várias abordagens de otimização de topologias de Redes Neurais. Será mostrado que as redes evoluídas não tem a mesma topologia de redes tradicionais, porém, muitas vezes, apresentam desempenho superior. Na seção 4.3, será focado a substituição do *backpropagation* pelo algoritmo genético, bem como suas vantagens e desvantagens. Na seção 4.4, é apresentado uma interessante abordagem na qual os Algoritmos Genéticos otimiza o próprio algoritmo de treinamento. Na seção 4.5, serão estudados modelos específicos de Algoritmos Genéticos para redes RBF. Por fim, na seção 4.6, serão discutidos os temas apresentados neste capítulo.

4.2 OTIMIZAÇÃO DA TOPOLOGIA

Vários modelos com codificação direta e indireta serão apresentados.

4.2.1 CODIFICAÇÃO DIRETA

Os modelos que utilizam codificação direta codificam diretamente a topologia da rede (pesos, conexões, etc..) no cromossomo. Geralmente, há pouco esforço computacional para decodificar o cromossomo e gerar a Rede Neural. Exemplos de tais modelos estão em

(Miller et al, 1989; Whitley et al, 1990; Schiffmann et al, 1991, 1992, 1993) na qual serão apresentados a seguir.

O MODELO DE MILLER, TODD E HEGDE (1989)

Esta abordagem foi uma das primeiras tentativas de combinar Algoritmos Genéticos com Redes Neurais. É também uma das mais simples.

A) Cromossomo

Representação do cromossomo foi mostrado na seção 4.2. Porém, vale notar que esta representação pode gerar conexões *feedback* (i.e. loops de conexões). Os autores desprezaram tais conexões para trabalhar somente com Redes *feedforward*.

B) Operadores genéticos

Este trabalho utilizou o operador de mutação tradicional (alternando os bits). O crossover empregado consistia na troca de uma linha inteira da matriz entre dois cromossomos pais. A idéia por trás deste operador é que cada linha contém o conjunto de conexões que chegam a uma única unidade. Considera-se este conjunto um importante *bloco de construção* (seção 3.4) da rede. É a mesma idéia do crossover de Montana e Davis (1989) visto mais tarde.

C) Experimentos

O cálculo da aptidão do cromossomo foi apresentado na seção 4.4.1. Os problemas tratados neste trabalho foram: XOR, 4-Quadrantes e Cópia de Padrões. Foi utilizada uma rede MLP em que o número de unidades intermediárias variou de 5 unidades para o XOR até 20 para a Cópia de Padrões. O tamanho da população era de 50 cromossomos, com taxa de crossover de 0.60 e taxa de mutação de 0.005. Este trabalho usou apenas problemas simples e ainda precisa ser testado em problemas mais complexos. A seguir será mostrada um outro modelo, porém semelhante a este.

O MODELO DE WHITLEY ET AL (1990)

Este modelo determina os pesos da Rede Neural ao mesmo tempo que determina sua conectividade.

A) Cromossomo

A representação é apresentada na Figura 4.3. Cada conexão é codificada em uma cadeia de 9 bits composta por dois campos. No primeiro, um bit indica se a conexão existe ou não. No segundo campo, uma cadeia de 8 bits codifica o peso da conexão. O primeiro bit é o sinal do peso (1 para +, e zero para -) e os 7 bits restantes é a magnitude do peso na base 2.

B) Aptidão, operadores genéticos e experimentos

Os operadores genéticos foram os convencionais. A aptidão foi baseada no erro da rede. Contudo, os autores alocaram mais ciclos de treinamento para redes pequenas, de modo que a aptidão favorecesse tais redes. Este sistema foi aplicado em funções booleanas simples. A seguir será discutida um modelo simples, porém diferente dos anteriores.

Figura 4.3. Representação do cromossomo por Whitley et al (1990).

O MODELO DE SCHIFFMANN ET AL (1991, 1992, 1993)

A) Cromossomo

É representado por uma lista de unidades. Cada unidade contém, por sua vez, uma sublista com as conexões que chegam a ela. Figura 4.4.

B) Operadores genéticos

O operador de mutação realiza as seguintes operações:

- Remoção e adição de conexões *feedforward*.
- Adição de unidades intermediárias com a sublista, de conexões *feedforward* que chegam a ela, gerada aleatoriamente.

No crossover, o ponto de corte ocorre somente entre unidades, isto é, a sublista de conexões não é dividida. O crossover, Figura 4.5, pode gerar redes com unidades sem conexão de saída (por exemplo, a unidade 3 mostrada na Figura 4.5) ou unidades sem conexão com unidades precedentes. Tais unidades podem ser eliminadas ou convertidas em *bias*. Cuidados devem ser tomados para garantir que as redes tenham a mesma interface entrada-saída. O corte do crossover deve posicionar-se após a última unidade de entrada (evita filhos iguais aos pai).

C) Experimentos

Este sistema foi aplicado no problema XOR e em um problema médico (classificação da glândula tireóide) obtendo bons resultados.

A seguir serão apresentadas abordagens baseadas na codificação indireta.

Figura 4.4. Representação do cromossomo por Schiffman et al (1992).

Figura 4.5. Crossover de Schiffmann et al (1992).

4.2.2 CODIFICAÇÃO INDIRETA

Existem modelos que não codificam diretamente a topologia da Rede Neural no cromossomo. Ao invés disso, regras ou algum tipo de informação são codificadas para mais tarde para gerar a Rede Neural. Geralmente, os cromossomos são complexos e requerem considerável esforço computacional para decodificá-los. Um exemplo encontra-se em (Harp et al, 1989) onde o cromossomo é dividido em áreas, onde cada área define um grupo de unidades da Rede Neural e suas conexões para as demais unidades da Rede Neural. Outro exemplo encontra-se em (Kitano, 1990a, 1990b) que codificou, no cromossomo, regras gramaticais. No processo de decodificação estas regras são utilizadas para produzir a Rede Neural. A seguir serão vistas estas duas abordagens.

O MODELO DE HARP ET AL (1989, 1991)

É uma dos primeiros modelos, senão o primeiro, a codificar a topologia da Rede Neural em um cromossomo.

A) Cromossomo

Primeiro, será descrito um exemplo simplificado do modelo de (Harp et al, 1991), mas que captura as principais idéias. O cromossomo é dividido em um número variável de áreas, conforme ilustra Figura 4.6. Cada área define um grupo de unidades da rede. A própria área, por sua vez, é subdividida em:

- Área de parâmetros
- Áreas de projeção.

O campo ID da área de parâmetros identifica um grupo de unidades. Em cada grupo, as unidades, que são totalmente conectadas, são espacialmente dispostas em forma de uma grade, cujas dimensões são dadas pelos campos X e Y da área de parâmetros. Cada área de parâmetros possui um número variável de áreas de projeção. A área de projeção define as conexões de um grupo de unidades para outro grupo de unidades. Os campos da área de projeção são:

- ID_DEST: identifica um grupo de unidades destino
- X_DEST e Y_DEST: definem a porção ou bloco da grade de unidades destino que receberá as conexões
- DESLOC: determina o deslocamento do bloco definido por X_DEST e Y_DEST em relação ao lado esquerdo da grade de unidades destino

Somente as unidades mais externas de um grupo de unidades possuem conexões para o grupo de unidades destino e é totalmente conectada com o bloco destino. Neste exemplo, o ID de cada grupo varia de 0 a 7 (não necessariamente único). Os números 0 e 7 são reservados para os grupos que contêm as unidades de entrada e as unidades de saída, respectivamente.

Na decodificação do cromossomo podem ser geradas Redes Neurais inválidas. Por exemplo, redes sem caminho entre a entrada e a saída, áreas que não contêm áreas de projeção etc.. No entanto, quando pequenas anormalidades ocorrem, elas podem ser corrigidas pelo programa.

É apresentado na Figura 4.7 um exemplo de decodificação. Para simplificar, são mostrados números reais no cromossomo, ao invés de bits. O grupo de unidades 0 (ID = 0) tem projeção sobre um bloco 1x1 do grupo de unidades ID = 5 com deslocamento 2. Mas, como deslocamento 2 não é possível, considera-se apenas deslocamento 1. Pequenos erros que surgiram foram corrigidos na decodificação.

O trabalho original (Harp et al, 1991) usou a representação da Figura 4.8. Na área de parâmetros, um campo define o número de unidades. Os parâmetros X e Y impõem uma organização espacial a um grupo de unidades da rede. O programa que irá gerar a rede deve achar uma grade de unidades que corresponda ao número de unidades e ao mesmo tempo permaneça dentro das dimensões X e Y.

As áreas de projeção contêm um parâmetro que determina a densidade de conexões entre grupos de unidades, isto é, nem todas as conexões são permitidas. Possuem também parâmetros que controlam o aprendizado através do algoritmo *backpropagation*. O parâmetro MODO_DE_ENDEREÇAMENTO pode ser absoluto ou relativo. Quando for absoluto o parâmetro DESTINO especifica a identificação ID de um grupo de unidades. Quando for relativo, DESTINO especifica a área em relação à área corrente. A área imediatamente após a área corrente será zero, a seguinte será 1, e assim por diante.

Figura 4.6. Cromossomo codificando as camadas da Rede Neural

Figura 4.7. Decodificação de um cromossomo e correção dos defeitos.

B) Operadores genéticos

As marcas de separação de áreas são necessárias para que o ponto de corte do crossover não ocorra incorretamente, por exemplo, entre uma área de parâmetros de um cromossomo e uma área de projeção de outro cromossomo, uma vez que cromossomos possuem comprimentos variáveis. O operador de mutação é a mutação tradicional (invertendo os bits). A aptidão foi calculada por (4.1.1.1).

C) Experimentos

Este modelo foi implementado em uma rede de computadores. Um programa principal, rodando em uma das máquinas, gerenciava o Algoritmo Genético enquanto outras máquinas treinavam as redes. O sistema foi utilizado no problema XOR, no reconhecimento de dígitos e na aproximação da função seno. Os autores ficaram surpresos quando o sistema encontrou uma rede sem camadas intermediárias para o problema de reconhecimento de dígitos. Eles não estavam informados que o problema era linearmente separável.

Em (Mandischer, 1993) é encontrada uma representação seguindo a mesma linha do trabalho apresentado. A seguir será apresentada uma abordagem, conhecida como codificação gramatical, muito diferente daquelas apresentadas até o momento.

Figura 4.8. Representação do cromossomo por Harp et al (1991)

O MODELO DE KITANO (1990A, 1990B)

É um dos primeiros modelos de codificação gramatical. A Rede Neural é produzida por regras de gramática⁸. O Algoritmo Genético busca um conjunto de regras que produza uma boa topologia.

A) O cromossomo

Um exemplo de um conjunto de regras é mostrado na Figura 4.9. Cada regra possui no lado direito uma matriz 2x2. Há dois subconjuntos de regras. No primeiro subconjunto, o lado esquerdo da regra é um símbolo não-terminal no intervalo A-Z e o lado direito é uma matriz 2x2 de símbolos nos intervalos A-Z e a-p. O segundo conjunto de regras é fixo e, portanto, não faz parte do cromossomo. É formado de símbolos terminais. Possui 16 regras, constituída de letras minúsculas a-p, representando todas as 16 combinações de matrizes 2x2 de uns e zeros.

$$\begin{aligned} S &= \begin{bmatrix} A & B \\ C & D \end{bmatrix} & A &= \begin{bmatrix} i & e \\ a & o \end{bmatrix} & B &= \begin{bmatrix} a & e \\ a & b \end{bmatrix} & C &= \begin{bmatrix} a & a \\ a & a \end{bmatrix} & D &= \begin{bmatrix} a & a \\ a & b \end{bmatrix} \\ a &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & b &= \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} & e &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} & i &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} & o &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \end{aligned}$$

Figura 4.9. Regras para produção de Redes Neurais por Kitano (1990).

A regra com o símbolo não-terminal S inicia a produção da rede. Os símbolos desta matriz são substituídos, seguindo as regras, produzindo uma matriz 4x4. Conforme mostrado na Figura 4.10. O processo se repete e até formar uma matriz 8x8 de somente uns e zeros. O número 1 na linha i e coluna j representa a conexão entre as unidades i e j . Quando $i = j$, o número 1 significa a existência da unidade i na Rede Neural. Este modelo só evoluiu Redes *feedforward*, ou seja, conexões *feedback* foram desprezadas.

⁸ Gramática é conjunto de regras para produzir estruturas. Uma gramática muito simples com duas regras é mostrada abaixo:

1) $S \rightarrow aSb$

2) $S \rightarrow c$

S é o símbolo inicial da gramática. As letras maiúsculas (neste caso existe apenas a letra S) são denominadas símbolos não-terminais. As letras minúsculas a, b e c são denominadas símbolos terminais.

Para produzir uma estrutura (i.e. uma cadeia de letras) com a gramática acima, basta iniciar com a regra 1, $S \rightarrow aSb$, para produzir aSb. Então, usando a regra 2, substituir S por c para produzir a cadeia acb, ou seja, $S \rightarrow aSb \rightarrow acb$. Para produzir a cadeia aacbb, a regra 1 é utilizada duas vezes. Por fim, usa-se a regra 2: $S \rightarrow aSb \rightarrow aaSbb \rightarrow aacbb$. É fácil ver que as cadeias produzidas têm a forma geral $a^n cb^n$. Há várias aplicações de gramáticas, entre elas, a definição formal de linguagens de programação como C, FORTRAN, Pascal etc... Um livro texto sobre gramáticas é (Hopcroft e Ullman, 1979).

Alguns autores (Roberts et al, 1995), limitam o tamanho da matriz e começam a produção da rede a partir do canto superior esquerdo da matriz, Figura 4.11. Os símbolos que ultrapassam os limites da matriz são descartados. Esta matriz é semelhante a matriz de Miller et al (1989), Figura 4.2.

O cromossomo, Figura 4.12, codifica diretamente as regras. Cada regra é subdividida em 5 partes. A primeira representa o lado esquerdo da regra e as restantes representam os 4 elementos da matriz 2x2. Caso ocorram duas ou mais regras com o mesmo símbolo no lado esquerdo, somente a primeira regra será considerada (Mitchell, 1996).

Figura 4.10. Produção de uma Rede Neural pelo modelo de Kitano (1990)

B) Operadores genéticos

Foi utilizado o crossover multipontos (corte do cromossomo em um ou mais pontos). A mutação simplesmente substitui um símbolo do cromossomo por outro símbolo do alfabeto A-Z e a-p. A probabilidade de mutação depende da distância de *Hamming* (número de letras diferentes) entre os dois cromossomos pai. Quanto menor a distância, maior a taxa de mutação. Isto tende a manter a diversidade da população, pois a taxa de mutação aumenta quando os cromossomos pai tornam-se “mais semelhantes”.

C) Experimentos

Este trabalho utilizou o problema da codificação/decodificação. Neste problema, a codificação gramatical apresentou desempenho superior àquele obtido pela codificação direta. Este método encontrou redes mais rapidamente e com menores taxas de erro. Além disso, a codificação gramatical mostrou melhor desempenho quando o número de unidades intermediárias aumentava. A codificação indireta tem a vantagem de gerar cromossomos menores e isto pode ser útil para evoluir grandes topologias de rede. Porém, não se pode afirmar que a codificação gramatical seja superior a codificação direta com base apenas nos

problemas simples utilizados na comparação. Outros modelos de codificação gramatical são encontrados em (Gruau, 1992, 1994). A seguir serão feitos alguns comentários sobre a otimização da topologia de Redes Neurais usando Algoritmos Genéticos.

Figura 4.11. Produção de uma Rede Neural usando regras de gramática

Figura 4.12. Representação de uma gramática no cromossomo

4.2.3 COMENTÁRIOS

Apesar de simples, a codificação direta gera um aumento explosivo do tamanho do cromossomo quando a rede cresce. Isto limita sua aplicação para pequenas topologias de rede. Já a codificação indireta permite grandes topologias serem codificadas em cromossomo mais compactos e tende a gerar redes mais regulares. Contudo, tais representações compactas podem em alguns casos restringir o espaço de busca.

Tem-se combinado Algoritmos Genéticos com Redes Neurais em funções simples como o problema do ou-exclusivo (XOR) em grande parte das pesquisas. São raras as pesquisas com problemas complexos. Este é um tema ainda em estudo. Outro tema ainda em estudo é o problema do mapeamento estrutural-funcional descrito a seguir.

4.2.4 O PROBLEMA DO MAPEAMENTO ESTRUTURAL-FUNCIONAL

Em geral, redes estruturalmente diferentes são representados por cromossomos diferentes mesmo que as topologias sejam funcionalmente iguais. Por exemplo, duas redes e seus respectivos cromossomos são mostradas na Figura 4.13. No cromossomo, cada conexão é representada por (b,w) onde b indica a existência ($b=1$) ou não ($b=0$) na conexão e w é o peso da conexão. A única diferença entre as redes está na posição relativa das unidades X e Y . Portanto, as redes são funcionalmente equivalentes e deveriam gerar cromossomos iguais. Porém, tal diferença estrutural gerou cromossomos diferentes.

Permutação de unidades não altera a função computada pela rede. Contudo, gera cromossomos diferentes. Este problema aumenta significativamente o espaço de busca e é conhecido como problema do mapeamento estrutural-funcional (também chamado de *competing conventions problem*).

Contudo, ainda não está claro como este problema afeta a busca genética. Teoricamente, o problema cresce exponencialmente com o número de unidades intermediárias. Alguns trabalhos sugerem em que este problema é menor do que se supõe, mas deve ser evitado quando possível (Blanke, 1995; Hancock, 1992). A seguir será visto como utilizar Algoritmo Genético para substituir a regra de aprendizado (ex. o algoritmo *backpropagation*) da Rede Neural.

Figura 4.13. O problema do mapeamento estrutural-funcional

4.3 TREINAMENTO

Nesta seção, o Algoritmo Genético será usado para substituir o *backpropagation*.

4.3.1 INTRODUÇÃO

Há vantagens e desvantagens no treinamento genético. O *backpropagation* é um método do gradiente que são métodos de busca local e por isso propensos a cair em mínimos locais. Já os Algoritmos Genéticos, como explicado no capítulo 3, realizam uma busca global e, em geral, superam os mínimos locais.

Os Algoritmos Genéticos não necessitam de nenhuma informação adicional para treinar a rede (e.g. a derivada da função) e não impõem nenhuma restrição à topologia uma vez que não requerem a retropropagação do erro (necessário no *backpropagation*). Porém, em grandes redes, o treinamento via Algoritmos Genéticos consome muito tempo de processamento, o que pode torná-lo impraticável. A seguir será apresentado um exemplo desta abordagem.

O MODELO DE MONTANA E DAVIS (1989)

A) Cromossomo

Na Figura 4.14 é apresentado a codificação do cromossomo que é uma lista de pesos em uma dada ordem. Esta representação usou números reais. Os cromossomos da população inicial possuem pesos aleatórios entre -1.0 e +1.0.

B) Operadores genéticos

O operador de mutação seleciona n unidades e acrescenta um número aleatório entre -1.0 e +1.0 aos pesos das conexões que chegam a cada unidade selecionada. No operador crossover, para cada unidade do cromossomo filho, escolhe-se a unidade correspondente de um dos dois cromossomos pai. Depois, copia-se para o filho, os pesos das conexões que chegam a tal unidade escolhida. Idéia semelhante foi usada em (Miller et al, 1989), seção 4.2.1. Um exemplo é mostrado na Figura 4.2. Neste trabalho, o crossover gerava apenas um filho.

C) Experimentos

Os autores aplicaram este modelo para classificar sons subaquáticos em duas classes. A topologia da rede era 4-7-10-1 com duas camadas internas e 108 conexões. Adicionalmente, haviam 18 conexões de saída de uma unidade *threshold*, totalizando 126 conexões. A população foi de 50 indivíduos usando 200 gerações. A aptidão foi baseada no erro da rede. O treinamento do Algoritmo Genético foi comparado com o *backpropagation* do seguinte modo: duas avaliações de aptidão (i.e. dois testes) equivalem a uma interação do *backpropagation* (propagação para frente + retropropagação). Com base nesta comparação, o Algoritmo Genético superou o *backpropagation*. Mais comentários serão vistos na próxima seção.

Figura 4.14. Representação do cromossomo por Montana e Davis (1989)

Figura 4.15. Crossover de Montana e Davis (1989)

4.3.2 COMENTÁRIOS

No trabalho de (Montana e Davis, 1989) o Algoritmo Genético superou o *backpropagation*. Porém, não significa que o mesmo ocorra em todos os casos. Além disso, conforme exposto em (Schaffer et al., 1992), os Algoritmos Genéticos não têm demonstrado superar melhoramentos no *backpropagation* como, por exemplo, o algoritmo *quickprop* (Fahlman, 1988).

