

# Redes Neurais Artificiais

para engenharia e ciências aplicadas

curso prático



Ivan Nunes da Silva

Danilo Hernane Spatti

Rogério Andrade Flauzino



Ivan Nunes da Silva  
Danilo Hernane Spatti  
Rogério Andrade Flauzino

Universidade de São Paulo

**Redes Neurais Artificiais**  
para engenharia e ciências aplicadas

**Artliber**  
editorial

Copyright© 2010 by Artíliber Editora Ltda.

Revisão:

*Maria Atenácia M. Eckendorff*

Capa e composição eletrônica:

*Espaco Editorial*

**Dados Internacionais de Catalogação na Publicação (CIP)  
(Câmara Brasileira do Livro)**

Silva, Ivan Nunes da

Redes neurais artificiais: para engenharia e ciências aplicadas / Ivan Nunes da Silva; Dâniel Hernane Spatti; Rogério Andrade Flauzino. – São Paulo: Artíliber, 2010.

ISBN: 978-85-88098-53-4

1. Redes neurais 2. Inteligências artificiais 3. Arquitetura de redes neurais  
I. Título II. Spatti, Dâniel Hernane; Flauzino, Rogério Andrade

10-0725

CDD-004.032.62

Índices para catálogo sistemático:

1. Redes neurais: Engenharia e ciências aplicadas 004.032.62

2010

Todos os direitos desta edição são reservados à

**Artíliber Editora Ltda.**

Av. Diógenes Ribeiro de Lima, 3294

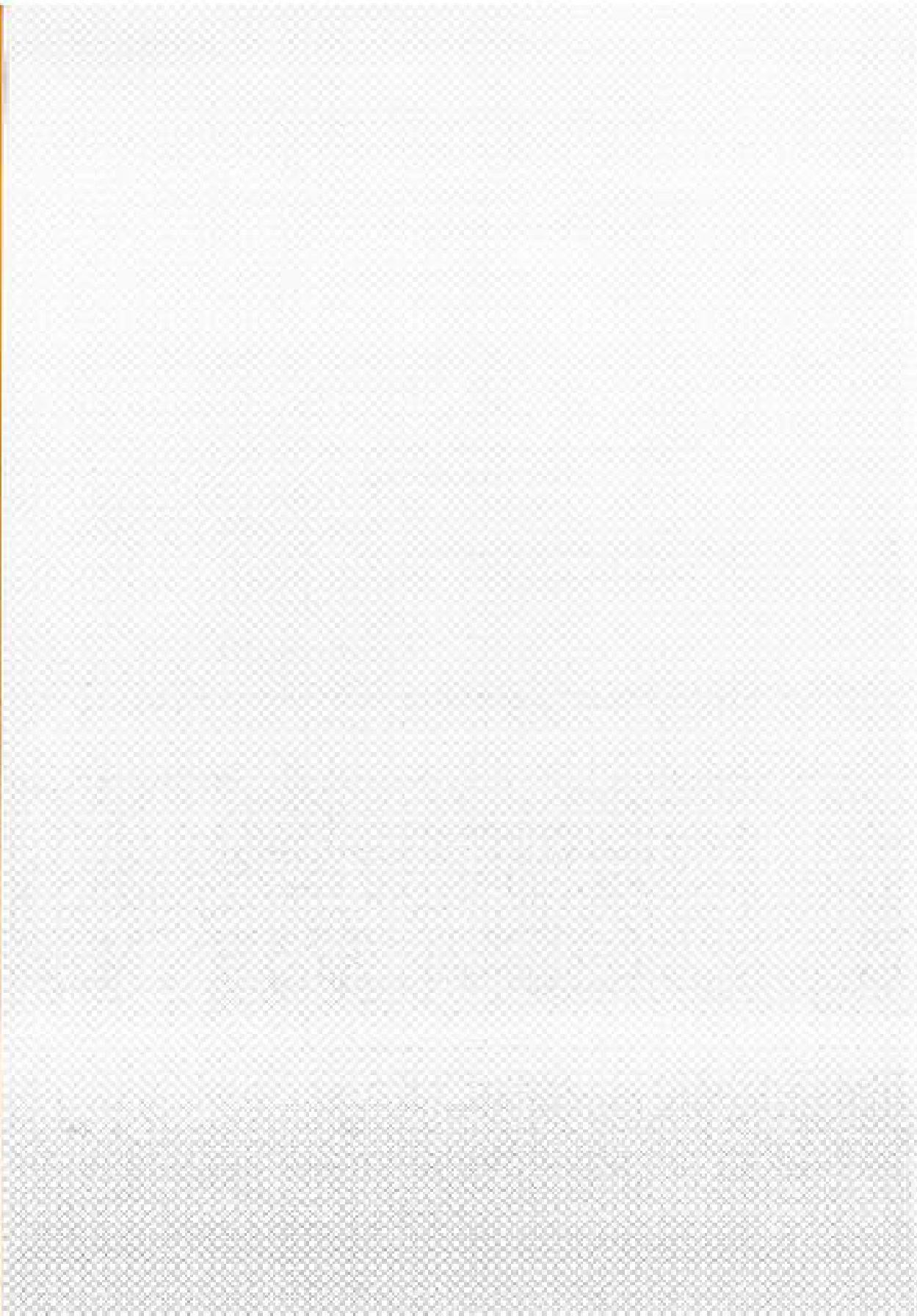