Alguns autores sugerem um treinamento que combine o Algoritmo Genético com o *backpropagation*, apesar desta abordagem não está ainda bem clara (Branke, 1995). Mas como seria este treinamento híbrido? Os Algoritmos Genéticos convencionais são bons em encontrar rapidamente uma solução aproximada (boas regiões do espaço de busca). Porém, a partir de determinadas gerações encontra dificuldades de melhorar a precisão da solução, porque tal “ajuste fino” passa a depender quase que exclusivamente do operador de mutação. Neste ponto, parece ser razoável inicializar uma Rede Neural com a solução aproximada dos Algoritmos Genéticos e utilizar o *backpropagation* para fazer o “ajuste fino” da solução, isto é, a subida do monte (*hill-climbing*).

Os Algoritmos Genéticos podem ser úteis para encontrar pesos em problemas de controle, que utilizam apenas aprendizado por reforço. Porque, nestes casos, o *backpropagation* não pode ser utilizado (Whitley, 1993).

A seguir será apresentado um terceiro modo, além do treinamento e da otimização da topologia, de combinar Algoritmos Genéticos com Redes Neurais. A regra de aprendizado será evoluída.

4.4 EVOLUÇÃO DA REGRA DE APRENDIZADO

A evolução da regra de aprendizado consiste na busca de algoritmos de treinamento semelhantes ao *backpropagation* ou a regra delta, por meio dos Algoritmos Genéticos. É uma idéia mais interessante que as anteriores porque começou a explorar como o processo evolucionário pode produzir sistemas que aprendem (Jones, 1993).

O MODELO DE CHALMERS (1991)

Neste trabalho, o Algoritmo Genético evoluiu regras de aprendizado para Redes Neurais sem camadas intermediárias, tais como o Perceptron e o Adaline (Widrow e Lehr, 1990).

A) Cromossomo

O autor baseou-se na regra Delta (seção 2.4.2) para dar a forma da regra de treinamento a ser evoluída. Na regra Delta, o ajuste de pesos para cada padrão é dado por

$$\Delta w_{ij} = \eta(x_i t_j - x_i y_j) \quad (4.4.1)$$

Onde,

Δw_{ij} é o ajuste do peso da conexão entre a unidade de entrada i e a unidade de saída j .

x_i é a ativação da unidade de entrada i .

y_j é a ativação da unidade de saída j .

t_j é a saída desejada fornecida pelo padrão de treinamento.

η é a taxa de aprendizado

A regra a ser evoluída tem a seguinte forma geral (na qual a regra Delta é um caso particular):

$$\Delta w_{ij} = k_0(k_1 w_{ij} + k_2 x_i + k_3 y_j + k_4 t_j + k_5 w_{ij} x_i + k_6 w_{ij} y_j + k_7 w_{ij} t_j + k_8 x_i y_j + k_9 x_i t_j + k_{10} y_j t_j) \quad (4.4.2)$$

O Algoritmo Genético evolui os 10 coeficientes k 's da regra. No cromossomo, foi utilizado cinco bits para o coeficiente k_0 (taxa de aprendizado) e três bits para os demais

k 's. Detalhes são mostrados na Figura 4.16. Seguindo esta codificação, a regra Delta seria representada por

11011 000 000 000 000 000 000 000 010 110 000

que ao ser decodificada resultaria em

4 0 0 0 0 0 0 0 -2 2 0

Substituindo os coeficientes em (4.4.2) obtém-se a regra Delta:

$$\Delta w_{ij} = 8x_i(t_j - y_j)$$

k_0 possui 5 bits onde,

O primeiro bit é o sinal (1 para + e zero para -)

O restante é um inteiro n , tal que se:
$$\begin{cases} n = 0, k_0 = 0 \\ n \neq 0, |k_0| = 2^{n-9} \end{cases}$$

Então, k_0 só pode assumir os valores: 0, $\pm 1 / 256$, $\pm 1 / 128, \dots$, ± 32 , ± 64

2) Os outros k 's possuem 3 bits onde,

O primeiro bit é o sinal (1 para + e zero para -)

Os demais representam um inteiro n , tal que se
$$\begin{cases} n = 0, k_0 = 0 \\ n \neq 0, |k_0| = 2^{n-1} \end{cases}$$

Figura 4.16. Esquema de codificação do cromossomo de Chalmers (1991)

B) aptidão

Foram utilizados 20 problemas com padrões linearmente separáveis para obter a aptidão na qual foi calculada como segue: para cada um dos 20 problemas foi construída uma Rede Neural que era treinada com a regra de aprendizado codificado no cromossomo. Em seguida, para cada problema, foi calculada uma aptidão pela fórmula:

$$f_n = 100(1 - \text{MSE})$$

Finalmente, a aptidão do cromossomo foi calculado pela média das aptidões de cada problema:

$$F = \frac{1}{20} \sum_{n=1}^{20} f_n$$

C) Experimentos

Utilizou-se uma população de 40 indivíduos, crossover de dois pontos (taxa de 0.8) e mutação tradicional (com taxa de 0.01) e 1000 gerações. As melhores regras evoluídas tiveram aptidões entre 80% e 98% com média de 92%. Segundo este esquema, a regra Delta obteve aptidão de 98%.

Foi verificado se as regras evoluídas apresentavam também bom desempenho em problemas diferentes daqueles 20 problemas utilizados na evolução da regras. Para isto, as melhores redes evoluídas foram avaliadas sobre 10 novos problemas e tiveram em média uma aptidão de 91,1%. Comparando este resultado contra os 92% obtidos dos 20 problemas originais, conclui-se que as regras evoluídas parecem ser gerais, isto é, que não são específicos apenas para os problemas usados no Algoritmo Genético.

Neste experimento, com redes sem camadas intermediárias, o Algoritmo Genético evoluiu com sucesso regras de aprendizado eficientes. O autor argumenta que, para redes mais complexas, é improvável que métodos evolucionários encontrem regras melhores que o *backpropagation*. Porém, considera que os Algoritmos Genéticos podem encontrar regras de aprendizado para paradigmas de aprendizado não-supervisionado (e.g. aprendizado por reforço) ou para novas classes de arquiteturas de rede (e.g. redes recorrentes).

Esta seção encerra uma etapa da combinação de Algoritmos Genéticos e Redes Neurais, na qual foi dada ênfase às Redes MLP (apêndice A). Na seção seguinte será apresentado modelos que otimizam redes RBF.

4.5 OTIMIZAÇÃO DE REDES DE FUNÇÕES BASE RADIAL

4.5.1 INTRODUÇÃO

Existem várias técnicas para otimizar a topologia da rede RBF, entre as quais, os algoritmos OLS e RAN e similares (ver capítulo 2). Tais algoritmos são métodos de subida do monte (*hill-climbing*). O algoritmo OLS, por exemplo, seleciona passo a passo centros extraídos do conjunto de treinamento. Em cada passo, o OLS usa uma heurística para selecionar o melhor centro naquele passo. O OLS não considera a possibilidade de que, naquele passo, outros centros possam gerar uma melhor topologia quando combinados com os centros selecionados nos passos seguintes. Portanto, o OLS é claramente um método de subida do monte.

Os métodos de subida do monte, apesar de rápidos, podem cair nos mínimos locais do complicado espaço de busca das topologias, isto é, podem gerar soluções que são sub-ótimas. Portanto, tais métodos não podem competir com algoritmos genéticos quando o objetivo é a busca pela solução ótima. Porém, os Algoritmos Genéticos têm a grande desvantagem de tomar muito tempo de processamento, o que nem sempre é desejável na prática. Além disso, como muitas vezes ocorre, uma solução quase ótima pode ser satisfatória na prática.

Basicamente os parâmetros de uma rede RBF a serem otimizados são:

- Número de unidades intermediárias
- Tipo da função base em cada unidade
- Distribuição dos centros no espaço de entrada
- O parâmetro largura da função base
- Pesos das conexões entre a camada intermediária e a camada de saída

Porém, nem todos modelos otimizam todos estes parâmetros. Os pesos, por exemplo, geralmente são otimizados por outras técnicas durante a avaliação do cromossomo na qual é realizada como segue:

- 1) Decodificar o cromossomo
- 2) Treinar os pesos da rede usando uma técnica dos mínimos quadrados (ex. pseudoinversa ou LMS)
- 3) Calcular a aptidão com base em uma função de erro (ex. MSE)

Na seção seguinte será apresentada uma sequência de modelos para otimizar redes RBF.

4.5.2 OTIMIZAÇÃO

Esta seção reúne trabalhos de otimização de redes RBF. Contudo, diferentemente dos modelos para Redes MLP, constatou-se que existem poucos modelos voltados para redes RBF na literatura até o momento.

O MODELO DE BILLINGS E ZHENG (1995)

Neste modelo, os Algoritmos Genéticos foram utilizados para selecionar um subconjunto de vetores de entrada para servir de centros da rede RBF.

A) Cromossomo

O cromossomo é uma simples lista de centros. Os centros provém do conjunto de treinamento, cujos vetores de entrada que são rotulados de 1 a p , onde p é o número de padrões de treinamento. Por exemplo, o cromossomo da Figura 4.17 codifica uma rede com os centros fixados no primeiro, nono, quinto e décimo-primeiro vetores de entrada. A função radial empregada, neste trabalho, foi a *Thin-Plate-Spline*: $\phi(v) = v^2 \log(v)$, utilizada sem o parâmetro largura.

1	9	5	11
---	---	---	----

Figura 4.17. Representação do cromossomo por Billings e Zheng (1995)

B) Operadores Genéticos

Os autores utilizaram os operadores genéticos propostos em (Lucasius e Kateman, 1992) que são adequados para problemas que envolvem seleção de um subconjunto. Um exemplo do operador crossover é apresentado a seguir.

Seja P1 e P2, dois pais selecionados para crossover. Criam-se duas cadeias de bits, T1 e T2, conforme ilustrado na Figura 4.18(a). Um bit da cadeia tem o valor 1 quando o mesmo gene (centro) existe em ambos os pais, caso contrário o bit é zero. Por exemplo, o quarto bit de T1 é igual a 1 porque o correspondente quarto gene de P1 (o centro 8) existe tanto em P1 como em P2. Em seguida, um número aleatório de genes começando pelo fim

dos cromossomos são trocados entre pais. Apenas os genes não comuns (bit = 0) participam da troca. É mostrado na Figura 4.18(b) o resultado da troca de dois genes de P1 com dois genes de P2 gerando os filhos F1 e F2.

Na Figura 4.19, é apresentado outro exemplo na qual troca dois genes de P1 com três genes de P2. Vale observar que o comprimento dos cromossomos foram alterados. Tais alterações de comprimento melhorou a diversidade da população nos experimentos realizados.

O operador de mutação troca genes do cromossomo por centros escolhidos aleatoriamente do conjunto de treinamento, mas com o cuidado de evitar centros repetidos no cromossomo. Foram utilizados ainda um operador de adição e deleção. Eles ajudam a manter a diversidade da população. O operador de adição concatena um número aleatório de genes no fim do cromossomo. O operador de deleção retira um número aleatório de genes a partir de uma posição aleatória no cromossomo.

Após o crossover e mutação, os dois melhores dos quatro indivíduos (dois pais + dois filhos) substituem os dois pais na população.

Figura 4.18. Crossover de Billings e Zheng (1995) – 1º exemplo

Figura 4.19. Crossover de Billings e Zheng (1995) – 2º exemplo

C) aptidão

Muitos trabalhos combinando Redes Neurais com Algoritmos Genéticos utilizam a função objetivo SSE. No entanto, tal função, quando utilizada para otimizar topologias de redes RBF, causa *overffiting*. Pois, o SSE ótimo é quando o número de unidades intermediárias m é igual a p (número de padrões) que resulta em $SSE = 0$ (*overffiting*). Os autores, então, utilizaram como função objetivo o AIC (*Akaike's information criterion*) (Akaike, 1974) (ver seção 2.5) sobre o conjunto de treinamento:

$$J_1 = p_1 \log \left(\frac{1}{p_1} SSE_1 \right) + 4m \quad (4.5.2.1)$$

O AIC fornece um compromisso entre o desempenho da rede e sua complexidade (número de unidades intermediárias m). Contudo, mesmo usando o AIC sobre o conjunto de treinamento ainda ocorreu um certo *overffiting*. Então, uma segunda função objetivo, AIC sobre o conjunto de validação, foi utilizada:

$$J_2 = p_2 \log \left(\frac{1}{p_2} SSE_2 \right) + 4m \quad (4.5.2.2)$$

Onde p_2 é o número de padrões do conjunto de validação. Com isso, a otimização da rede RBF passou a ser um problema de otimização multiobjetivo.

D) Otimização multiobjetiva

Na otimização multiobjetiva foi utilizado o *pareto optimal set* (Goldberg, 1989). Cada indivíduo i contém um par $(J_1^{(i)}, J_2^{(i)})$ referente as duas funções objetivos (4.5.2.1) e (4.5.2.2). Um indivíduo j é dominado pelo indivíduo i se $J_1^{(i)} \leq J_1^{(j)}$ e $J_2^{(i)} \leq J_2^{(j)}$ e pelo menos uma das seguintes desigualdades é verdadeira: $J_1^{(i)} < J_1^{(j)}$ ou $J_2^{(i)} < J_2^{(j)}$. Um

indivíduo i é ótimo quando ele não é dominado por nenhum outro indivíduo. O conjunto de indivíduos não-dominados é chamado de *pareto optimal set*.

E) Ranking

A aptidão foi baseado no *ranking* na qual foi realizado pelo seguinte procedimento:

- Passo 1: Seja S o conjunto de pares $\{(J_1^{(i)}, J_2^{(i)}); i = 1, \dots, \text{POP}\}$, onde POP é o tamanho da população.
- Passo 2: $r \leftarrow 1$.
- Passo 3: Atribuir *rank* r aos indivíduos não dominados por nenhum outro indivíduo da população.
- Passo 4: Remover todos os pares $(J_1^{(i)}, J_2^{(i)})$ do conjunto S tal que o indivíduo i tenha *rank* r .
- Passo 5: $r \leftarrow r+1$.
- Passo 6: Se S é vazio então termine. Caso contrário vá ao passo 3.

F) Experimentos

Nos experimentos foram utilizados padrões gerados de um sistema de nível de água. Uma bomba de água alimentava um frasco cônico que, por sua vez, alimentava um tanque. A entrada da rede era a voltagem da bomba e a saída o nível de água no frasco cônico. Foram utilizados 500 padrões de treinamento, 500 padrões de validação, população de 60 indivíduos e 400 gerações.

Alguns experimentos otimizaram a função objetivo J_1 (4.5.2.1). O Algoritmo Genético melhorou a complexidade da rede pois gerou redes com 21 centros, enquanto métodos convencionais geraram redes com 41 centros. A generalização ainda foi melhorada quando otimizou-se ao mesmo tempo as funções J_1 e J_2 (4.5.2.2).

O MODELO DE MAILLARD E GUERIOT (1997)

Em (Maillard e Gueriot, 1997), o modelo de Billings e Zheng (1995) foi modificado para que os centros não fossem fixados apenas sobre os vetores de entrada. Neste modelo foi experimentado o uso de múltiplos tipos de funções radiais em uma única rede.

A) Cromossomo

Cada unidade intermediária foi representada por 5 genes, Figura 4.20(a). Na Figura 4.20(b), é mostrado a decodificação dos genes. A posição decodificada do vetor centro μ encontra-se no baricentro entre dois vetores de entrada $Id1=(X1,Y1)$ e $Id2=(X2, Y2)$. O peso do baricentro é dado por λ . O tipo da função base da unidade é determinado pelo primeiro gene (T). Foram utilizados os seguintes tipos de funções base: Gaussiana, linear, cubica, *thin-plate-spline*, multiquadrática e multiquadrática inversa (ver Tabela 2.1).

Figura 4.20. Representação do cromossomo por Maillard e Gueriot (1997).

B) Aptidão e Experimentos

A aptidão dos indivíduos foi calculado pelo critério de Akaike (AIC) sobre o conjunto de treinamento, Equação (4.5.2.1). Nos experimentos foi utilizada a série temporal caótica de Mackey-Glass (Mackey e Glass, 1977). Dois tipos de redes RBF foram determinadas pelo Algoritmo Genético: redes utilizando apenas a função Gaussiana e redes utilizando múltiplas funções base. As redes evoluídas com múltiplas funções tiveram menor número de unidades intermediárias e alcançaram na média um erro menor do que as redes evoluídas utilizando apenas funções Gaussiana. O Algoritmo Genético utilizou uma população de 100 indivíduos.

O MODELO DE CARSE E FOGART (1996)

Este modelo otimiza os centros e larguras da rede RBF. A função radial que foi utilizada é uma Gaussiana que possui uma largura para cada componente do vetor centro na qual é dada por:

$$\phi_i(x_1, x_2, \dots, x_n) = \prod_{j=1}^n \exp\left(-\frac{(x_j - \mu_{ij})^2}{2\sigma_{ij}^2}\right) \quad (4.5.2.3)$$

Onde: n é o número de unidades de entrada.

$\mu_i = [\mu_{i1}, \mu_{i2}, \dots, \mu_{in}]^T$ é o vetor centro da i -ésima função base.

$\sigma_i = \{\sigma_{i1}, \sigma_{i2}, \dots, \sigma_{in}\}$ é as larguras da i -ésima função base.

$x = [x_1, x_2, \dots, x_n]^T$ é o vetor de entrada.

A) Cromossomo

Cada gene G_i do cromossomo codifica uma unidade intermediária e é representado por:

$$G_i = (\mu_{i1}, \sigma_{i1}, \dots, \mu_{ij}, \sigma_{ij}, \dots, \mu_{in}, \sigma_{in})$$

Os centros μ_{ij} e larguras σ_{ij} foram representados por números reais.

B) Operadores genéticos

O operador crossover é uma versão modificada do crossover de 2 pontos. Os dois pontos de corte estão contidos nos vetores $\mathbf{a} = \{a_1, a_2, \dots, a_n\}$ e $\mathbf{b} = \{b_1, b_2, \dots, b_n\}$. Os componentes desses vetores são dados por:

$$a_j = \text{MIN}_j + (\text{MAX}_j - \text{MIN}_j) (\text{Rd}_1) \quad (4.5.2.4)$$

$$b_j = a_{1j} + (\text{MAX}_j - \text{MIN}_j) (\text{Rd}_2)^{1/n} \quad (4.5.2.5)$$

Onde Rd_1 e Rd_2 são selecionados aleatoriamente dentro do intervalo $[1,0]$. $[\text{MIN}_j, \text{MAX}_j]$ é o intervalo permitido do componente x_j do vetor de entrada. Após o crossover, o primeiro cromossomo filho conterá os genes do primeiro pai que satisfazem a:

$$\forall j, ((\mu_{ij} > a_j) \text{ AND } (\mu_{ij} < b_j)) \text{ OR } ((\mu_{ij} + \text{MAX}_j - \text{MIN}_j) < b_j) \quad (4.5.2.6)$$

Ao primeiro filho serão também adicionados os genes do segundo cromossomo pai que não satisfazem a condição (4.5.2.6). O segundo cromossomo filho será composto pelos genes que restaram dos dois pais.

Este operador troca as unidades intermediária da rede cujos centros situam-se dentro do hipervolume determinado pelos vetores $\mathbf{a}$ e $\mathbf{b}$. O que conta para este crossover é

posição dos centros no espaço de entrada e não a posição do gene dentro do cromossomo. Isto diminui o problema conhecido como *competing convention problem* (ver o modelo de Neruda, 1995).

Os operadores de mutação são os operadores tradicionais utilizados em números reais. O operador *creep* (seção 3.5) é aplicado aos centros e larguras das funções radiais. Além destes, foram utilizados operadores para adicionar e remover unidades intermediárias.

C) Aptidão e Experimentos

Dois problemas foram utilizados nos experimentos. No primeiro, utilizou-se padrões gerados da função $y = \text{seno}(20x^2)$. A população foi de 50 indivíduos por 50 gerações. Em cada geração, os piores 40 indivíduos eram substituídos. Os valores dos centros e larguras foram escolhidos nos intervalos $[1,0]$ e $[0,1/\text{NúmeroDeUnidades}]$, respectivamente. O crossover modificado foi comparado com os crossovers tradicionais de 1 e 2 pontos e mostrou melhor desempenho. As redes evoluídas superaram redes com os centros igualmente espaçados. Os pesos foram determinados pelo algoritmo LMS. Foi constatado que utilizando poucos ciclos do LMS nas primeiras gerações e aumentando os ciclos conforme o avanço das gerações obtém-se um aprendizado mais rápido.