05083-010 – São Paulo – SP – Brasil

Tel.: (11) 3832-5223 Fax: (11) 3832-5489

[info@artliber.com.br](mailto:info@artliber.com.br)

[www.artliber.com.br](http://www.artliber.com.br)

*Para  
Sílvia, Fernanda e Selângel*



# Sumário

---

Prefácio.....	13
Organização.....	15
Agradecimentos.....	17
<b>Parte I – Arquiteturas de redes neurais artificiais e seus aspectos teóricos.....</b>	<b>19</b>
<b>Capítulo 1 – Introdução.....</b>	<b>21</b>
1.1    Conceitos iniciais.....	24
1.1.1    Características principais.....	24
1.1.2    Resumo histórico.....	25
1.1.3    Potenciais áreas de aplicações.....	27
1.2    Neurônio biológico.....	29
1.3    Neurônio artificial.....	33
1.3.1    Funções de ativação parcialmente diferenciáveis.....	36
1.3.2    Funções de ativação totalmente diferenciáveis.....	38
1.4    Parâmetros de desempenho.....	42
1.5    Exercícios.....	43
<b>Capítulo 2 – Arquiteturas de redes neurais artificiais e processos de treinamento.....</b>	<b>45</b>
2.1    Introdução.....	45
2.2    Principais arquiteturas de redes neurais artificiais.....	46
2.2.1    Arquitetura <i>feedforward</i> de camada simples.....	46
2.2.2    Arquitetura <i>feedforward</i> de camadas múltiplas.....	47
2.2.3    Arquitetura recorrente ou realimentada.....	49