No segundo experimento utilizou-se a série temporal caótica de Mackey-Glass (Mackey e Glass, 1977). Os experimentos foram realizados com 20, 40, 60, 80 e 100 unidades intermediárias com 200 gerações. Uma heurística foi usada para o número de ciclos e a taxa de aprendizado do LMS:

- $\text{NoCiclos} = 5 + \text{NoGeração}/10$
- $\text{TaxaAprendizado} = 0.6 - 0.002 * \text{NoGeração}$

Aqui, novamente, o crossover modificado superou os crossovers tradicionais. Os resultados finais foram comparáveis aos resultados obtidos em (Whitehead e Choate, 1994).

O MODELO DE NERUDA (1995)

Neste trabalho foi abordado o problema do mapeamento estrutural-funcional (ver também seção 4.2.4) em redes RBF e proposto uma representação de cromossomo e operadores genéticos para tratar com este problema.

São mostradas na Figura 4.21 duas redes com topologias diferentes. Estas redes têm as mesmas unidades intermediárias, porém situadas em posições relativas diferentes. Elas, depois de treinadas, apresentam o mesmo desempenho, mas podem ser codificadas no cromossomo de maneira totalmente diferente. Tais redes são estruturalmente diferentes, mas funcionalmente equivalentes. Este problema aumenta consideravelmente o espaço de

busca, pois gera sem necessidade cromossomos diferentes. A seguir será apresentado um tratamento mais formal sobre redes funcionalmente equivalentes.

Figura 4.21. Redes RBF funcionalmente equivalentes.

A) Redes RBF funcionalmente equivalentes

Considere a parametrização da rede RBF dada pelo conjunto: $P = \{(w_i, \sigma_i, \mu_i); i = 1, \dots, m\}$ onde w_i é o peso, σ_i é a largura e μ_i é o vetor centro da unidade intermediária i . Pode ser mostrado que duas redes RBF com parametrizações $P = \{(w_i, \sigma_i, \mu_i); i = 1, \dots, m\}$ e $P' = \{(w'_i, \sigma'_i, \mu'_i); i = 1, \dots, m\}$, respectivamente, são funcionalmente equivalentes se e somente se existe uma permutação π no conjunto $\{1, \dots, m\}$, tal que $w_i = w'_{\pi(i)}$, $\sigma_i = \sigma'_{\pi(i)}$ e $\mu_i = \mu'_{\pi(i)}$ para cada $i \in \{1, \dots, m\}$.

Uma parametrização $P = \{p_1, p_2, p_3, p_4\}$ onde $p_i = (w_i, \sigma_i, \mu_i)$ possui $4! = 24$ permutações, ou seja, 24 redes RBF funcionalmente equivalentes. Consequentemente, a escolha de uma única parametrização (**parametrização canônica**) para representar toda uma classe de redes RBF funcionalmente equivalentes reduzirá o espaço de busca. A parametrização canônica é definida como segue:

Seja p e q dois elementos de P onde $p = (p_1, p_2, \dots, p_{m+2})$ e $q = (q_1, q_2, \dots, q_{m+2})$. O autor definiu a relação $p \prec q$, que significa que p precede q se existe um índice $i \in \{1, \dots, m\}$, tal que $p_j = q_j$, para $j < i$ e $p_i < q_i$. Uma parametrização $P = \{p_1, p_2, \dots, p_m\}$ é canônica se:

$$p_1 \prec p_2 \prec \dots \prec p_m \quad (4.5.2.7)$$

B) O Cromossomo

O cromossomo é uma simples parametrização canônica, conforme mostrado na Figura 4.22.

Figura 4.22. Representação do cromossomo por Neruda (1995)

C) Operadores Genéticos

O operadores genéticos foram projetados para preservar a parametrização canônica. Na mutação, um elemento p_i é escolhido aleatoriamente. Como a relação de ordenamento $p_{i-1} < p_i < p_{i+1}$ deve ser mantida para conservar a parametrização canônica, a mutação deve ser realizada respeitando os limite inferior p_{i-1} e superior p_{i+1} .

Por exemplo, considere que o elemento $p_3 = (2, 3, 4, 9)$ da Figura 4.23, foi escolhido aleatoriamente para mutação. Os valores das duas primeiras posições, representado pelos números 2 e 3, não podem ser alterados pois não manteriam o ordenamento $p_2 < p_3 < p_4$. O número 4 pode ser alterado, mas está limitado ao intervalo $[1, 8]$. O número 9 pode ser alterado sem restrições, pois não modifica o ordenamento $p_2 < p_3 < p_4$.

No operador crossover, dados dois cromossomos $P = \{p_1, p_2, \dots, p_m\}$ e $Q = \{q_1, q_2, \dots, q_m\}$ escolhe-se uma posição de corte aleatória i tal que o filho ainda satisfaça a condição (4.5.2.7), ou seja, a condição $p_i < q_{i+1}$ deve ser mantida. Um exemplo é mostrado na Figura 4.24.

Figura 4.23. Um exemplo do cromossomo de Neruda (1995)

P =	<table><tr><td>p_1</td><td>$\dots$</td><td>p_{i-1}</td><td>p_i</td><td>p_{i+1}</td><td>$\dots$</td><td>p_m</td></tr></table>	p_1	$\dots$	p_{i-1}	p_i	p_{i+1}	$\dots$	p_m
p_1	$\dots$	p_{i-1}	p_i	p_{i+1}	$\dots$	p_m		
Q =	<table><tr><td>q_1</td><td>$\dots$</td><td>q_{i-1}</td><td>q_i</td><td>q_{i+1}</td><td>$\dots$</td><td>q_m</td></tr></table>	q_1	$\dots$	q_{i-1}	q_i	q_{i+1}	$\dots$	q_m
q_1	$\dots$	q_{i-1}	q_i	q_{i+1}	$\dots$	q_m		
Filho	<table><tr><td>p_1</td><td>$\dots$</td><td>p_{i-1}</td><td>p_i</td><td>q_{i+1}</td><td>$\dots$</td><td>q_m</td></tr></table>	p_1	$\dots$	p_{i-1}	p_i	q_{i+1}	$\dots$	q_m
p_1	$\dots$	p_{i-1}	p_i	q_{i+1}	$\dots$	q_m		

Figura 4.24. Crossover de Neruda (1995)

D) aptidão

A aptidão foi dada por $C - SSE$, onde C é uma constante que pode ser uma estimativa do erro máximo da rede ou simplesmente o SSE máximo da população.

E) Considerações finais

Este modelo reduz o espaço de busca. Caso existam m unidades intermediárias em uma rede, irão existir $m!$ permutações de unidades intermediárias. Logo, o espaço de busca é reduzido de $1/m!$ do espaço inteiro de topologias de redes RBF. Neste trabalho, porém, não foi apresentado nenhum resultado experimental do modelo proposto.

Um breve resumo de outras abordagens será apresentado a seguir.

4.5.3 OUTROS MODELOS

Em (Whitehead e Choate, 1994) o Algoritmo Genético evoluiu “curvas de preenchimento de espaço” para determinar os centros das funções bases. A idéia básica desta representação é o mapeamento do espaço m -dimensional dos centros situados ao longo das “curvas de preenchimento de espaço” para um espaço unidimensional na qual é codificado o cromossomo.

O mapeamento é realizado por um algoritmo denominado *dovetail algorithm*. A descrição deste algoritmo pode ser encontrada no próprio trabalho de Whitehead e Choate (1994) ou mais detalhadamente em (Patrick et al, 1968). Este modelo foi aplicado na previsão de séries temporais caóticas de Mackey-Glass (Mackey e Glass, 1977). Os resultados foram melhores do que aqueles obtidos pelo algoritmo k-médias.

Em (Whitehead e Choate, 1996) os centros e as larguras das funções bases foram evoluídas através de um elegante Algoritmo Genético cooperativo-competitivo. Neste algoritmo, cada indivíduo da população codificava apenas uma função base, ou seja, uma unidade intermediária da Rede. A população inteira representava apenas uma rede RBF. Os

indivíduos da população competiam entre si ao mesmo tempo em que cooperam para melhorar o desempenho global da rede RBF representada pela população.

Este modelo foi aplicado no conhecido conjunto padrões IRIS (Murphy e Aha, 1994) para classificação de padrões e na previsão de séries temporais caóticas de Mackey-Glass (Mackey e Glass, 1977). Os resultados também foram melhores do que os resultados obtidos pelo algoritmo k-médias.

Em (Chen et al, 1995) foi utilizado o algoritmo OLS com regularização (ver capítulo 2). O Algoritmo Genético foi apenas utilizado para evoluir as larguras e o parâmetro de regularização λ , uma vez que o algoritmo OLS determina automaticamente o número de unidades intermediárias e as posições dos centros.

A seguir alguns considerações serão apresentados sobre o que foi visto neste capítulo.

4.6 COMENTÁRIOS FINAIS

Neste capítulo foram mostradas as três principais maneiras de combinar os Algoritmos Genéticos com Redes Neurais: no treinamento da rede, na otimização da arquitetura e para evoluir a regra de aprendizado. Na otimização da rede, foi visto que as abordagens utilizadas se dividem em dois grupos: as abordagens que codificam a estrutura da Rede Neural (pesos, conexões etc..) diretamente no cromossomo e as abordagens que incluem no cromossomo regras ou algum tipo de informação que indiretamente servirá para a construção da rede.

Foi visto que as Redes Neurais geradas pelos Algoritmos Genéticos possuem conexões poucos usuais. Conexões podem partir da camada de entrada e ir direto para a camada de saída sem passar por camadas internas. Portanto, as topologias convencionais nem sempre são as melhores. Foi observado também, que é possível otimizar a Rede Neural em vários aspectos: na velocidade de treinamento, capacidade de generalização, tamanho da topologia, etc...

O ponto crítico de tais modelos é o cálculo do valor de aptidão, que geralmente demanda muito tempo de processamento. Contudo, em algumas aplicações poderá ser compensador utilizar um Algoritmo Genético para produzir redes com topologias otimizadas. Por exemplo, tais redes poderiam ser implementadas em hardware com um custo menor.

A última seção deste capítulo foi dedicado especialmente as Redes Neurais RBF. A otimização de redes RBF é mais rápida que a de Redes MLP. Isto deve-se ao fato de que a determinação dos pesos em redes RBF é um problema de otimização linear (resolvido com o rápido algoritmo LMS) enquanto em Redes MLP é não-linear (resolvido com o

backpropagation ou similares), o que aumenta de forma considerável o tempo de processamento.

A conclusão deste estudo é que não há um modelo padrão a ser adotado. Portanto, a análise de vários modelos pode ser útil para aqueles que estão desenvolvendo o seu próprio Algoritmo Genético na área de Redes Neurais ou até mesmo em outras áreas. Na próxima seção será apresentado um novo modelo aplicado a redes RBF, onde novas idéias são propostas juntamente com o aproveitamento e aperfeiçoamento de algumas idéias deste capítulo.

5

UM NOVO ALGORITMO GENÉTICO

"Seja qual seja seu problema, o trabalho será sempre a sua base de solução. Não existe processo de angústia que não se desfaça ao toque do trabalho. Diante de qualquer sofrimento, o trabalho é o nosso melhor caminho de libertação"
(Espírito André Luiz)

"Estude a si mesmo, observando que o auto-conhecimento traz humildade e sem humildade é impossível ser feliz"
(Espírito André Luiz)

"Enriquecer o trabalho profissional, adquirindo conhecimentos novos, é simples dever"
(Espírito André Luiz)

5.1 DESCRIÇÃO

Este capítulo descreve o Algoritmo Genético proposto nesta dissertação. Este algoritmo otimiza os seguintes parâmetros de uma rede RBF:

- Número de centros;
- Posição dos centros;
- Larguras de cada centro.

Os pesos são otimizados utilizando a pseudoinversa. Para reduzir o espaço de busca, os centros foram restritos a regiões onde há aglomeração de padrões. Um exemplo é mostrado na Figura 5.1, onde as posições dos centros estão limitados a uma das três regiões.

Figura 5.1. Regiões permitidas aos centros

5.1.1 REPRESENTAÇÃO

O cromossomo utilizado codifica um grupo de unidades intermediárias cujos centros estão situados em determinadas regiões do espaço de entrada. Considere $\{R_k; k = 1 \dots n_R\}$ o conjunto de tais regiões, que são definidas como segue:

A) Definição das regiões

Cada região R_k é um hipercubo com ponto central $\mathbf{c}_k$ e lado l_k . As regiões são obtidas a partir dos aglomerados de vetores de entrada $\{S_k; k = 1 \dots n_R\}$ gerados pelo algoritmo k-médias. O algoritmo k-médias é executado sobre o conjunto de treinamento $\{(\mathbf{x}_i, t_i); i = 1 \dots p\}$, onde $\mathbf{x}_i \in \mathcal{R}^n$ representa o vetor de entrada e t_i é a saída desejada para $\mathbf{x}_i$. O ponto central $\mathbf{c}_k$ da região R_k é dado por:

$$\mathbf{c}_k = \frac{1}{n_k} \sum_{\mathbf{x}_i \in S_k} \mathbf{x}_i \quad (5.1.1.1)$$

Onde n_k é o número de padrões que pertencem ao aglomerado S_k . O lado l_k da região R_k é dado por:

$$l_k = \max(\|\mathbf{x}_i - \mathbf{c}_k\|), \text{ onde } \mathbf{x}_i \in S_k \quad (5.1.1.2)$$

B) Cromossomo

Cada gene G_j codifica uma unidade intermediária na qual é dado por:

$$G_j = (k, \sigma_j, \mu_{j1}, \mu_{j2}, \dots, \mu_{jn}) \quad (5.1.1.3)$$

que representa a j -ésima função base de largura σ_j cujo centro $\mu_j = [\mu_{j1}, \mu_{j2}, \dots, \mu_{jn}]^T \in R_k$. Cada região R_k pode conter um número ilimitado de centros (genes G_j). O cromossomo é uma simples lista de genes G_j de comprimento variável.

C) Codificação e decodificação

Por conveniência, os parâmetros do gene G_j (5.1.1.3) assumem somente valores no intervalo $[0,1]$. Na decodificação do cromossomo, a conversão do centro μ_j do gene $G_j \in R_k$ para o seu real valor é dado por:

$$\mu'_{ji} = c_{ki} + \frac{l_k}{2}(\mu_{ji} - 0,5) \quad (5.1.1.4)$$

Onde c_{ki} é o i -ésimo componente do vetor c_k . A largura σ_j é multiplicada por um fator de escala α :

$$\sigma'_j = \alpha \sigma_j \quad (5.1.1.5)$$

O fator de escala α é definido heurísticamente como a maior distância separando dois padrões de treinamento, ou seja:

$$\alpha = \max(\|x_i - x_j\|), \text{ para } i = 1 \dots p, j = 1 \dots p \quad (5.1.1.6)$$

5.1.2 OPERADORES GENÉTICOS

Os operadores genéticos utilizados nesta Dissertação foram: um novo operador de crossover e quatro tipos de operadores de mutação.

A) Crossover

Considere os dois cromossomos PAI1 e PAI2 mostrados na Figura 5.2. Neste exemplo, os centros estão situados em um espaço unidimensional e o número de regiões é $n_R = 3$. A primeira vista, é possível utilizar os crossover's tradicionais de 1-ponto e 2-pontos. Surge,

porém, um problema: o cromossomo FILHO1 possui os genes (1; 0,3; 0,1) e (2; 0,4; 0,2) em duplicata (ver também seção 4.2.4 e modelo de Neruda na seção 4.5.2).

Uma abordagem mais eficiente seria aplicar o crossover efetuando troca de regiões entre cromossomos. Deste modo, evita-se o aparecimento de genes duplicados devido ao crossover. Por pertencerem à mesma região, tais genes não poderiam fazer parte do mesmo filho, já que seriam trocados entre os pais. O novo crossover, apresentado na Figura 5.3, é um tipo de crossover uniforme entre regiões. Um exemplo é mostrado na Figura 5.4, que gerou filhos sem genes duplicados.

PAI1	(1; 0,3; 0,1)(2; 0,4; 0,2)(3; 0,9; 0,1)
PAI2	(1; 0,2; 0,4)(3; 0,5; 0,7)(2; 0,4; 0,2)(1; 0,3; 0,1)
FILHO1	(1; 0,3; 0,1)(2; 0,4; 0,2)(2; 0,4; 0,2)(1; 0,3; 0,1)
FILHO2	(1; 0,2; 0,4)(3; 0,5; 0,7)(3; 0,9; 0,1)

Figura 5.2. Crossover de 1 ponto

Gerar uma máscara de bits aleatoriamente: $(b_1, b_2, \dots, b_{n_R})$

para $k=1$ **até** n_R **faça**

se $(b_k = 1)$ **então**

 Copiar todos os genes G_j de PAI1 tal que $G_j \in R_k$ para FILHO1

senão

 Copiar todos os genes G_j de PAI2 tal que $G_j \in R_k$ para FILHO1

fim para

Copiar todos os genes G_j que sobraram de PAI1 e PAI2 para FILHO2.

Figura 5.3. Novo crossover

Máscara de bits: 011

Figura 5.4. Novo crossover

B) Operadores de mutação

Quatro operadores de mutação foram utilizados:

- Mutação tradicional: substitui o valor de σ , $\mu_1, \mu_2, \dots, \mu_n$ por um valor no intervalo $[0,1]$
- Mutação *creep*: soma aos parâmetros σ , $\mu_1, \mu_2, \dots, \mu_n$ um pequeno valor aleatório extraído do intervalo $[-\text{RANGE}, +\text{RANGE}]$. Onde RANGE é definido pelo usuário
- Adição: adiciona ao cromossomo um gene gerado aleatoriamente
- Remoção: remove do cromossomo um gene situado em uma posição aleatória

5.1.3 APTIDÃO

O erro SSE sobre o conjunto de treinamento é utilizado como função objetivo de muitos Algoritmos Genéticos. Contudo, para o presente trabalho o SSE é inadequado. O melhor valor do SSE para o conjunto de treinamento é obtido quando o número de unidades intermediárias m é igual ao número de padrões de treinamento p . Tal topologia tem ótimo desempenho sobre os padrões de treinamento: $\text{SSE} = 0$ (*overfitting*). Porém, possui fraca generalização.

A) Escolha da função objetivo

Há uma variedade de critérios que fornecem um compromisso entre desempenho e generalização (ver seção 2.5, item C). Em um trabalho anterior foi utilizado o *Akaike's Information Criterion* (AIC) (Lacerda e Carvalho, 1997). No presente trabalho foi experimentado o *Generalized Crossvalidation* (GCV), Equação 2.5.1. Em estudos pilotos, o GCV foi aplicado somente sobre o conjunto de treinamento. Tal abordagem causou *overfitting*. Em vista disto, foram utilizadas duas funções objetivos. A primeira foi o GCV sobre conjunto de treinamento:

$$T = \frac{m \text{SSE}_T}{(p_T - m)^2} \quad (5.1.3.1)$$

Onde p_T é número de padrões do conjunto de treinamento. A segunda função objetivo foi o GCV sobre conjunto de validação:

$$V = \frac{m \text{SSE}_V}{(p_V - m)^2} \quad (5.1.3.2)$$

onde p_V é número de padrões do conjunto de validação. A otimização com duas funções objetivos é apresentada a seguir.

B) Otimização multiobjetivo

A cada indivíduo i é associado um par (T_i, V_i) referente às duas funções objetivos (5.1.3.1) e (5.1.3.2). Um par (T_j, V_j) é **dominado** pelo par (T_i, V_i) se $T_i \geq T_j$ e $V_i \geq V_j$ e pelo menos uma das seguintes desigualdades é verdadeira: $T_i > T_j$ ou $V_i > V_j$. Um par (T_i, V_i) é ótimo quando ele não é dominado por nenhum outro par. O conjunto de todos os pares ótimos é denominado de *pareto optimal set* (Goldberg, 1989). A meta é encontrar o conjunto de pares ótimos.

C) Ranking

A aptidão é calculada pelo *Ranking*, que é definido pelo seguinte procedimento de ordenação:

Passo 1: Seja S o conjunto de pares $\{(T_i, V_i); i=1,...,POP\}$, onde POP é o tamanho da população.

Passo 2: $r \leftarrow 1$

Passo 3: Atribuir *rank* r aos indivíduos i cujo par (T_i, V_i) não é dominado por nenhum outro par.

Passo 4: Remover todos os pares (T_i, V_i) do conjunto S tal que o indivíduo i possua *rank* r .

Passo 5: $r \leftarrow r + 1$

Passo 6: Se S é vazio então terminar. Caso contrário, ir ao passo 3.

Algumas informações adicionais serão apresentadas a seguir.

5.1.4 IMPLEMENTAÇÃO

O Algoritmo Genético utilizado (implementado em linguagem C) foi do tipo *steady-state* (ver seção 3.3.5). Isto é, a cada geração somente dois filhos são gerados. Os dois filhos substituem os pais, caso sejam melhores do que eles (i.e. não dominados pelos pais). De outro modo, os filhos são descartados.

Apesar do algoritmo ter sido projetado para determinar a topologia automaticamente, o número de centros foi limitado a um certo número máximo, definido pelo usuário. Tal procedimento mostrou-se eficiente na prática, pois reduz significativamente o espaço de busca. Contudo, isto só é válido quando o usuário já sabe de antemão que o número de unidades não ultrapassa um certo limite.

A seguir serão apresentados os experimentos realizados:

5.2 EXPERIMENTOS

Três problemas (dois de regressão e um de classificação) foram utilizados nos experimentos. Os experimentos foram realizados de acordo com os critérios propostos em (Prechelt, 1994).

5.2.1 EXPERIMENTO I: REGRESSÃO

A) Polinômio de Hermite

O polinômio de Hermite utilizado neste experimento é dado por:

$$f(x) = 1.1(1 - x + 2x^2) \exp\left(-\frac{x^2}{2}\right) \quad (5.2.1.1)$$

Este polinômio foi utilizado em (Orr, 1995) e (Kadirkamanathan e Niranjan., 1993) para testar o desempenho dos algoritmos RFS (este algoritmo é o mesmo que o OLS com regularização, ver seção 2.8) e RAN-EKF (seção 2.9), respectivamente.

B) Conjuntos de padrões

Os conjuntos de treinamento e teste foram gerados de modo idêntico àqueles dos trabalhos mencionados no item A.

- **Conjunto de treinamento:** 40 padrões aleatoriamente escolhidos no intervalo $[-4, +4]$ com ruído.
- **Conjunto de teste:** 200 padrões sem ruído igualmente espaçados no intervalo $[-4, +4]$.

O Algoritmo Genético utilizado precisa ainda de um conjunto de validação. Assim, este conjunto foi gerado do mesmo modo que o conjunto de treinamento:

- **Conjunto de validação:** 40 padrões aleatoriamente escolhidos no intervalo $[-4, +4]$ com ruído.

C) Experimentos

Uma execução do Algoritmo Genético utilizando os parâmetros mostrados na Tabela 5.1 sobre um conjunto de treinamento com ruído de variância igual a 0,001 gerou um conjunto

de redes ótimas, segundo o critério do *pareto optimal set*. Tal conjunto é mostrado na Tabela 5.2 ordenado pela campo GCV sobre o conjunto de treinamento. O erro RMS é dado por (2.6.4). A mediana das redes (isto é, a rede situada na linha da Tabela 5.2 que tem a mesma quantidade de linhas acima e abaixo dela) foi o critério utilizado para escolher a melhor rede de uma execução do Algoritmo Genético.

Portanto, para esta execução, a melhor rede foi a de número 3 na Tabela 5.2. Tal rede apresentou um erro RMS sobre o conjunto de teste igual a 0,02890. Para acessar apropriadamente o erro sobre o conjunto de teste, foram realizadas 30 execuções. Os resultados são mostrados na Tabela 5.3 com média e desvio padrão. Três *outliers* foram retirados. Prechelt (1994) recomenda que até no máximo 10% dos valores sejam removidos. Provavelmente, eles foram consequência de convergência prematura do Algoritmo Genético.

Como informado anteriormente, este resultado foi sobre um conjunto de treinamento com ruído de variância 0,001. Este procedimento foi repetido para vários níveis de ruído sobre o conjunto de treinamento. Os resultados estão resumidos nas Tabelas 5.4 e 5.5, onde são comparados com os resultados dos algoritmos RFS e RAN-EKF extraídos da Figura 4 de (Orr, 1995).

D) Discussão

Conforme mostrado nas Tabelas 5.4 e 5.5, o Algoritmo Genético proposto apresentou melhores resultados em quase todos os níveis de ruído do que os outros algoritmos. O resultado mais surpreendente é a capacidade do Algoritmo Genético em gerar redes de tamanhos mínimos. Contudo, vale frisar que os algoritmos convencionais foram treinados apenas com o conjunto de treinamento, enquanto o Algoritmo Genético usou os conjuntos de treinamento e validação. Ou seja, a comparação não foi perfeita. Os experimentos seguintes não terão este problema.

A seguir, o desempenho Algoritmo Genético será investigado em um problema real: Avaliação de Risco de Crédito.

Tabela 5.1 Os parâmetros do AG para o polinômio de Hermite

População	50
Número de gerações	50
Número de regiões	4
Taxa de crossover	0.8
Taxa de mutação	0.1
Taxa de <i>creep</i>	0.1
Intervalo do operador <i>creep</i>	[-0.01,+0.01]
Taxa de adição	0.05
Taxa de remoção	0.05
Número máximo de centros permitidos	15
Função de ativação	GAUSSIAN

Tabela 5.2. O *pareto optimal set* de uma das execuções do AG

Rede	Funções Base	Erro RMS Treinamento	GCV Treinamento	Erro RMS Validação	GCV Validação
1	9	0,01880	0,000589	0,03682	0,00226
2	9	0,01883	0,000590	0,03644	0,00221
3	9	0,01884	0,000591	0,03643	0,00221
4	8	0,01956	0,000598	0,03752	0,00220
5	8	0,01960	0,000600	0,03735	0,00218
6	9	0,01944	0,000629	0,03600	0,00216

Tabela 5.3. Várias execuções do AG para o polinômio de Hermite

Execução	Funções base	RMS Teste
1	8	0,014689
2	9	0,028899
3	10	0,022017
4	9	0,026044
5	8	0,018778
6	9	0,023478
7	13	0,076331
8	11	0,045060
9	13	0,019482
10	9	0,021096
11	9	0,022208
12	9	0,092313
13	8	0,019447
14	7	0,023000
15	7	0,027292
16	10	0,017425
17	8	0,022012
18	10	0,024222
19	8	0,019460
20	8	0,020354
21	6	0,017705
22	11	0,023749
23	9	0,018575
24	7	0,018053
25	8	0,025921
26	7	0,018449
27	9	0,018466
28	12	0,019967
29	9	0,028329
30	9	0,021373

Média	8,778	0,021500
Desvio Padrão	1,553	0,003589

Tabela 5.4. Comparação do número de funções base no polinômio de Hermite

Variância do ruído	RANEKF	RFS	Algoritmo Genético	
			Média	Desvio Padrão
0,00010	10	13,3	10,67	1,41
0,00030	11	13,4	10,11	1,58
0,00100	14	12,4	8,78	1,55
0,00300	14	11,6	8,41	1,55
0,01000	17	11,0	7,59	1,22
0,03000	26	8,8	6,70	1,17
0,10000	29	6,5	5,25	1,00

Tabela 5.5. Comparação do RMS no polinômio de Hermite

Variância do ruído	RANEKF	RFS	Algoritmo Genético	
			Média	Desvio Padrão
0,00010	0,0223	0,0085	0,0087	0,0021
0,00030	0,0262	0,0145	0,0131	0,0040
0,00100	0,0325	0,0236	0,0215	0,0036
0,00300	0,0543	0,0343	0,0378	0,0081
0,01000	0,1000	0,0666	0,0594	0,0127
0,03000	0,1619	0,1078	0,1061	0,0202
0,10000	0,3046	0,1841	0,1550	0,0438

5.2.2 EXPERIMENTO II: CLASSIFICAÇÃO

A) Avaliação de Risco de Crédito

Os experimentos de Avaliação de Risco de Crédito utilizaram a base de dados *card1* extraído do conjunto *benchmark* de problemas para Redes Neurais PROBEN1 (Prechelt, 1994). O *card1* contém um conjunto de dados de Avaliação de Risco de Crédito na área de Cartões de Crédito. Nele, os padrões são classificados em duas categorias: aprovação ou não aprovação de uma solicitação de cartão de crédito. O Apêndice B traz diversas informações sobre o problema de Avaliação de Risco de Crédito em geral..

O *card1* é composto de 690 padrões com 15 atributos de entrada e um atributo de classe. Destes padrões, 345 padrões estão no conjunto de treinamento, 173 no conjunto de validação e 172 no conjunto de teste.

B) Experimentos

O Algoritmo Genético foi executado variando o número de regiões n_R e o limite máximo do número de centros. Os outros parâmetros são mostrados na Tabela 5.6. Os resultados de 8 execuções do Algoritmo Genético são mostrados na Tabela 5.7. A Percentagem de Erro Quadrático (do inglês, *Squared Error Percentage*) foi utilizada para ser consistente com os resultados de (Prechelt, 1994). A Percentagem de Erro Quadrático é dada por:

$$E = \frac{o_{\max} - o_{\min}}{pc} \sum_{i=1}^p \|t^{(i)} - y^{(i)}\|^2$$

onde c é o número de unidades de saída; $o_{\max}$ e $o_{\min}$ são os valores máximo e mínimo das saídas desejadas. A melhor rede foi resultado da terceira execução, uma vez que apresentou a melhor aptidão (i.e. GCV sobre treinamento e validação).

A rede RBF otimizada pelo Algoritmo Genético foi comparada com a rede MLP cujo desempenho foi extraído da Tabela 3 de (Prechelt, 1994). Foi também comparada com a rede RBF otimizada pelo algoritmo OLS, cujo desempenho foi extraído da Figura 5(b) de (Lacerda e Carvalho, 1998). Por fim, os resultados da comparação são mostrados na Tabela 5.8. A última coluna desta tabela apresenta a percentagem de padrões do conjunto de teste que foram incorretamente classificados.

C) Discussão

Diferente dos experimentos da seção anterior, aqui todos os algoritmos utilizaram os conjunto de treinamento e validação. O a rede RBF otimizada pelo Algoritmo Genético superou a rede MLP e a rede RBF otimizada pelo Algoritmo OLS. A Tabela 5.7 mostra que a Percentagem de Erro Quadrático da redes Genéticas (até mesmo as piores) sobre o conjunto de validação são menores que nas redes MLP e OLS (na Tabela 5.8). Esta boa generalização deve-se ao fato de que o erro sobre o conjunto de validação foi otimizado indiretamente através da função objetivo GCV sobre o conjunto de validação, o que resultou em erros bastante baixos.

Novamente o Algoritmo Genético foi capaz de gerar redes surpreendentemente reduzidas. A Tabela 5.7 mostra uma rede com apenas 8 funções base com excelente desempenho. A grande desvantagem, porém, é o excessivo tempo de processamento. Enquanto o OLS treinou redes em alguns minutos, o Algoritmo Genético levou cerca de 7 horas para completar uma execução sobre o mesmo computador.

Tabela 5.6. Os parâmetros do Algoritmo Genético para o problema *cardl*

População	100
Número de Gerações	200
Taxa de Crossover	0.8
Taxa de Mutação	0.01
Taxa de Creep	0.05
Intervalo do Operador Creep	[-0.01,+0.01]
Taxa de adição	0.05
Taxa de remoção	0.05
Função de ativação	GAUSSIAN

Tabela 5.7. Resultados do AG sobre o problema *cardl*

Número Da execução	Regiões	Máximo número de centros	Funções base	Percentagem de Erro Quadrático Treinamento	Percentagem de Erro Quadrático Validação	GCV Treinamento	GCV validação
1	15	30	15	9,50	7,11	0,208	0,171
2	10	20	12	9,46	6,13	0,203	0,142
3	5	20	8	10,32	7,38	0,216	0,162
4	5	20	10	9,81	6,62	0,208	0,149
5	5	20	13	9,45	6,71	0,204	0,157
6	8	10	9	10,53	7,66	0,222	0,170
7	8	10	8	10,49	7,41	0,220	0,163
8	8	10	9	10,36	7,68	0,218	0,171
Média	-	-	10,50	9,99	7,09	0,212	0,161
Desv.pad.	-	-	2,56	0,48	0,55	0,008	0,011

Tabela 5.8. Comparação de desempenho no problema *heartal*

Rede	Funções base	Percentagem de Erro Quadrático Treinamento	Percentagem de Erro Quadrático Validação	Percentagem de Erro Quadrático Teste	Classificação Teste
MLP	-	8,92	8,89	10,53	13,64 %
AG	12	9,46	6,13	11,20	11,63 %
OLS	21	9,65	8,39	10,36	12,21 %

5.2.3 EXPERIMENTO III: REGRESSÃO

A) Problema de diagnóstico médico

Nestes experimentos utilizou-se novamente um conjunto de dados do PROBEN1 (Prechelt, 1994). O *heartal* é um problema de diagnóstico de doenças do coração com 920 padrões com 13 atributos de entrada e uma de saída com valores assumindo 5 níveis. Dos 920 padrões, somente 299 estão completos. Os outros padrões têm algum valor de atributo faltando. Isto torna o *heartal* um problema difícil. Segundo Prechelt, (1994), as atributos sem valor foram representados pela adição de uma entrada extra nos padrões com valor 1 se o valor do atributo está faltando e zero caso contrário. Os padrões foram divididos em 345 para treinamento, 173 para validação e 172 para teste.

B) Experimentos

O Algoritmo Genético foi executado três vezes com os parâmetros da Tabela 5.9. Os resultados obtidos estão na Tabela 5.10. Dos três resultados, dois ficaram empatados com respeito ao GCV (os resultados das execuções 1 e 3). A média dessas duas execuções foi utilizada para fins de comparação. A comparação utilizou-se dos algoritmos M-RAN, RAN-EKF e RAN cujo desempenho foi extraído da Tabela 1 de (Yingwei et al, 1997) e a rede MLP cujo desempenho foi extraído da Tabela 11 de (Prechelt, 1994). A comparação é mostrada da Tabela 5.11.

C) Discussão

Os resultados mostram que o Algoritmo Genético novamente superou os algoritmos convencionais e que as redes RBF geradas possuem tamanho mínimo. Mais uma vez os Algoritmos Genéticos demonstram ser um eficiente método de busca.

Tabela 5.9. Parâmetros do AG para o problema *heartal*

População	100
Número de gerações	200
Número de regiões	3
Taxa de crossover	0,8
Taxa de mutação	0,01
Taxa de <i>creep</i>	0,05
Intervalo do operador <i>creep</i>	[-0.01,+0.01]
Taxa de adição	0,05
Taxa de remoção	0,05
Número máximo de centros permitidos	7
Função de ativação	GAUSSIAN

Tabela 5.10. Resultados do AG sobre o problema *heartal*

Número da execução	Funções base	GCV Treinamento	GCV Validação	Percentagem de Erro Quadrático Treinamento	Percentagem de Erro Quadrático Validação	Percentagem de Erro Quadrático Teste
1	5	0,0401	0,0427	3,13	3,27	3,63
2	3	0,0408	0,0425	3,22	3,31	-
3	6	0,0402	0,0419	3,14	3,18	3,91

Tabela 5.11. Comparação de desempenho no problema *heartal*

Rede	Funções base	Percentagem de Erro quadrático Teste
AG	5,5	3,77
M-RAN	8	4,14
RAN-EKF	13	4,21
RAN	13	5,20
MLP	-	4,55

5.3 COMENTÁRIOS FINAIS

Os experimentos realizados levaram a três conclusões:

- Os Algoritmos Genéticos foram capazes de gerar redes com bom desempenho e generalização;
- Os Algoritmos Genéticos foram capazes gerar redes de tamanho reduzido mantendo o bom desempenho e generalização;
- Os Algoritmos Genéticos em otimização de redes neurais consomem excessivo tempo de processamento.

As duas primeiras conclusão comprovam que o Algoritmos Genéticos são eficientes em encontrar soluções globalmente ótimas (ou soluções próximas da ótima). Algoritmos

convencionais, como OLS e RAN, determinam a topologia baseado em heurísticas, o que limita a capacidade de encontrar soluções globalmente ótimas.

A última conclusão já era esperada. O excessivo tempo de processamento provém quase todo do cálculo da aptidão. Uma simples execução do AG com 100 cromossomos em 200 gerações equivale a calcular a pseudoinversa $100 \times 200 = 20.000$ vezes. Consome cerca de 10^3 vezes mais tempo para treinar uma rede pelo treinamento híbrido (assumindo que metade do treinamento híbrido deve-se a fase supervisionada). Saber se é viável ou não utilizar um Algoritmo Genético em redes RBF dependerá do problema. Os Algoritmos Genéticos serão uma alternativa viável em problemas em que após treinadas as redes RBF não precisam mais ser modificadas e onde o tempo de treinamento não é um fator crucial. De qualquer modo, dada a natureza do treinamento das redes RBF, a sua otimização genética é bem mais rápida que a otimização genética de redes MLP.

6

CONCLUSÃO

“Recorda: toda criatura, neste mundo, tem um recado a dizer”
(Espírito Emmanuel)

Os Algoritmos Genéticos e as Redes Neurais RBF tem de larga aplicação prática em diversificadas áreas. Algoritmos Genéticos é uma poderosa ferramenta de otimização de propósito geral. redes RBF é aplicado em problemas de classificação, regressão e previsão de séries temporais. Contudo, tais técnicas são novas e pouco conhecidas. Conforme mostra Capítulos 2 a 4, o desenvolvimento desta dissertação possibilitou o domínio dessas duas técnicas que teve como meta a otimização de Redes Neural RBF via Algoritmos Genéticos.

Experimentos, descritos no capítulos 5, comprovaram que o Algoritmo Genético superou alguns recentes algoritmos de treinamento de redes RBF. Vale frisar que estes resultados já eram esperados, visto que os Algoritmos Genéticos utiliza uma estratégia de busca global que permite encontrar soluções ótimas (ou bem próximas), ao passo que os demais algoritmos são métodos de busca local que permitem achar somente soluções sub-ótimas.

Porém, constatou-se que a viabilidade da otimização genética de redes RBF está comprometida pelo longo tempo de processamento. Como um exemplo, durante experimentos o treinamento de uma rede RBF por métodos tradicionais gastou 2 minutos de processamento. Em condições semelhantes, o Algoritmo Genético consumiu 7 horas de processamento. Contudo, em muitas aplicações a rede RBF não é mais modificada após o treinamento. Nesta caso pode ser compensador gastar longo tempo de treinamento em troca

dos benefícios de uma rede altamente otimizada. Uma vez que os experimentos mostraram que as redes geradas tem tamanho reduzido (econômicas) e mantém o bom desempenho das redes maiores.

No que diz respeito ao Algoritmo Genético projetado nesta dissertação, conclui-se que o mesmo apresenta novas sugestões além de incorporar e aperfeiçoar as melhores idéias de trabalhos anteriores:

- **Técnicas de aglomeração:** usada para restringir a busca dos centros da rede RBF nas áreas altamente povoadas por padrões de treinamento.
- **Crossover:** o crossover não é simples troca de partes da estrutura do cromossomo como ocorre em Algoritmos Genéticos tradicionais. Ao invés, hipervolumes do espaço de entrada das redes RBF são trocadas. Carse e Fogarty (1996) utilizou idéia semelhante, porém diferente nos detalhes devido a representação genética adotada.
- **Otimização multiobjetiva:** a maioria dos Algoritmos Genéticos otimizam Redes Neurais usando uma função de aptidão. Neste trabalho foi utilizado duas funções de aptidão combinadas através do método pareto-ótimo. Por este método, o Algoritmo Genético produz um conjunto de redes RBF (o conjunto pareto-ótimo) na qual nenhuma Rede de tal conjunto é melhor as demais considerando as duas funções de aptidão ao mesmo tempo. Billings e Zheng (1995) utilizou-se da otimização multiobjetiva por apresentar melhor generalização da Rede.
- **Critério de seleção do modelo:** a aptidão foi calculada com base no critério GCV (Generalised Cross-Validation). Orr (1995) encontrou bons resultados com GCV, porém usou um método diferente do Algoritmo Genético.

Esta série de melhoramentos resultou em um algoritmo eficiente que foi capaz de obter Redes Neurais surpreendentemente pequenas mas com alto desempenho e capacidade de generalização. Contudo é possível melhorar a diversidade de topologias incentivando a formação de nichos na população do Algoritmo Genético (Goldberg, 1989). Diversos trabalhos tem utilizado nichos em problemas de otimização multiobjetiva usando pareto-ótimo (Horn et al, 1994).

Trabalhos futuros podem ser direcionados em dois sentidos. Primeiro, aplicando as técnicas aqui desenvolvidas e apresentadas em outros tipos de redes neurais. O segundo direcionamento vem do domínio de duas técnicas que tem largo campo de aplicação na qual possibilitara seu emprego em muitas áreas. Algoritmos Genéticos são simples e poderosos e tem apresentado resultados efetivos como desmonstra esta Dissertação.

Com isso, espera-se que a presente dissertação tenha atingido o seu objetivo maior que é contribuir para a desenvolvimento científico.

7

BIBLIOGRAFIA

*"Vale mais o teu braço amigo ao irmão caído no precipício
do sofrimento, que a mais ampla biblioteca do mundo em
cintilações verbalistas na tua boca"*
(Espírito Emmanuel)

"Age em favor de alguma criança sem proteção"
(Espírito Emmanuel)

- AKAIKE, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, v.19, p.716-723, 1974.
- ALMEIDA, F.; DUMONTIER, P. O uso de redes neurais em avaliação de riscos de inadimplência. *Revista de Administração*, v.31, n.1, p.52-63, Março de 1996.
- ALTMAN, E.I. Financial ratios, discriminant analysis and the prediction of corporation bankruptcy. *Journal of Finance*, v.23, n.4, p.589-609, 1968.
- ALTMAN, E.I.; BAIDYA, T. K. N.; DIAS, L. M. R. Previsão de problemas financeiros em empresas. *Revista de Administração de Empresas*, v.19, n.1, p.17-28, 1979.
- ANDERBERG, M. R. *Cluster Analysis for Applications*. Academic Press, New York, 1973.
- ANDERSON, J.A. An Introduction to Neural Networks, Cambridge, MA: The MIT Press, 1995.
- BALAKRISHNAN, K.; HONAVAR, V. Evolutionary design of neural architectures - a preliminary taxonomy and guide to literature. Artificial Intelligence Research Group. CS TR #95-01, 1995. (<http://www.cs.iastate.edu/~gannadm/homepage.html>)
- BEALE, R.; JACKSON, T. Neural Computing, an Introduction. Adam Hilger, IOP Publishing Ltd : Bristol. 1990.
- BEASLEY, D.; BULL, D.R.; MARTIN, R.R. An Overview of Genetic Algorithms: Part 1, Fundamentals. *University Computing*, v.15, n.2, p.58-69, 1993a. (Disponível por ftp no ENCORE no arquivo: GA/papers/over93.ps.gz)
- BEASLEY, D.; BULL, D.R.; MARTIN, R.R. An Overview of Genetic Algorithms: Part 2, Research Topics, *University Computing*, 15(4) 170-181, 1993b. (Disponível por ftp no ENCORE no arquivo: GA/papers/over93-2.ps.gz)
- BERTHOLD, M. R.; DIAMOND, J. Boosting the performance of rbf networks with dynamic decay adjustment. In: TESAURO, G.; TOURETZKY, D. S.; LEEN, T. K. (eds). *Advances in Neural Information Processing Systems*. volume 7, p.521-528, 1995.
- BOLDRINI, J. L.; COSTA, S. R.; FIGUEIREDO, V. L.; WETZLER, H. G. *Álgebra linear*. 3.ed, Harper &

- Row do Brasil, 1980.
- BILLINGS, S. A.; ZHENG, G. L. Radial basis function network configuration using genetic algorithms. *Neural Networks*, v.8, n.6, p.877-890, 1995.
- BISHOP, C. M. *Neural Networks for Pattern Recognition*. Oxford, Clarendon Press, 1995.
- BRANKE, J. *Evolutionary algorithms for neural network design and training*. Technical Report n.322, University of Karlsruhe, Institute AIFB, 1995.
- BREIMAN L.; FRIEDMAN, J. H.; OLSHEN, R. A.; STONE, C. J.. *Classification and regression trees*. Belmont, CA: Wadsworth, 1984.
- BROOMHEAD D. S.; LOWE D. Multivariable functional interpolation and adaptive networks. *Complex Systems*, v.2, p.321-355, 1988.
- CARSE, B; FOGARTY, T.C. Fast evolutionary learning of minimal radial basis function neural networks using a genetic algorithm. In: FOGARTY, T.C., ed. *AISB Workshop on Evolutionary Computing. Lectures Notes in Computer Science* n.1143, p.1-22, Springer-Verlag, 1996.
- CARTER. C.; CATLETT, J. Assessing credit card applications using machine learning. *IEEE Expert*, v.2, n.3, p.71-79, 1987.
- CHALMERS, D.J. The evolution of learning: An experiment in genetic connectionism. In: TOURETZKY, D.S., ELMAN J.L.; HINTON, G. E., eds. *Connectionist models: Proceedings of the 1990 summer school*, Pittsburgh, 1990, p.81-90, San Mateo, CA, Morgan Kaufmann, 1991.
([ftp://archive.cis.ohio-state.edu/pub/neuroprose/chalmers.evolution.ps.Z](http://archive.cis.ohio-state.edu/pub/neuroprose/chalmers.evolution.ps.Z)).
- CHEN, S.; COWAN, C. F. N; GRANT, P. M. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, v.2, n.2, p.302-309, 1991.
- CHEN, S.; WU, Y.; ALKADHIMI K. A two-Layer learning method for radial basis function networks using combined genetic and regularised OLS algorithms. In: *Proceedings of the 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, p.245-249, 1995.
- DARWIN, C. A origem das espécies e a seleção natural. Hemus editora.
- DAVIS, L. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.
- DEJONG, K. *The analysis and behaviour of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.
- DUDA, R. O.; HART, P. E. *Pattern Classification and Scene Analysis*. New York, John Wiley, 1973.
- ENCORE. The Evolutionary Computation REpository network. 1998. (<http://alife.santafe.edu/~joke/encore/>)
(<http://www.cs.purdue.edu/coast/archive/clife/Welcome.html>).
- FAHLMAN, S.E. An empirical study of learning speed in back-propagation networks. (Technical Report CMU-CS-88-162, 1988).
- FISHER, R. A. The use of multiple measurements in taxonomic problems. In: *Annals of Eugenics* v.7, 179-188, 1936. /Reimpresso in: *Contributions to Mathematical Statistics*. New York, John Wiley, 1950/
- GALLANT, S. *Neural networks and expert systems*. Cambridge, MA, MIT Press, 1993.
- GOLDBERG, D. E. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.
- GOLUB, G. H. ; HEATH, M. ; WAHBA, G. Generalised cross-validation as a method for choosing a good ridge parameter. *Technometrics*, v.21, n.2, p.215-223, 1979.
- GOLUB, G. H.; VAN LOAN, C. F. *Matrix Computations*. Johns Hopkins University Press, 1983.
- GRUAU, F. Genetic synthesis of boolean neural networks with a cell rewriting developmental process. In WHITLEY; SCHAFFER, eds. *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks*, p.55-74, 1992.
- GRUAU, F. *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Ecole Normale Supérieure de Lyon, 1994. (anonymous ftp: lib.ens-lyon.fr (140.77.1.11) directory pub/Rapports/PhD file PhD94-01-E.ps.Z).
- HANCOCK, P. J. B. Genetic algorithms and permutation problems: a comparison of recombination operators for neural net structure specification. In: *Proceedings of IEEE Workshop on Combinations of Genetic Algorithms and Neural Networks*. p.108-122, 1992.
- HARP, S.A.; SAMAD, T.; GUHA, A. Towards the genetic synthesis of neural networks. In: *Proceedings of the Third International Conference on Genetic Algorithms*, p.360-369, Morgan Kaufmann, 1989.
- HARP, S.A.; SAMAD, T. Genetic Synthesis of neural network architecture. In: DAVIS, L. *Handbook of Genetic Algorithms*. p.202-221, Van Nostrand Reinhold, 1991.

- HARTMAN, E.; KELLER, J. D. Predicting the future: advantages of semilocal units. *Neural Computation*. V.3, n.4, p.566-578, 1991.
- HASSOUN, M. H. *Fundamentals of artificial neural networks*. MIT Press, 1995.
- HAYKIN, S. *Neural Networks - A Comprehensive Foundation*. Macmillan College Publishing Company, 1994.
- HEITKOETTER J.; BEASLEY D., eds. The Hitch-Hiker's guide to evolutionary computation: a list of frequently asked questions, 1998. (Disponível por ftp no ENCORE).
- HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.
- HOPCROFT, J.E; ULLMAN J.D. *Introduction to automata theory, languages and computation*, Addison-Wesley, 1979.
- HORN, J.; NAFPLIOTIS, N.; GOLDBERG, D.E. A niched pareto genetic algorithm for multiobjective optimization. In: *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence (ICEC '94)*. Volume 1, Piscataway, NJ: IEEE Service Center, 82-87. 1994. (<http://gal4.ge.uiuc.edu/>)
- HORNIK K; STINCHCOMBE M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, v.2, n.5, p.359-366, 1989.
- HORST P.; PADILHA T., ROCHA, C.; REZENDE, S.; CARVALHO, A. Knowledge acquisition using symbolic and connectionist algorithms for credit evaluation. In: *Proceedings of the IEEE World Congress on Computational Intelligence, WCCI'98*, Anchorage, USA, May, 1998.
- JENSEN, H. L. Using neural networks for credit scoring. *Managerial Finance*. p.15-26, 1992.
- JONES, A. J. Genetic algorithms and their applications to the design of neural networks. *Neural Computing & Applications*, v.1, p.32-45, 1993.
- KADIRKAMANATHAN, V. ; NIRANJAN, M.. A function estimation approach to sequential learning with neural networks. *Neural Computation*, v.5, n.6, p.954-975, 1993.
- KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, v.220, p.671-680, 1983.
- KITANO, H. Designing neural networks using genetic algorithms with graph generation systems. *Complex Systems*, n.4, p.461-476, 1990a.
- KITANO, H. Empirical studies on the speed of convergence of neural network training using genetic algorithms. In: *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*. v.II, p.789-795, PROC AAAI-90, Cambridge, MA, MIT Press, 1990b.
- KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics* v.43, 59-69., 1982.
- KOHONEN, T. Self-organization and associative memory. 3rd edition. Springer Verlag, 1989.
- KOLODNER, J. *Case-Based Reasoning*, San Mateo, CA, Morgan Kaufmann, 1993.
- LACERDA, E.G.M.; CARVALHO, A.C.P.L.F. Combinando os Algoritmos Genético e K-média para Configurar Redes de Funções Base Radial. In: *Anais do IV Simpósio Brasileiro de Redes Neurais*, p.95-97, 1997.
- LACERDA, E.G.M.; CARVALHO, A.C.P.L.F. Redes de Funções Base Radial para Avaliação de Risco de Crédito. In: *Anais do V Simpósio Brasileiro de Redes Neurais*, 1998.
- LEITHOLD, L. *O Cálculo com geometria analítica*. Ed. Harper & Row, SP, 1977.
- LINDENMAYER, A. Mathematical models for cellular interactions in development. *Journal of Theoretical Biology*, parts I and II, v.18, p.280-299, 1968.
- LOWE, D. Adaptive radial basis function nonlinearities and the problem of generalisation, in: First IEEE International Conference on Artificial Neural Networks, Conference Publication 313, London, Institute of Electrical Engineers, p.171-175, 1989.
- LOWE, D. Radial basis function networks. In: Arbib, M. A. ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
- LUCASIUS, C. B.; KATEMAN, G. Towards solving subset selection problems with the aid of the genetic algorithm. In MANNER, R.; MANDERICK, B. eds. *Parallel Problem Solving from Nature* (vol.2) Amsterdam. Elsevier Science Publishers, 1992.
- MAILLARD, E. P.; GUEROT, D. RBF neural network, basis functions and genetic algorithm. *Proceedings of International Conference on Neural Networks*, vol 4, p.2187-2192. 1997.
- MACKEY, M. C.; GLASS L. Oscillations and Chaos in Physiological Control Systems. *Science* p.197:287, 1977.
- MANDISCHER, M. Representation and Evolution of Neural Networks. In: ALBRECHT, R.F.; REEVES,

- C.R.; STEELE N.C. (eds). *Artificial Neural Nets and Genetic Algorithm Proceedings of the International Conference at Innsbruck, Austria*, p. 643-649, Springer, Wien and New York, 1993. (Disponível por ftp no ENCORE no arquivo: GA/papers/icannga93.ps.gz).
- MENDES E. F.; CARVALHO A. C. P. L. F.; MATIAS; A. B. Utilização de redes neurais artificiais na análise de risco de crédito a pessoas físicas. In: *Anais do III Simpósio Brasileiro de Redes Neurais*, p.287-292, 1996.
- MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. 2.ed. Springer-Verlag, 1994.
- MILLER, G. F.; TODD, P. M.; HEDGE, S. U. Designing neural networks using genetic algorithms. In: *Proceedings of the Third International Conference on Genetic Algorithms*, p.379-384, Morgan Kaufmann, 1989.
- MITCHELL, M. *An Introduction to Genetic Algorithms*. MIT Press, 1996.
- MONTANA, D. J.; DAVIS L. Training feedforward neural networks using genetic algorithms. In: *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, p.762-767, Morgan Kaufmann, 1989.
- MOODY, J. Prediction risk and architecture selection for neural networks. In: Cherkassky, V. Friedman; J. H.; Wechsler, H., eds, *From statistics to neural networks: theory and pattern recognition applications*, NATO ASI series F, Springer-verlag, 1994. (<ftp://neural.cse.ogi.edu/pub/neural/papers/moody94.predictionrisk.ps.Z>).
- MOODY, J.; DARKEN, C. J. Fast learning in networks of locally-tuned processing units. *Neural Computation*, v.1, n.2, p.281-294, 1989.
- MURPHY, P. M.; AHA, D. W. *UCI Repository of machine learning databases*. Irvine, CA: University of California. Department of Information and Computer Science. 1994. (<http://www.ics.uci.edu/~mlearn/MLRepository.html>).
- MUSAVI, M. T.; AHMED, W.; CHAN, K. H.; FARIS, K. B.; HUMMELS, D. M. On the training of radial basis function classifiers. *Neural Networks*, v5, n.4, p.595-603, 1992.
- ORR, M. J. L. Regularisation in the selection of radial basis function centres. *Neural Computation*, v.7, n.3, p.606-623, 1995. (<http://www.cns.ed.ac.uk/people/mark>)
- ORR, M. J. L.. Introduction to radial basis function networks. TR Centre for Cognitive Science, University of Edinburgh, Scotland, 1996. (<http://www.cns.ed.ac.uk/people/mark>) .
- PARK, J.; SANDBERG, I.W. Universal approximation using radial basis function networks. *Neural Computation*, v.3, n.2, p.246-257, 1991.
- PLATT, J. A resource-allocating network for function interpolation. *Neural Computation*, v.3, n.2, p.213-225, 1991.
- PODDIG, T. Bankruptcy prediction: a comparison with discriminant analysis. Refenes, A. P. *Neural networks in capital markets*. p.311-340. New York, John Wiley & Sons, 1995.
- POGGIO, T.; GIROSI F. Networks for approximation and learning. *Proceedings of the IEEE*, v.78, n.9, p.1481-1497, 1990.
- POWELL, M.J.D. The theory of radial basis function approximation in 1990. In: LIGHT, W. ed. *Advances in Numerical Analysis*, v.3, p.105-210, Oxford: Clarendon, 1992.
- PRECHTEL, L. Proben1- a set of neural network benchmark problems and benchmarking rules. Tech. Rep. 21/94, Fakultät für Informatik, University Karlsruhe, (<ftp://ira.uka.de/pub/neuron/proben1.tar.gz>), 1994.
- PRESS, W.H.; FLANNERY, B.P.; TEUKOLSKY, S.A.; VETTERLING, W.T. *Numerical Recipes in C*. Cambridge University Press, 1988
- QUINLAN, J. R. Induction of decision trees. *Machine Learning*. v.1, n.1, p.81-106, 1986.
- QUINLAN, J. R. *C4.5 : Programs for Machine Learning*. Morgan Kaufman, 1993.
- RIPLEY, B. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996.
- ROBERTS, S. G.; TUREGA M. Evolving neural network structures: an evaluation of encoding techniques. In: PEARSON, D. W.; STEELE, N. C.; ALBRECHT, R. F., eds, *Artificial neural nets and genetic algorithms*, Springer-Verlag, 1995.
- ROBBINS, H.; MONRO, S. A. Stochastic Approximation Method. *Annals of Mathematical Statistics*, 22, 400-407, 1951.
- ROOIJ, A. J. F. VAN; JAIN, L. C.; JOHNSON, R. P. *Neural Network Training Using Genetic Algorithms* . Series in Machine Perception and Artificial Intelligence, Vol 26, World Scientific Pub Co, 1997.
- ROY, A.; GOVIL, S.; MIRANDA, R. An algorithm to generate radial basis function (RBF)-like nets for classification problems. *Neural Networks*, v.8, n.2, p.179-201, 1995.

- SAHA, A.; KELLER, J. D. Algorithms for better representation and faster learning in radial basis function networks. In: Touretzki, D. S. ed. *Advances in Neural Information Processing Systems*. Volume 2, pp.482-489, 1990.
- SARLE, W.S., ed. Neural Network FAQ, part 3 of 7: Introduction, periodic posting to the Usenet newsgroup comp.ai.neural-nets, 1997 (<ftp://ftp.sas.com/pub/neural/FAQ.html>)
- SCHAFFER, J.D; WHITLEY, D.; ESHELMAN L. J. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In: *Proceedings of the International Workshop on Combinations of genetic algorithms and neural networks*, p.1-37, Baltimore, IEEE, 1992.
- SHANNON, C.E. A mathematical theory of communication. *The Bell System Technical Journal*, v.27, n.3, p.379-423 and p.623-656, 1948.
- SCHIFFMANN W.; JOOST, M.; WERNER R. Synthesis and performance analysis of neural network architectures. Technical Report 16/1992, University of Koblenz, Germany, (<ftp://archive.cis.ohio-state.edu/pub/neuroprose/schiff.nnga.ps.Z>), 1992.
- SCHIFFMANN W.; JOOST, M.; WERNER R. Application of genetic algorithms to the construction of topologies for multilayer perceptrons. In: *Proceedings of the International Joint Conference on Neural Networks and Genetic Algorithms*, p.675-682, Innsbruck., 1993.
- SCHIFFMANN W.; JOOST, M.; WERNER R. Performance evaluation of evolutionary created neural network topologies. In: SCHWEFEL, H. P.; MAENNER R., eds, *Proceedings of Parallel Problem Solving from Nature*, p.274-283. *Lectures Notes in Computer Science*, Springer, 1991.
- SCHRICKEL, W.K. *Análise de crédito: concessão e gerência de empréstimos*. Atlas, 1994.
- SILVA, J.P. *Análise e decisão de crédito*. São Paulo, São Paulo, SP: Atlas, 1988.
- SILVA, J.P. *Gestão e análise de risco de crédito*. São Paulo, SP: Atlas, 1997.
- TRIPPI, R.; TURBAN, E; (eds). *Neural Networks in Finance and Investing*. Chicago: Probus Publishing Company, 1993.
- VOGT, M. Combination of radial basis function neural networks with optimized vector quantization. *Proc. IEEE Int. Conf. on Neural Networks*, vol 3, p.1841-1846, 1993.
- WETTSCHERECK, D.; DIETTERICH, T. Improving the performance of radial basis function networks by learning center localizations. In: MOODY, J. E.; HANSON, S. J.; LIPPMANN, R. P. (eds.). *Advances in Neural Information Processing Systems*. volume 4, pp.847-854. San Mateo, CA: Morgan-Kaufmann., 1992.
- WHITEHEAD, B.A; CHOATE, T.D. Evolving space-filling curves to distribute radial basis functions over an input space. *IEEE Transactions on Neural Networks*, v.5, n.1, p.15-23, 1994.
- WHITEHEAD, B.A.; CHOATE, T.D. Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Transactions on Neural Networks*, v.7, n.4, p.869-880, 1996.
- WHITLEY, D. L. A genetic algorithm tutorial. Technical Report CS-93-103, Colorado State University, 1993. (Disponível por ftp no ENCORE no arquivo: GA/papers/tutor93.ps).
- WHITLEY, D.; DOMINIC, S.; DAS R.; ANDERSON, C.W. Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, v.13, p.259-289, 1993.
- WHITLEY, D.; STARKWEATHER, T.; BOGART, C. Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, v.14, n.3, p.347-361, 1990.
- WIDROW, B.; HOFF, M. E. Adaptive switching circuits. In: *IRE-WESCON Convention Record*, v.4, p.96-104, New York, 1960. /Reimpresso in: ANDERSON, J. A; ROSENFELD E., eds. *Neurocomputing: Foundations of Research*. Cambridge, MA: MIT Press, 1988/
- WIDROW, B.; LEHR, M. A. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*. v.78, n.9, p.1415-1442, 1990.
- WIDROW, B.; RUMELHART D. E.; LEHR, M. A. Neural networks: Applications in industry, business and science. *Communications of the ACM*, v.37, n.3, 1994.
- WILSON, R.L.; SHARDA, R. Bankruptcy prediction using neural networks. *Decision Support Systems*, v.11, n.5, p.545-557, 1994.
- YINGWEI, L., SUNDARARAJAN, N.; SARATCHANDRAN, P. A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, v.9, p.461-478, 1997.
- ZELL, A. et al. SNNS: Stuttgart Neural Network Simulator user manual version 4.1. Report n.6/95., 1995. (<http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/>).

A

REDES NEURAIS PERCEPTRON DE MULTICAMADAS

A.1 INTRODUÇÃO

A Rede Neural MLP (*Multilayer-Perceptron*), Figura A.1, é talvez a Rede Neural mais conhecida e utilizada, e tem sido aplicada com sucesso em problemas de classificação, sistemas de controle, regressão, previsão de séries temporais, processamento de sinais, entre outros.

A Rede MLP quando treinada, isto é, com seus parâmetros apropriadamente determinados, pode realizar o mapeamento do **conjunto de treinamento** que é constituído por P padrões que são pares entrada-saída :

$$[\mathbf{x}^{(1)}, \mathbf{t}^{(1)}], [\mathbf{x}^{(2)}, \mathbf{t}^{(2)}], \dots, [\mathbf{x}^{(P)}, \mathbf{t}^{(P)}]. \quad (\text{A.1.1})$$

Onde, $[\mathbf{x}^{(i)}, \mathbf{t}^{(i)}]$ é conhecido por vários nomes como padrão, exemplo ou observação.

$\mathbf{x}^{(i)}$ é o **vetor de entrada** do padrão i .

$\mathbf{t}^{(i)}$ é o **vetor saída desejada** do padrão i .

Para o treinamento de Redes MLP, existe um eficiente algoritmo denominado *backpropagation*. Este algoritmo foi descoberto e popularizado por Rumelhart e McClelland, (1986) e também descoberto por vários pesquisadores independentemente (Werbos, 1974; Parker, 1985).

Vale frisar que a Rede MLP treinada com o algoritmo *Backpropagation* pode realizar complexos e complicados mapeamentos e é capaz de aproximar qualquer função com precisão arbitrária (Haykin, 1994). Uma vez treinada, a Rede Neural realiza mapeamento dos padrões como descrito a seguir.

A.2 FORWARD PROPAGATION

Considere a Rede MLP mostrada na Figura A.1 na qual todas as unidades de uma camada estão conectadas com as unidades da camada seguinte. A Rede Neural recebe em cada unidade da camada de entrada um componente do vetor de entrada $\mathbf{x}$

Em cada unidade da Rede Neural MLP, com exceção das unidades da camada de entrada, é computada a seguinte soma ponderada:

$$\text{net}_j = \sum_{i=1}^m a_i w_{ji} + \beta_j \quad (\text{A.2.1})$$

Onde, w_{ji} é o peso sobre a conexão que sai da unidade i e chega na unidade j .

a_i é ou uma entrada x_i ou a ativação da unidade i .

β é uma constante denominada **bias**.

Por simplicidade notacional, é adicionado uma unidade extra em cada camada (com exceção da camada de entrada) com peso $w_{j0} = \beta_j$ e ativação sempre igual a um, isto é, $a_0 = 1$. Deste modo, o *bias* β_j não precisa ser escrito explicitamente. Então (A.2.1) é rescrita como:

$$\text{net}_j = \sum_{i=0}^m a_i w_{ji} \quad (\text{A.2.2})$$

A soma computada em (A.2.2) é transformada pela equação:

$$a_j = f(\text{net}_j) \quad (\text{A.2.3})$$

Onde, $f(x)$ é a **função de ativação** da unidade. Em geral, função de ativação utilizada é a função sigmóide logística (A.2.4), plotada na Figura A.2.

$$f(x) = \frac{1}{1 + \exp(-x)} \quad (\text{A.2.4})$$

O uso sucessivo de (A.2.2) e (A.2.3) começando pelas unidades da camada de entrada e seguindo até a camada de saída resulta em um processo denominado **forward propagation** (propagação para frente), Figura A.3. Deste modo o vetor de entrada $\mathbf{x}$ é propagado através da rede até a camada de saída gerando o vetor de saída $\mathbf{y}$. O vetor $\mathbf{y}$ é denominado **vetor de saída real** da rede.

Para que o mapeamento do $\mathbf{x}$ para o vetor $\mathbf{y}$ seja satisfatório é necessário que os pesos estejam ajustados apropriadamente. Tal ajuste é feito através do algoritmo **backpropagation**.

Figura A.1. Uma Rede Neural Perceptron de Multicamadas.

Figura A.2. Função sigmóide logística.

Figura A.3. Propagação para frente (*forward propagation*)

A.3 GRADIENTE DESCENDENTE

Treinar a Rede Neural usando o *backpropagation* significa ajustar o vetor peso $\mathbf{w}$ (cujos componentes são todos os pesos da rede) de modo a minimizar a soma dos erros quadráticos da rede que é dado por:

$$E = \frac{1}{2} \sum_{p=1}^P E^{(p)} \quad (\text{A.3.1})$$

$$E^{(p)} = \sum_{k=1}^c (t_k - y_k)^2 \quad (\text{A.3.2})$$

Onde, c é o número de unidades de saída da rede. O fator $\frac{1}{2}$ em (A.3.1) é conveniente para a obtenção das derivadas visto mais tarde. O *backpropagation* minimiza (A.3.1) usando o método do gradiente descendente.

Neste método, inicia-se os pesos em um ponto arbitrário $\mathbf{w}(0)$, e em seguida obtém-se uma sequência de pontos $\mathbf{w}(n)$, $n = 1, 2, 3, \dots$ até atingir o vetor peso ótimo $\mathbf{w}^*$ que minimiza (A.3.1). Cada componente de $\mathbf{w}(n+1)$ é obtido de $\mathbf{w}(n)$ como segue:

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \quad (\text{A.3.3})$$

$$\Delta w_{ji}(n) = -\eta \frac{\partial E}{\partial w_{ji}(n)} \quad (\text{A.3.4})$$

Onde, η é uma constante de pequeno valor denominada **taxa de aprendizado**.

A finalidade deste procedimento é guiar o movimento do ponto $\mathbf{w}(n)$ na direção de maior decréscimo da função E , que é dado por $-\nabla_{\mathbf{w}}E$ (A.3.5), até atingir (como esperado) o ponto ótimo $\mathbf{w}^*$. O vetor $\nabla_{\mathbf{w}}E$ é denominado **vetor gradiente**.

$$\nabla_{\mathbf{w}}E = \left(\frac{\partial E}{\partial w_{1,0}}, \frac{\partial E}{\partial w_{1,1}}, \dots, \frac{\partial E}{\partial w_{J,1}}, \dots \right) \quad (\text{A.3.5})$$

É ilustrado na Figura A.4, uma superfície hipotética da função erro E em função de $\mathbf{w}$. Começando em um ponto arbitrário $\mathbf{w}(0)$ que é deslocado, passo a passo, em direção oposta ao vetor gradiente, isto é, $-\nabla_{\mathbf{w}}E$, alcança-se o ponto ótimo $\mathbf{w}^*$.

Na Figura A.4 existem dois vales. O primeiro vale contém o ponto A (mínimo global) e representa a solução ótima. O segundo vale contém o ponto B (mínimo local) e representa uma solução sub-ótima $\mathbf{w}^*$. Se $\mathbf{w}(0)$ for iniciado próximo do segundo vale, o vetor $-\nabla_{\mathbf{w}}E$ que indica a direção de maior decréscimo da superfície, levará o algoritmo a cair erroneamente dentro do segundo vale (solução sub-ótima). Este problema pode ser superado reinicializando $\mathbf{w}(0)$ e rodando o algoritmo novamente. Mas vale frisar que em muitos problemas práticos, uma solução sub-ótima pode ser satisfatória.

A seguir as derivadas de (A.3.4) necessárias ao método do gradiente serão calculadas.

Figura A.4. Superfície de erro da Rede Neural.

A.4 CÁLCULO DAS DERIVADAS

Nesta seção será obtida o valor da derivada encontrada em (A.3.4) mostrada a seguir.

$$\frac{\partial E}{\partial w_{ji}} \quad (A.4.1)$$

Para simplificar o cálculo desta derivada, a função E (A.3.1) que refere-se a todos os padrões, será substituída pela função $E^{(p)}$ (A.3.2) que refere-se a um único padrão, na qual será denotada nesta seção apenas por E . Assim:

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial w_{ji}} \quad (A.4.2)$$

Introduzindo a seguinte notação:

$$\delta_j = -\frac{\partial E}{\partial \text{net}_j} \quad (\text{A.4.3})$$

E derivando,

$$\frac{\partial \text{net}_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_j a_j w_{ji} = a_j \quad (\text{A.4.4})$$

Obtém-se,

$$\frac{\partial E}{\partial w_{ji}} = -\delta_j a_j \quad (\text{A.4.5})$$

A seguir será obtido o valor de δ_j para as unidades de saída e intermediárias.

I) O cálculo de δ_j para as unidades de saída

Para as unidades da camada de saída, o cálculo de δ_j é direto:

$$\delta_j = -\frac{\partial E}{\partial \text{net}_j} = -\frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial \text{net}_j} \quad (\text{A.4.6})$$

$$\frac{\partial E}{\partial y_j} = \frac{\partial}{\partial y_j} \left[\frac{1}{2} \sum_{j=1}^c (t_j - y_j)^2 \right] = -(t_j - y_j) \quad (\text{A.4.7})$$

$$\frac{\partial y_j}{\partial \text{net}_j} = f'(\text{net}_j) \quad (\text{A.4.8})$$

A derivada da função de ativação sigmóide (A.2.4) é:

$$f'(x) = f(x)(1 - f(x)) \quad (\text{A.4.9})$$

Vale observar que é a derivada da função sigmóide é simples e requer apenas duas operações aritméticas, que a torna adequada para programas de computador. Substituindo (A.4.8) em (A.4.9) é obtido:

$$\frac{\partial y_j}{\partial \text{net}_j} = y_j(1 - y_j) \quad (\text{A.4.10})$$

Substituindo (A.4.7) e (A.4.10) em (A.4.6) obtém-se o valor de δ_j para as unidades de saída:

$$\delta_j = (t_j - y_j) y_j (1 - y_j) \quad (\text{A.4.11})$$

II) O cálculo de δ_j para as unidades intermediárias

Para as unidades das camadas intermediárias o valor de δ_j é:

$$\delta_j = -\frac{\partial E}{\partial \text{net}_j} = -\sum_k \frac{\partial E}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial \text{net}_j} \quad (\text{A.4.12})$$

Onde as unidades com índice k referem-se as unidades que recebem uma conexão fornecida pela unidade j . Aplicando a notação definida em (A.4.3) na (A.4.12) obtém-se:

$$\delta_j = \sum_k \delta_k \frac{\partial \text{net}_k}{\partial \text{net}_j} \quad (\text{A.4.13})$$

Desenvolvendo (A.4.13) obtém-se.

$$\frac{\partial \text{net}_k}{\partial \text{net}_j} = \frac{\partial \text{net}_k}{\partial a_j} \frac{\partial a_j}{\partial \text{net}_j} \quad (\text{A.4.14})$$

$$\frac{\partial \text{net}_k}{\partial a_j} = \frac{\partial}{\partial a_j} \sum_j a_j w_{kj} = w_{kj} \quad (\text{A.4.15})$$

$$\frac{\partial a_j}{\partial \text{net}_j} = f'(\text{net}_j) = a_j(1 - a_j) \quad (\text{A.4.16})$$

Substituindo (A.4.15) e (A.4.16) em (A.4.13) o valor de δ_j para as unidades intermediárias é dado por:

$$\delta_j = a_j(1 - a_j) \sum_k \delta_k w_{kj} \quad (\text{A.4.17})$$

Vale notar que o valor de δ_j depende dos valores de δ_k de todas as unidades k que recebe uma conexão fornecida pela unidade j . Portanto, o cálculo de δ_j pode ser interpretado como uma retropropagação (do inglês, **backpropagation**) dos δ_k em direção a unidade j . Ver Figura A.5.

III) Resumo

Resumindo os resultados desta seção, obtém-se a regra de ajuste de pesos, aplicando (A.4.5) em (A.3.4).

$$\Delta w_{ji}(n) = \eta \delta_j(n) a_j(n) \quad (\text{A.4.18})$$

Onde δ_j é obtido por (A.4.11) quando a unidade j pertence a camada de saída e obtido por (A.4.17) quando a unidade j pertence as camadas intermediárias.

A seguir será discutido a convergência do algoritmo *backpropagation*.

Figura A.5. Retropropagação (*backpropagation*)

A.5 MOMENTO

O algoritmos baseados no gradiente descendente pode convergir da forma ilustrada na Figura A.6. Contudo, se o valor da taxa de aprendizado η em (A.3.4) for muito pequeno, o algoritmo convergirá lentamente, pois os ajustes dos pesos Δw_{ji} serão pequenos. Se o valor de η for aumentado, grandes ajustes de pesos ocorrerão e o algoritmo apresentará um comportamento oscilatório divergindo para o infinito conforme mostra Figura A.6

Este problema pode ser aliviado com a adição de um termo denominado **momento** em (A.4.18), ou seja:

$$\Delta w_{ji}(n) = \eta \delta_j(n) a_j(n) + \alpha \Delta w_{ji}(n-1) \quad (\text{A.5.1})$$

Onde, α é uma constante que especifica a influência do momento. A adição do termo momento acelera a convergência sem a provocar divergência oscilatória, conforme mostrado a seguir. Desenvolvendo (A.5.1) obtém-se:

$$\Delta w_{ji}(n) = \eta \delta_j(n) a_j(n) + \alpha \eta \delta_j(n-1) a_j(n-1) + \alpha^2 \Delta w_{ji}(n-2)$$

$$\Delta w_{ji}(n) = \eta \delta_j(n) a_j(n) + \alpha \eta \delta_j(n-1) a_j(n-1) + \alpha^2 \eta \delta_j(n-2) a_j(n-2) + \alpha^3 \Delta w_{ji}(n-3)$$

...

$$\Delta w_{ji}(n) = \eta \delta_j(n) a_j(n) + \eta \sum_{t=0}^{n-1} \alpha^{n-t} \delta_j(t) a_j(t) \quad (\text{A.5.2})$$

Analisando (A.5.2), observa-se que quando ocorre a situação da Figura A.6 os termos do somatório, $\delta_j(t)a_j(t)$, possuem o mesmo sinal, então o valor do ajuste de pesos tenderá a crescer. Neste caso, o termo momento age acelerando a convergência.

Por outro lado, não irá provocar divergência oscilatória, pois quando os termos do somatório passarem a ter sinais contrários, devido a oscilação, eles tenderão a se cancelar, reduzindo o ajuste de pesos. Aqui, o termo momento age como elemento estabilizador.

O valor da constante α deverá ser $0 \leq \alpha < 1$ para que ocorra convergência (Haykin, 1994). A técnica dos momentos é simples e melhora significativamente a convergência.

A seguir será apresentado a versão final do algoritmo *backpropagation*.

Figura A.6. Convergência do método do gradiente.

Figura A.7. Divergência oscilatória no método do gradiente.

A.6 BACKPROPAGATION

O treinamento do *backpropagation* é considerado um **aprendizado supervisionado** (i. e. com professor). Quando o treinamento é realizado por padrões constituídos apenas pela parte de entrada sem a respectiva saída, o aprendizado é denominado **não-supervisionado** (i.e. sem professor). Um exemplo deste tipo de treinamento é apresentado no capítulo 2. O algoritmo *backpropagation* é sintetizado a seguir:

Para cada padrão apresentado ao algoritmo, os pesos são ajustados de acordo com:

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \quad (\text{A.6.1})$$

$$\Delta w_{ji}(n) = \eta \delta_j(n) a_j(n) + \alpha \Delta w_{ji}(n-1) \quad (\text{A.6.2})$$

Onde,

$$\delta_j = \begin{cases} (t_j - y_j) y_j (1 - y_j) & \text{se } j \text{ for uma unidade de saída} \\ a_j (1 - a_j) \sum_k \delta_k w_{kj} & \text{se } j \text{ for uma unidade intermediária} \end{cases}$$

Quando todos os padrões forem apresentados, completa-se um **ciclo** ou **época** (do inglês, *epoch*). Então recomeça-se um novo ciclo. É mostrado na Figura A.8, uma sugestão de um algoritmo *backpropagation* visando apenas mostrar o funcionamento básico.

Em muitos problemas práticos, a taxa de aprendizado η é na ordem de 0.05 a 0.25 e o valor da constante α é $0 \leq \alpha < 1$. É frequente treinamentos onde o número de ciclos aproxima-se de 1000 ou mais. Há vários critérios de parada do treinamento. Alguns bem simples são:

- Parar depois de um número máximo de ciclos
- Parar quando o MSE atingir um valor mínimo aceitável (durante o treinamento o erro MSE decresce com o número de ciclos n).

Um melhor critério de parada é a técnica **early stopping**, visto mais tarde, que tem por objetivo melhorar a generalização da rede (i. e. o desempenho sobre novos padrões não utilizados no treinamento).

Em geral, para determinar a topologia da rede (i.e. o número de unidades e camadas da rede), o projetista da rede treina várias topologias (escolhidas pela sua experiência) e a melhor topologia é selecionada usando técnicas como *hold out* e *crossvalidation* (ver seção 2.5).

O algoritmo *backpropagation* tem sido constantemente aperfeiçoado. Alguns exemplos bem conhecidos são: o *quickpropagation* (Fahlman, 1988) e o RProp (Riedmiller e Braun, 1993). Na próxima seção, será apresentada a técnica **early stopping**.

Algoritmo A.1. *Backpropagation*

DADO:

- 1- Uma rede neural de multicamadas com função de ativação sigmóide. Onde, cada camada l possui $m^{(l)}$ unidades. A camada de entrada é $l = 0$ e a camada de saída é $l = L$.
- 2- Um conjunto de P pares entrada-saída (padrões): $\{[x^i, t^i], i=1...p\}$.
- 3- Um critério de parada: Exemplos: fixar um número máximo de ciclos, ou fixar um erro mínimo aceitável, utilizar *early-stopping*, etc...

Inicializar todos os pesos com pequenos valores aleatórios, por exemplo: $w_{ji}^{(l)} = \text{aleatório}(-0.1, 0.1)$

repita

```

para  $i = 1$  até  $L$  faça  $a_o^{(i)} = 1$  /* Ativação das unidades bias */
para  $n = 1$  até  $P$  faça
/*-----Propagação para frente-----*/
para  $j = 1$  até  $m^{(0)}$  faça  $a_j^{(0)} = x_j$  /* Ativação das unidades da camada de entrada */
para  $l = 1$  até  $L$  faça
para  $j = 1$  até  $m^{(l)}$  faça

$$\text{net} = \sum_{i=0}^{m^{(l-1)}} a_i^{(l-1)} w_{ji}^{(l)}, \quad a_j^{(l)} = \frac{1}{1 + \exp(-\text{net})}$$

para  $j = 1$  até  $m^{(L)}$  faça  $y_j = a_j^{(L)}$  /* Saída da rede */
/*-----Retropropagação-----*/
para  $j = 1$  até  $m^{(L)}$  faça

$$\delta_j^{(L)} = (t_j - y_j) y_j (1 - y_j)$$

para  $l = L-1$  até 1 faça
para  $j = 1$  até  $m^{(l)}$  faça

$$\delta_j^{(l)} = a_j^{(l)} (1 - a_j^{(l)}) \sum_{k=1}^{m^{(l+1)}} \delta_k^{(l+1)} w_{kj}^{(l+1)}$$

/*-----Atualização dos pesos-----*/
para  $l = 1$  até  $L$  faça
para  $j = 1$  até  $m^{(l)}$  faça
para  $i = 0$  até  $m^{(l-1)}$  faça

$$\Delta w_{ji}^{(l)}[n] = \eta \delta_j^{(l)} a_i^{(l-1)}, \quad w_{ji}^{(l)} = w_{ji}^{(l)} + \Delta w_{ji}^{(l)}[n] + \alpha \Delta w_{ji}^{(l)}[n-1]$$


```

fim para
até critério de parada

A.7 EARLY STOPPING

Esta técnica, que visa melhorar a capacidade de generalização da rede, utiliza dois conjuntos de padrões durante o treinamento:

- **Conjunto de treinamento:** utilizado para determinar os pesos da Rede Neural.
- **Conjunto de validação:** não participa da determinação dos pesos. Serve como um pseudo teste para a Rede Neural. Sua finalidade é medir a capacidade de generalização durante o treinamento da rede.

O comportamento típico do MSE sobre os dois conjuntos mencionados anteriormente em função do número de ciclos n é mostrado na Figura A.8. O MSE (do inglês, *Mean Squared Error*) serve para medir o desempenho da rede e é dado por:

$$\text{MSE} = \frac{1}{P} \sum_{i=1}^P \sum_{j=1}^c \left(t_j^{(i)} - y_j^{(i)} \right)^2 \quad (\text{A.7.1})$$

O MSE sobre o conjunto de treinamento tem a tendência de diminuir até atingir o valores muito pequenos. Neste caso, a rede apresenta excelente desempenho sobre o conjunto de treinamento. Contudo, muitas vezes, a rede também apresenta pobre generalização, ou seja, fraco desempenho sobre padrões que não constam no conjunto de treinamento. O comportamento da curva MSE sobre o conjunto de validação é diminuir e depois aumentar ao longo do treinamento. O aumento do MSE de validação significa que a rede está perdendo em generalização. A idéia do método *early stopping* é terminar o treinamento no ciclo onde o MSE de validação é mínimo.

Após o treinamento, a rede pode estar por demais “ajustada” ao conjunto de validação. Então, é necessário um terceiro conjunto de padrões (**conjunto de teste**) para confirmar o desempenho.

Aqui termina a apresentação das Redes MLP. Estas redes são bem descritas em praticamente todos os livros texto sobre Redes Neurais. Na literatura brasileira encontra-se os livros (Carvalho et al, 1998; Kovács, 1996). Alguns poucos exemplos de livros na literatura internacional são (Beale e Jackson, 1990; Bishop, 1995; Ripley, 1996). Em (Sarle, 1997) há uma bibliografia de livros sobre o assunto.

Figura A.8. Erros durante o treinamento da Rede Neural

B

AVALIAÇÃO DE RISCO DE CRÉDITO

*“Nos negócios, reinveste uma porção de tudo que ganhares,
guarda uma porção para teu uso e outra para os necessitados”*
(Buda)

*“Mobilize o capital do sorriso e observará que semelhante investimento
lhe trará precioso rendimento de colaboração e felicidade”*
(Espírito André Luiz)

Este apêndice descreve uma aplicação prática da área de finanças para modelo neural genético proposto nesta dissertação: a Avaliação de Risco de Crédito, ARC. A ARC é um problema que envolve uma avaliação da lucratividade e da garantia de uma aplicação de crédito. São exemplos de aplicação de crédito: empréstimo bancário, financiamento de um bem por um banco, venda a prazo em lojas comerciais (crediário), concessão de cartão de crédito, etc...

Na próxima seção será mostrado uma introdução a ARC. Na seção B.2 um estudo mais detalhado baseado em (Silva, 1988 e 1997) será apresentado, onde são enfatizados os cinco C's do crédito. Na seção B.3 serão abordados os principais métodos utilizados em ARC. Na seção B.4 uma comparação entre tais métodos será exposta.

B.1 INTRODUÇÃO

O crédito é a entrega de um valor mediante promessa de que este valor será restituído posteriormente. Existe, porém, o risco da promessa de pagamento da dívida não ser cumprida. Contratos formais (ex. notas promissórias) são geralmente realizados para garantir àquele que empresta o direito de recebimento da dívida. Contudo, isto não assegura que a dívida será paga, pois, o devedor pode não dispôr dos recursos necessários para o seu pagamento.

Surge, portanto, a necessidade de realizar uma análise técnica antes que o crédito seja concedido ao cliente. Esta análise é denominada ARC. O Risco de Crédito caracteriza os diversos fatores que poderão contribuir para que aquele que concedeu o crédito não receba do devedor o pagamento requerido (Silva, 1988).

A ARC é tradicionalmente realizada pelo analista de crédito (ex. gerente de banco) baseado na sua experiência anterior e na política de crédito adotada pela instituição de crédito a qual pertence. Os analistas desenvolvem suas próprias heurísticas para suportar suas decisões e consideram os cinco C's do crédito (**Caráter, Capacidade, Condições, Capital e Colateral**) importantes fatores nas suas decisões. Os cinco C's do crédito serão apresentado na próxima seção. Antes, porém, é interessante ver exemplos de concessão de crédito:

A) Exemplo de concessão de crédito a pessoas físicas

Algumas lojas comerciais solicitam os dados pessoais do cliente, salário, referências, etc... e consultam o SPC (Sistema de Proteção ao Crédito). Caso o SPC não retorne informação negativa sobre o cliente, a venda a prazo é autorizada. Em setores onde o valor de crédito é alto, tais como financiamento de habitação, uma grande quantidade de informações minuciosas é solicitada do cliente.

B) Exemplo de concessão de crédito a pessoas jurídicas:

Algumas grandes indústrias dispõem de departamentos especializadas em crédito. Estes departamentos desenvolvem várias atividades antes de conceder o crédito, entre as quais:

- Elaborar pastas cadastrais com dados de identificação dos clientes
- Elaborar análises financeiras sobre os clientes
- Coletar informações sobre protestos e atrasos de pagamento dos clientes
- Visitar os clientes para se formar um conceito mais sólido da quantidade de crédito que o cliente pode assumir

- Consultar outros fornecedores do cliente.

A conclusão é que, quando maior o valor do crédito concedido, tanto mais rigorosa é a Avaliação do Risco de Crédito. A seguir serão apresentados os fatores que os analistas de crédito julgam importantes na Avaliação do Risco de Crédito.

B.2 OS C'S DO CRÉDITO

Os C's de crédito (Weston e Brigham, 1972) representam uma forma tradicional para classificar os riscos envolvidos na concessão de crédito. Obviamente, na prática existem interações entre os riscos classificados pelos C's de crédito.

B.2.1 CARÁTER

Esta relacionado com a intenção de pagamento do cliente. A honestidade do cliente pode não ser fácil de ser identificada. A investigação do C de Caráter é feita através de pastas cadastrais que contêm informações, como identificação e qualificação dos clientes (ex. nome, endereço, profissão) e informações de relações comerciais com outros credores com respeito a pontualidade, títulos protestados etc...

Além disso, é comum empresas se organizarem através de convênios com o objetivo de trocarem informações sobre atrasos de clientes. Existem também os SPC's que são grandes fontes de informação sobre a falta de pontualidade de pessoas físicas. A pontualidade no pagamento das contas ajuda muito no conceito de Caráter do cliente. Entretanto, um atraso pode não ser uma questão de caráter, mas de falta de recursos. Outros fatores como cultura, hábitos, modo de vida, profissionalismo, honestidade nos negócios etc... são também importantes para o conceito de Caráter do cliente.

B.2.2 CAPACIDADE

É medido pelos fatores que contribuem na competência e na competitividade de uma empresa. O C de Capacidade pode ser investigada através do nível técnico e administrativo do pessoal da empresa, pela sua administração, pelo seu potencial comercial e seu potencial de produção.

Em (Silva, 1997), é observado que a Capacidade é diferente do sentido de capacidade de pagamento. A empresa com bom conceito de Capacidade terá maior

facilidade de chegar ao ponto de ter condições de pagamento. O C de Capacidade não é propriamente a capacidade de pagamento, ele apenas facilita alcançar tal situação.

Informações da Capacidade podem ser obtidas verificando aspectos da empresa como por exemplo: o grau de modernização da produção e tecnologias adotadas, os métodos de trabalho, as suas estratégias de marketing, as pessoas-chaves da empresa, o mercado coberto pelos produtos da empresa, etc.. Existem vários aspectos importantes relacionados com a estrutura organizacional da empresa que informam sobre sua Capacidade.

O conhecimento das decisões estratégicas da empresa também fornece valiosas informações sobre sua Capacidade. Por exemplo, através da decisão de crescimento da empresa pela aquisição de novas máquinas é importante que se conheça qual a fonte dos recursos do financiamento da compra e se gerará dinheiro para pagar tal financiamento.

A investigação da Capacidade de pessoas físicas pode ser feita através do currículo profissional, analisando os cargos que ocupou e os resultados atingidos, se possui estabilidade de emprego, etc..

B.2.3 CAPITAL

Refere-se a situação econômico-financeira da empresa. Não é medido apenas pelo capital social da empresa, isto é, o montante de capital subscrito pelos proprietários (acionistas, quotistas) da empresa cuja finalidade é gerar lucros. O conceito de Capital tem um sentido mais abrangente, relacionando-se com toda a estrutura econômico-financeira da empresa. Os lucros não provêm somente do capital social, mas de toda uma estratégia econômico-financeira (Schrickel, 1994).

O Capital pode ser medido através da análise dos índices financeiros encontrados nos demonstrativos da contabilidade. Alguns métodos apresentados mais adiante, como Análise Discriminante ou Redes Neurais, usam os índices financeiros para cálculo dos pesos das funções discriminantes ou das unidades da rede. Portanto, o que estes métodos fornecem é uma medida do C de Capital (Silva, 1987).

B.2.4 CONDIÇÃO

São os fatores que compõe o ambiente externo da empresa e que estão, a princípio, fora de seu controle. Por exemplo, a economia pode ser alterada em face de "pacotes econômicos" do governo. Como resultado desses pacotes, podem ser afetados o padrão monetário, as taxas de juros, os preços, os incentivos fiscais e as reservas de mercado. Tudo isto são fatores externos à empresa que podem afetar o seu desempenho.

Outros fatores externos que influenciam a condição de uma empresa são: o ramo de atividade que a empresa atua (ex. metalurgia, têxtil, diversões), região geográfica, forças da natureza (ex. prejuízos devido a secas ou enchentes), concorrentes que lançam produtos mais competitivos, efeitos da moda (ex. prejuízos devido a roupas que saem de moda), etc.. Enfim, todo um ambiente externo exercendo boa ou má influência sobre a saúde da empresa e que portanto deve se considerado no momento da concessão do crédito.

B.2.5 COLATERAL

O C de Colateral⁹ significa garantia. Serve para contrabalançar os riscos de não pagamento da dívida. Em vários casos, a concessão de crédito dependerá da garantia. São exemplos de garantias: Penhor, Hipoteca, Caução, Aval e Fiança.

B.3 MÉTODOS PARA AVALIAÇÃO DE RISCO DE CRÉDITO

Existem basicamente dois grupos de métodos para ARC:

- Métodos baseados na experiência anterior (métodos subjetivos);
- Métodos quantitativos.

A) Experiência Anterior

A utilização de experiência anterior do analista de crédito é o método mais antigo e ainda o mais utilizado na ARC. Constitui um valioso instrumento, principalmente quando as questões envolvidas são subjetivas. Por exemplo, o potencial administrativo da empresa, o grau de modernidade da produção, a honestidade do cliente, a análise do atual momento econômico do país etc.. Contudo, a experiência anterior não é fácil de ser adquirida, e só com o tempo e vivência em situações relevantes se pode obtê-la.

B) Métodos Quantitativos

Estes métodos, entre eles o *Credit Scoring*, servem para classificar quantitativamente se é possível ou não a operação com uma empresa que solicita crédito. Este problema é um caso particular de um problema mais geral, que na literatura de Rede Neurais é denominado **Classificação de Padrões**.

⁹ O que parece é que a palavra colateral é empregada aqui apenas para concordar com o inglês *collateral*. No Brasil, a palavra colateral significa algo que está ao lado, em paralelo e não é utilizada para caracterizar garantia em operações de crédito. (Silva, 1988).

Existe grande quantidade de métodos para tratar com o problema de Classificação de Padrões e a princípio todos podem ser utilizados na ARC. São métodos vindo de várias áreas: Estatística, Redes Neurais Artificiais, Aprendizado de Máquina, Sistemas Especialistas etc... Atualmente, com o avanço dos computadores e da descoberta de novas técnicas na área da Inteligência Artificial, têm surgido novos e sofisticados métodos. Um exemplo é um sistema de *Credit Scoring* utilizando Raciocínio Baseados em Casos (Kolodner, 1993).

C) Experiência Anterior versus Métodos quantitativos

A utilização de modelos construídos por métodos quantitativos apresenta as seguintes vantagens sobre o uso da Experiência Anterior:

- **Consistência:** Como diferentes analistas possuem diferentes critérios de julgamento, o modelo elimina a subjetividade na decisão permitindo à instituição financeira uma maior consistência nas suas decisões de crédito.
- **Flexibilidade:** os modelos quantitativos possuem maior flexibilidade no sentido de que podem se adaptar a mudanças de situação econômica. Eles podem ser reconstruídos a partir de um novo conjunto de treinamento que reflita a atual situação econômica. Esta flexibilidade não ocorre no método subjetivo. Neste método, consome-se muito tempo para que o analista adquira experiência suficiente para tratar com novas situações. Além disso, a Experiência Anterior não pode ser transferida facilmente de uma geração para outra pelo fato de não ser necessariamente adequada para a geração seguinte. Devido a esta falta de flexibilidade da Experiência Anterior, conclui Silva (1988):

“Daí, a necessidade dos analistas e/ou gestores de crédito utilizarem também outros métodos como simulações e recursos quantitativos, que, adicionados a uma experiência atualizada, lhes proporcionarão adequado domínio dos fatores envolvidos no processo decisório do crédito”.

- **Rapidez:** aumenta a rapidez na aprovação de crédito permitindo que as instituições financeiras analisem grandes quantidades diárias de propostas. Além disso, ao invés do analista de crédito analisar fatores que podem ser tratados pelos modelos, seu tempo pode ser empregado para grandes negócios, analisar empresas, acompanhar a economia etc...
- **Segurança:** acrescenta mais segurança ao analista de crédito, pois os métodos quantitativos confirmam empiricamente a sua decisão de crédito, com base no grande número de amostras reais empregadas no modelo.

É possível acrescentar ainda uma outra vantagem: não há discriminação racial ou étnica na ARC. Os métodos quantitativos, porém, apresentam algumas desvantagens:

- Com o passar do tempo, a situação econômica altera-se e os modelos construídos com base em amostras (conjunto de treinamento) desatualizadas perdem sua eficácia.
- Mesmo o modelo apresentando uma baixa taxa de erro (cerca de 5%), esta pequena taxa pode ainda ser alta e até fatal para alguns tipos de empresas (Silva, 1988).
- Os modelos são dependentes da região geográfica em que foram geradas as amostras.

A seguir serão apresentados métodos quantitativos clássicos de diversas áreas utilizados em ARC.

B.3.1 CREDIT SCORING

O *Credit Scoring* é um método de classificação direcionado para ARC. Constitui-se de uma tabela de atributos e valores. Os atributos representam as características do cliente. Cada atributo assume um conjunto de valores e para cada valor está associado um peso. Um exemplo de tabela de *Credit Scoring* é apresentado na Tabela B.1 (Jensen, 1992).

Utiliza-se a tabela de *Credit Scoring* do seguinte modo: para cada linha obtêm-se o valor que satisfaz ao atributo do cliente. Somam-se todos os pesos obtidos de cada linha. Se a soma ultrapassa um determinado limite (no caso da Tabela B.1 este limite é 200) então o crédito é concedido ao cliente, caso contrário não há concessão. Geralmente os pesos da tabela são calculados utilizando técnicas estatísticas, como Análise Discriminante ou Regressão Multivariada, com base em dados de empréstimos passados. Os mais conhecidos modelos de *Credit Scoring* são os modelos Fair-Issac desenvolvidos pela agência de crédito americana TRW (Jensen, 1992). A seguir será apresentado a método estatístico da Análise Discriminante.

Tabela B.1.Um *Credit Scoring*

ATRIBUTOS VALORES →

Own/Rent	Owns 45	Rents 18	All Other 24		
Years with employer	Under 1 year 15	1-2 years 22	3-9 years 26	10-12 years 29	13 years 36
Credit cards	Card 19	No card 0			
Store Account	Yes 36	No 0			
Bank Account	Check & savings 50	Current 31	Deposit 32	No account 5	
Occupation	Professional 29	Office staff 25	Production 15	Sales 22	All other 15
Previous Account	Unsatisfactory 0	New 55	Satisfactory 87		
Credit Bureau	No file 15	Derogatory -33	Satisfactory 24	Three satisfactory 30	

B.3.2 ANÁLISE DISCRIMINANTE

A Análise Discriminante (Fisher, 1936) é uma importante técnica estatística utilizada em problemas de classificação. É uma técnica utilizada em ARC e problemas similares a muitas décadas (Altman, 1968). Ela também tem sido bastante utilizada no Brasil (Altman et al, 1979; Silva, 1987). A seguir será apresentado uma introdução a esta técnica.

A) Classificação em duas classes

Nesta seção será abordado o problema de classificação em duas classes. Considere que objetos do mundo real pode ser representado por um vetor $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$. Considere também que o vetor $\mathbf{x}_i$ pertence a uma das duas classes C_1 ou C_2 . Como exemplo, o vetor $\mathbf{x}_i$ poderia representar uma empresa, onde $x_{i1}, x_{i2}, \dots, x_{in}$ seriam índices financeiros de tal empresa. As classes C_1 e C_2 poderiam representar, por exemplo: C_1 = “Empresa adimplente” e C_2 = “Empresa inadimplente” ou então C_1 = “Empresas boas” e C_2 = “Empresas falidas” e assim por diante.

Considere um conjunto de vetores $\{\mathbf{x}_i, i=1 \dots p_1\}$ pertencente a classe C_1 e um conjunto de vetores $\{\mathbf{x}_i, i=1 \dots p_2\}$ pertencente a classe C_2 . A partir desses dois conjuntos a Análise Discriminante constrói um modelo para classificar um dado vetor $\mathbf{x}_i$ na sua respectiva classe. A construção do modelo é descrita como segue.

Cada vetor $\mathbf{x}_i$ é projetado para um espaço unidimensional. Tal projeção é dada por:

$$Z_i = \sum_{j=1}^n w_j x_{ij} = \mathbf{w}^T \mathbf{x}_i \quad (\text{B.3.2.1})$$

O vetor de coeficientes $\mathbf{w} = [w_1, w_2, \dots, w_n]^T$ é determinado pela Análise Discriminante conforme mostrado mais tarde. A interpretação geométrica da Análise Discriminante é mostrada na Figura B.1. Cada elipse representa a região onde há maior densidade de vetores $\mathbf{x}_i$ de uma determinada classe. Os vetores $\mathbf{x}_i$ são projetados sobre o eixo Z . É no eixo discriminante Z que ocorre a menor sobreposição (maior separação) entre as duas regiões de classes diferentes.

Considere os vetores média das duas classes: μ_1 e μ_2 dados por:

$$\mu_1 = \frac{1}{p_1} \sum_{\mathbf{x}_i \in C_1} \mathbf{x}_i, \quad \mu_2 = \frac{1}{p_2} \sum_{\mathbf{x}_i \in C_2} \mathbf{x}_i \quad (\text{B.3.2.2})$$

Onde, p_k é o número de pontos da classe C_k . As projeções dos vetores média μ_k sobre o eixo Z são dadas por:

$$\bar{Z}_1 = \mathbf{w}^T \mu_1 \quad (\text{B.3.2.3})$$

$$\bar{Z}_2 = \mathbf{w}^T \mu_2 \quad (\text{B.3.2.4})$$

O eixo discriminante Z é dividido em duas regiões (a região da classe C_1 e a região da classe C_2) por um ponto q na qual é dado por:

$$q = \frac{\bar{Z}_1 + \bar{Z}_2}{2} \quad (\text{B.3.2.5})$$

Assumindo $\bar{Z}_1 < \bar{Z}_2$, a classificação de um vetor $\mathbf{x}_i$ qualquer é realizado como segue: primeiro calcula-se $Z_i = \mathbf{w}^T \mathbf{x}_i$. Em seguida, se $Z_i < q$ então $\mathbf{x}_i$ pertencerá a classe C_1 , caso contrário, $\mathbf{x}_i$ pertencerá a classe C_2 . Vale frisar que erros de classificação podem ocorrer nas regiões onde as classes se sobrepõem. O cálculo do vetor $\mathbf{w}$ é mostrado a seguir.

Figura B.1. Interpretação geométrica da Análise Discriminante.

B) Determinação dos coeficientes

A direção do eixo discriminante Z é definido pelo vetor $\mathbf{w}$ que é obtido maximizando o critério de Fisher. A idéia básica deste critério é maximizar a distância entre classes, $\bar{Z}_1 - \bar{Z}_2$, padronizada em termos de covariância intra-classes. A covariância intra-classe s_k^2 da classe C_k é dada por:

$$s_k^2 = \sum_{x_i \in C_k} (Z_i - \bar{Z}_k)^2 \quad (\text{B.3.2.6})$$

E o critério de Fisher que será maximizado é dado por:

$$J(\mathbf{w}) = \frac{(\bar{Z}_1 - \bar{Z}_2)^2}{s_1^2 + s_2^2} \quad (\text{B.3.2.7})$$

Ao maximizar este critério é assumido que cada componente $x_{i1}, x_{i2}, \dots, x_{in}$ dos vetor $\mathbf{x}_i$ possui distribuição normal. É assumido, também, que a matriz de covariância é a mesma para as duas classes, isto é, as dispersões são as mesmas para as duas classes. Isto quer dizer que as duas elipses da Figura B.1 devem apresentar a mesma forma ou formato. Já o número de vetores $\mathbf{x}_i$ não precisa ser o mesmo nas duas classes. Para rescrever a Equação (B.3.2.7) em função de $\mathbf{w}$, são utilizadas as Equações (B.3.2.1), (B.3.2.3) e (B.3.2.4) na qual tem-se:

$$J(\mathbf{w}) = \frac{\left\{ \sum_j w_j (\mu_{1j} - \mu_{2j}) \right\}^2}{\sum_j \sum_k w_j w_k S_{jk}} \quad (\text{B.3.2.8})$$

Onde,

$$S_{jk} = \sum_{\mathbf{x}_i \in C_1} (x_{ij} - \mu_{1j})(x_{ik} - \mu_{1k}) + \sum_{\mathbf{x}_i \in C_2} (x_{ij} - \mu_{2j})(x_{ik} - \mu_{2k}) \quad (\text{B.3.2.9})$$

Colocando as Equações (B.3.2.8) e (B.3.2.9) na forma matricial tem-se:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_E \mathbf{w}}{\mathbf{w}^T \mathbf{S}_I \mathbf{w}} \quad (\text{B.3.2.10})$$

Onde $\mathbf{S}_E$ é a matriz de covariância entre-classes na qual é dada por:

$$\mathbf{S}_E = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \quad (\text{B.3.2.11})$$

E $\mathbf{S}_I$ é a matriz de covariância intra-classes na qual é dada por:

$$\mathbf{S}_I = \sum_{\mathbf{x}_i \in C_1} (\mathbf{x}_i - \mu_1)(\mathbf{x}_i - \mu_1)^T + \sum_{\mathbf{x}_i \in C_2} (\mathbf{x}_i - \mu_2)(\mathbf{x}_i - \mu_2)^T \quad (\text{B.3.2.12})$$

O valor máximo da Equação (B.3.2.10) é obtido diferenciando-a em relação a cada coeficiente w_j ($j=1,...,n$) e igualando a zero. Assim, obtém-se:

$$(\mathbf{w}^T \mathbf{S}_E \mathbf{w}) \mathbf{S}_I \mathbf{w} = (\mathbf{w}^T \mathbf{S}_I \mathbf{w}) \mathbf{S}_E \mathbf{w} \quad (\text{B.3.2.13})$$

Como não há interesse no módulo do vetor $\mathbf{w}$, apenas na sua direção, então os fatores de escala (entre parênteses) da Equação (B.3.2.13) podem ser desprezados. Além disso, $\mathbf{S}_E \mathbf{w}$ pode ser substituído por $\mu_2 - \mu_1$ por que tem a mesma direção, conforme mostra a Equação (B.3.2.11). Com estas alterações na Equação (B.3.2.13) obtém-se:

$$\mu_1 - \mu_2 \propto \mathbf{S}_I \mathbf{w} \quad (\text{B.3.2.14})$$

$$\mathbf{w} \propto \mathbf{S}_I^{-1} (\mu_1 - \mu_2) \quad (\text{B.3.2.15})$$

Portanto, o vetor $\mathbf{w}$ é determinado por um sistema linear. A seguir será visto um exemplo prático.

B) Um exemplo prático

Neste exemplo é utilizado um pequeno conjunto de dados extraído de (Silva, 1988). Ver Tabela B.2. O objetivo é determinar uma função discriminante, Equação (B.3.2.1), que permita classificar uma Empresa em uma das duas classes: C_1 = “Empresas boas” e C_2 = “Empresas falidas”. Cada empresa é representada por um vetor $\mathbf{x}_i$ cujos componentes são dados pelos seguintes índices financeiros:

$$\begin{aligned}x_{i1} &= \text{endividamento} = (\text{exigível total/patrimônio líquido}). \\x_{i2} &= \text{rentabilidade} = (\text{lucro líquido/patrimônio líquido}).\end{aligned}$$

A partir das Equações (B.3.2.2), obtém-se:

$$\mu_1 - \mu_2 = \begin{bmatrix} 0,990 \\ 0,211 \end{bmatrix} - \begin{bmatrix} 4,307 \\ -0,010 \end{bmatrix} = \begin{bmatrix} -3,317 \\ 0,221 \end{bmatrix}$$

Os elementos da matriz de covariância intra-classes $\mathbf{S}_I$ são obtidos pela Equação (B.3.2.9) ou pela Equação (B.3.2.12).

$$\mathbf{S}_I = \begin{bmatrix} 54,4582 & -4,0596 \\ -4,0596 & 0,6018 \end{bmatrix}$$

É então obtida a inversa de $\mathbf{S}_I$:

$$\mathbf{S}^{-1} = \begin{bmatrix} 0,03694 & 0,24917 \\ 0,24917 & 3,34251 \end{bmatrix}$$

Aplicando Equação (B.3.2.15) tem-se:

$$\mathbf{w} = \begin{bmatrix} 0,03694 & 0,24917 \\ 0,24917 & 3,34251 \end{bmatrix} \begin{bmatrix} -3,317 \\ 0,221 \end{bmatrix} = \begin{bmatrix} -0,0675 \\ -0,0878 \end{bmatrix}$$

Por fim, a função discriminante desejada é:

$$Z = -0,0675x_1 - 0,0878x_2$$

Onde o ponto de separação entre os grupos é calculado abaixo:

$$\begin{aligned}\bar{Z}_1 &= -0,0675(0,990) - 0,0878(0,211) = -0,0854 \\ \bar{Z}_2 &= -0,0675(4,307) - 0,0878(-0,01) = -0,2898 \\ q &= (\bar{Z}_1 + \bar{Z}_2)/2 = (-0,0854 - 0,2898)/2 = -0,1876\end{aligned}$$

Aqui termina esta introdução a Análise Discriminante¹⁰. A generalização deste modelo para várias classes é encontrado em livros sobre o assunto, como por exemplo (Duda e Hart, 1973). A seguir será apresentado uma outra técnica (Redes Neurais) para construir modelos para classificação.

Tabela B.2. Conjunto de dados - Empresas boas e empresas falidas

Empresas boas		Empresas falidas	
x_1	x_2	x_1	x_2
1,34	0,24	7,45	-0,14
1,21	0,20	3,21	-0,02
1,48	0,36	4,27	0,06
0,81	0,15	1,85	-0,08
1,15	0,21	1,45	0,11
0,66	0,20	9,25	-0,62
0,73	0,17	2,76	0,25
0,69	0,29	3,54	0,01
1,53	0,17	4,88	0,25
0,30	0,12	4,41	0,08

B.3.4 REDES NEURAIS ARTIFICIAIS

As Redes Neurais Artificial têm sido aplicadas com sucesso em ARC e em áreas de natureza semelhante como Previsão de Falência (Trippi e Turban, 1993; Poddig, 1995; Wilson e Sharda, 1994). Vários bancos e companhias de crédito internacionais, tais como o Mellon Bank e o Visa International inc, estão usando Redes Neurais para estudar o comportamento do uso de cartões de crédito e para detectar transações potencialmente fraudulentas (Widrow et al, 1994). No Brasil, pesquisas sobre o assunto também têm sido realizadas (Almeida e Dumotier, 1996; Horst et al, 1998; Lacerda e Carvalho, 1998, Mendes et al, 1996).

As Redes Neurais foram apresentadas amplamente no capítulo 2 e apêndice A. Portanto, o estudo seguirá com a apresentação do método Árvore de Decisão.

¹⁰ Vale observar que a Análise Discriminante é muito semelhante às Redes Neurais Perceptron e Adaline (Bishop, 1996). A diferença está no aprendizado. O primeiro determina o vetor w maximizando o critério de Fisher, Eq(6.3.2.10) e o segundo minimizando o MSE (no caso da Rede Adaline) (ver capítulo 2).

B.3.5 ÁRVORE DE DECISÃO

Árvores de decisão (Breiman et al, 1984; Quinlan, 1986) são modelos de aprendizado de máquina utilizados em problemas de classificação. Em (Carter e Catlett, 1987) encontra-se uma aplicação de Árvore de Decisão em ARC. No Brasil, Horst et al (1998) traz um estudo comparativo entre Árvore de Decisão e Redes Neurais em ARC. As árvores são construídas a partir de um conjunto de exemplos¹¹ de treinamento e um conjunto de registros com uma estrutura de pares atributo e valor. A seguir será descrito um exemplo.

A) Um exemplo de Árvore de Decisão

Um conjunto de exemplos de treinamento é mostrado na Tabela B.3. O respectivo conjunto de atributos, mostrado na Tabela B.4, descreve os possíveis valores que um dado atributo pode assumir. Por exemplo, o atributo 'Histórico de crédito' pode assumir apenas dois valores (satisfatório e insatisfatório) no conjunto de treinamento. Já o atributo 'Anos no emprego' pode assumir qualquer valor numérico contínuo. O atributo 'Aceita/Rejeita' é um atributo especial denominado atributo de categoria, pois seus valores representam a categoria ou classe que um exemplo pode assumir.

Uma árvore de decisão é mostrada na Figura B.2. Tal árvore foi construída para classificar os exemplos do conjunto de treinamento da Tabela B.3. Cada nó da árvore é um teste relativo a um atributo. Os ramos do nó representa valores que o atributo pode assumir. As folhas (nós sem ramos da árvore) contêm uma classe, isto é, um dos valores de atributo de categoria 'Aceita/Rejeita'. Iniciando pela raiz da árvore (nó mais alto da árvore) e descendo pelos ramos até alcançar as folhas obtém-se a classe do exemplo. A construção desta Árvore de Decisão foi realizada pelo algoritmo ID3, o qual é mostrado a seguir.

Tabela B.3. Conjunto de exemplos de treinamento

Histórico de crédito	Divida	Anos de residência	Casa própria	Anos no emprego	Aceita/ Rejeita
	Renda				
Satisfatório	19,8 %	30	Sim	29	Aceita
Satisfatório	10,2 %	23	Não	20	Aceita
Satisfatório	18,3 %	13	Não	10	Aceita
Satisfatório	34,0 %	8	Sim	5	Aceita
Insatisfatório	19,1 %	9	Sim	2	Aceita
Satisfatório	50,4 %	10	Sim	8	Rejeita
Insatisfatório	17,9 %	14	Não	11	Rejeita
Insatisfatório	32,8 %	8	Sim	4	Rejeita
Insatisfatório	33,0 %	5	Sim	3	Rejeita
Insatisfatório	62,3 %	2	Não	1	Rejeita

¹¹ O termo "Exemplo", utilizado em Aprendizado de Máquina, é empregado com o mesmo significado de "Padrões" no contexto de Redes Neurais e de "Observações" no contexto de Estatística.

Tabela B.4. Conjunto de atributos.

Atributos	Possíveis valores
Histórico de crédito	Satisfatório, Insatisfatório
Dívida/Renda	Contínuo
Anos de residência	Contínuo
Casa própria	Sim, Não
Anos no emprego	Contínuo
Aceita-Rejeita	Aceita, Rejeita

Figura B.2. Um árvore de decisão obtida com ID3

B) O Algoritmo ID3

O algoritmo ID3 (Quinlan, 1986) é apresentado no Algoritmo B.1. Ao construir uma árvore de decisão, o ID3 coloca inicialmente nos nós da árvore os atributos “mais informativos”. A medida para definir os atributos “mais informativos” é baseada na Entropia, noção originária da Física e introduzida por Shannon (1948) na Teoria da Informação. A utilização deste critério permite gerar árvores menores e mais eficientes. A seguir será descrito o conceito de Entropia e como utilizar este conceito para escolher quais atributos devem ser aparecer primeiro na árvore.

Algoritmo B.1. ID3 (Árvore de decisão)

```
função ID3 ( T : Conjunto de exemplos de treinamento
              C : Atributo de categoria
              A : Conjunto de atributos não categóricos )
início
  Criar um nó que será denominado NÓ
  se (todos exemplos de T possuem o mesmo valor no atributo de categoria C) então
    Retorne NÓ com o valor do atributo de categoria C
  fim se
  se (A está vazio) então
    Retorne NÓ com a classe que ocorre mais vezes no atributo de categoria C dos exemplos
    contidos em T
    /* Note que aqui alguns exemplos podem ser classificados erroneamente */
  fim se
  Seja X o atributo de A com maior ganho de informação (menor Entropia)
  para cada valor V de X faça
    Crie um RAMO para NÓ
    Seja T(V) o subconjunto de exemplos de T tal que X = V
    se (T(V) está vazio) então
      Crie um NÓ_FOLHA para RAMO
      Atribua ao NÓ_FOLHA a classe que ocorre mais vezes no atributo de categoria C dos
      exemplos contidos em T.
      /* Aqui também alguns exemplos podem ser classificados erroneamente */
    senão
      Atribua a RAMO a árvore criada por ID3(T(V), C, A-{X})
    fim se
  fim para
fim
```

C) Entropia

Sejam os eventos $E_1, E_2, \dots, E_n$ e a distribuição de probabilidade $P = (p_1, p_2, p_3, \dots, p_n)$, onde p_i é a probabilidade do evento E_i ocorrer. Considere a seguinte Equação:

$$I(E_i) = -\log_2(p_i) \quad (\text{B.3.4.1})$$

Ela representa uma medida de incerteza, ou equivalentemente, o “grau de surpresa” do evento E_i . Pode ser interpretada também como a quantidade de informação carregada pelo evento E_i . Devido a esta última interpretação, é usual o logaritmo da Equação (B.3.4.1) ter base 2, com o resultado dado em bits. Antes da apresentação de justificativas intuitivas desta medida, será definido o conceito de Entropia. Entropia é a média desta medida de incerteza tomada sobre todos os eventos E_i . Portanto, é dada por:

$$H(E) = - \sum_{i=1}^n p_i \log_2(p_i) \quad (\text{B.3.4.2})$$

Considere, como exemplo, as duas distribuições de probabilidade:

$$(a) \quad A = [A_1, A_2] \quad P_A = \left(\frac{1}{512}, \frac{511}{512} \right)$$

$$(b) \quad B = [B_1, B_2] \quad P_B = (1/2, 1/2)$$

As suas respectivas Entropias são:

$$(a) \quad H(A) = - \left(\frac{1}{512} \log_2 \left(\frac{1}{512} \right) + \frac{511}{512} \log_2 \left(\frac{511}{512} \right) \right) = 0,0204 \text{ bit}$$

$$(b) \quad H(B) = -(1/2 \log_2(1/2) + 1/2 \log_2(1/2)) = 1 \text{ bit}$$

No sistema (a), o evento A_2 quase sempre ocorre, devido a sua alta probabilidade. Então não haverá quase “surpresa” ou informação a ser aprendida se A_2 ocorrer. Já no sistema (b) é difícil adivinhar qual dos eventos B_1 e B_2 ocorrerá, visto que possuem a mesma probabilidade. Portanto, faz sentido que seja atribuída ao sistema (a) uma Entropia ou incerteza baixa e que ao sistema (b) seja atribuída uma Entropia ou incerteza alta. Vale observar que quanto mais uniforme for a distribuição de probabilidade maior será a Entropia. A seguir a Entropia será utilizada para seleccionar o atributo “mais informativo”.

D) Medindo a Entropia de um atributo

Considere T o conjunto de todos os exemplos e os subconjuntos disjuntos $C_1, C_2, \dots, C_k$ uma partição de T sobre os valores do atributo de categoria. A informação necessária para classificar um exemplo de T é dada por:

$$\text{INFO}(T) = H(P) \quad (\text{B.3.4.3})$$

Onde $H(P)$ é a entropia da distribuição de probabilidade P na qual é dada por:

$$P = \left(\frac{|C_1|}{|T|}, \frac{|C_2|}{|T|}, \dots, \frac{|C_k|}{|T|} \right) \quad (\text{B.3.4.4})$$

Para o conjunto da Tabela B.3, tem-se $\text{INFO}(T) = H(5/10, 5/10) = 1.0$. Considere $V_1, V_2, \dots, V_n$ uma nova partição do conjunto T sobre os valores do atributo não categórico X . Considere ainda que já foi obtido o valor de X . Assim, a informação necessária para classificar um exemplo de T é dada pela média das informações necessárias para classificar um exemplo em cada partição V_i , isto é:

$$\text{INFO}(X,T) = \sum_{i=1}^n \frac{|V_i|}{|T|} \text{INFO}(V_i) \quad (\text{B.3.4.5})$$

Em outras palavras, a Equação (B.3.4.5) representa a entropia condicional $H(T|X)$ que é a entropia de T dado que o valor de X já foi obtido. Para o atributo $X = \text{'Casa Própria'}$ da Tabela B.3, tem-se:

$$\begin{aligned} \text{INFO}(\text{Casa Própria}, T) &= \frac{7}{10} \text{INFO}(\text{Sim}) + \frac{3}{10} \text{INFO}(\text{Nao}) = \\ &= -\frac{7}{10} \left[4/7 \log_2(4/7) + 3/7 \log_2(3/7) \right] - \frac{3}{10} \left[1/3 \log_2(1/3) + 2/3 \log_2(2/3) \right] = 0,965 \end{aligned}$$

E) Definição de GAIN

Por fim, é definido $\text{GAIN}(X,T)$, que representa o ganho de informação devido ao fato de já se conhecer o valor do atributo X :

$$\text{GAIN}(X,T) = \text{INFO}(T) - \text{INFO}(X,T) \quad (\text{B.3.4.6})$$

Aquele atributo que tiver o maior GAIN será preferido sobre os demais pois melhor classifica os exemplos. Por exemplo, considere os quatro histogramas mostrados na Figura B.3 que, por sua vez, foram gerados particionando os 10 exemplos da Tabela B.3 com base nos valores de dois atributos. Observa-se que os histogramas construídos com base no atributo 'Casa própria' possuem distribuição mais uniforme que os histogramas do atributo 'Histórico de Crédito'. Portanto, será "mais fácil" identificar uma classe pelo atributo 'Histórico de Crédito' do que pelo atributo 'Casa Própria'. Isto está correto, pois $\text{GAIN}(\text{'Histórico de Crédito'}) = 0,278 > \text{GAIN}(\text{'Casa própria'}) = 0,035$. Ou seja, quanto mais uniforme a distribuição de probabilidade, maior a entropia e, conseqüentemente, menor o GAIN.

Figura B.3. Histogramas de atributos

F) Definição de GAIN RATIO

O conceito de GAIN definido na seção anterior favorece os atributos que possuem muitos valores diferentes. Por exemplo, considere um atributo X do conjunto de exemplos T na qual X possui um valor diferente para cada exemplo de T. Neste caso, o valor de $INFO(X, T)$ é zero, na qual implica que $GAIN(X, T)$ é o máximo possível. Este problema é amenizado utilizando a seguinte medida no lugar de $GAIN(X, T)$:

$$GAIN_RATIO(X, T) = \frac{GAIN(X, T)}{SPLIT_INFO(X, T)} \quad (B.3.4.7)$$

Onde $SPLIT_INFO(X, T)$ é a informação devido a divisão de T sobre os valores do atributo X na qual é dado por:

$$SPLIT_INFO(X, T) = H(P) \quad (B.3.4.8)$$

Onde $H(P)$ é a entropia da distribuição de probabilidade P na qual é dada por:

$$P = \left(\frac{|V_1|}{|T|}, \frac{|V_2|}{|T|}, \dots, \frac{|V_n|}{|T|} \right) \quad (B.3.4.9)$$

Onde $\{V_1, V_2, \dots, V_n\}$ é a partição de T sobre os valores do atributo X. Por exemplo, o $GAIN_RADIO$ do atributo 'Casa Própria' é calculado do seguinte modo:

$$SPLIT_INFO('Casa Própria') = -(6/10) \log_2(6/10) - (4/10) \log_2(4/10) = 0,971$$

$$GAIN_RATIO('Casa Própria') = \frac{0,035}{0,971} = 0,036$$

G) Gerando regras através da Árvore de Decisão

Uma das vantagens da Árvore de Decisão sobre os demais métodos citados neste texto é a sua capacidade de derivar regras de produções que explicam suas classificações. É fácil derivar uma regra de uma Árvore de Decisão: escreve-se uma regra para cada caminho do nó raiz até a folha. O lado esquerdo da regra é construído a partir dos atributos dos nós e dos valores dos ramos neste caminho. Há ainda vários métodos para simplificar tal conjunto de regras (Quinlan, 1993).