2.2.4 Arquitetura em estrutura reticulada .....	50
2.3 Procedimentos de treinamento e aspectos de aprendizado .....	50
2.3.1 Treinamento supervisionado .....	51
2.3.2 Treinamento não-supervisionado .....	52
2.3.3 Treinamento com reforço .....	53
2.3.4 Aprendizagem usando teste de padrões ( <i>off-line</i> ) .....	53
2.3.5 Aprendizagem usando padrão-por-padrão ( <i>on-line</i> ) .....	54
2.4 Exercícios .....	54
 Capítulo 3 – Rede <i>Perceptron</i> .....	57
3.1 Introdução .....	57
3.2 Princípio de funcionamento do <i>Perceptron</i> .....	59
3.3 Análise matemática do <i>Perceptron</i> .....	61
3.4 Processo de treinamento do <i>Perceptron</i> .....	63
3.5 Exercícios .....	68
3.6 Projeto prático .....	70
 Capítulo 4 – Redes <i>Adaline</i> e regra Delta .....	73
4.1 Introdução .....	73
4.2 Princípio de funcionamento do <i>Adaline</i> .....	74
4.3 Processo de treinamento do <i>Adaline</i> .....	76
4.4 Comparação entre o processo de treinamento do <i>Adaline</i> e <i>Perceptron</i> .....	83
4.5 Exercícios .....	86
4.6 Projeto prático .....	87
 Capítulo 5 – Redes <i>Perceptron</i> multicamadas .....	91
5.1 Introdução .....	91
5.2 Princípio de funcionamento do <i>Perceptron</i> multicamadas .....	92
5.3 Processo de treinamento do <i>Perceptron</i> multicamadas .....	94
5.3.1 Derivação do algoritmo <i>backpropagation</i> .....	95
5.3.2 Implementação do algoritmo <i>backpropagation</i> .....	108
5.3.3 Versões aperfeiçoadas do algoritmo <i>backpropagation</i> .....	111
5.4 Aplicabilidade das redes <i>Perceptron</i> multicamadas .....	120
5.4.1 Problemas envolvendo classificação de padrões .....	121
5.4.2 Problemas envolvendo aproximação funcional .....	132
5.4.3 Problemas envolvendo sistemas variantes no tempo .....	137
5.5 Aspectos de especificação topológica de redes PMC .....	146
5.5.1 Aspectos de métodos de validação cruzada .....	147
5.5.2 Aspectos de subconjuntos de treinamento e teste .....	151
5.5.3 Aspectos de situações de <i>overfitting</i> e <i>underfitting</i> .....	153
5.5.4 Aspectos de inclusão de parada antecipada .....	155

5.5.5	Aspectos de convergência para mínimos locais.....	157
5.6	Aspectos de implementação de redes <i>Perceptrons</i> multicamadas .....	158
5.7	Exercícios.....	163
5.8	Projeto prático 1 (aproximação de funções) .....	164
5.9	Projeto prático 2 (classificação de padrões) .....	166
5.10	Projeto prático 3 (sistemas variantes no tempo) .....	169
 Capítulo 6 – Redes de funções de base radial (RBF).....		173
6.1	Introdução .....	173
6.2	Processo de treinamento de redes RBF .....	174
6.2.1	Ajuste dos neurônios da camada intermédia (estágio I) .....	174
6.2.2	Ajuste dos neurônios da camada de saída (estágio II) .....	181
6.3	Aplicabilidades das redes RBF.....	183
6.4	Exercícios.....	190
6.5	Projeto prático 1 (classificação de padrões) .....	191
6.6	Projeto prático 2 (aproximação de funções) .....	194
 Capítulo 7 – Redes recorrentes de Hopfield.....		199
7.1	Introdução .....	199
7.2	Princípio de funcionamento da rede de Hopfield .....	201
7.3	Condições de estabilidade da rede de Hopfield .....	204
7.4	Memórias associativas.....	207
7.4.1	Método do pendulo externo .....	208
7.4.2	Método da matriz pseudo-inversa.....	210
7.4.3	Capacidade de armazenamento das memórias .....	211
7.5	Aspectos de projeto de redes de Hopfield .....	213
7.6	Aspectos de implementação em hardware .....	215
7.7	Exercícios.....	217
7.8	Projeto prático .....	218
 Capítulo 8 – Redes auto-organizáveis de Kohonen .....		221
8.1	Introdução .....	221
8.2	Processo de aprendizado competitivo .....	222
8.3	Mapas auto-organizáveis de Kohonen (SOM) .....	229
8.4	Exercícios.....	237
8.5	Projeto prático .....	238
 Capítulo 9 – Redes <i>LVQ</i> e <i>counter-propagation</i> .....		243
9.1	Introdução .....	243
9.2	Processo de quantização vetorial .....	244
9.3	Redes <i>LVQ</i> ( <i>learning vector quantization</i> ) .....	247

9.3.1	Algoritmo de treinamento <i>LJ/Q-1</i> .....	248
9.3.2	Algoritmo de treinamento <i>LJ/Q-2</i> .....	252
9.4	Redes <i>master-propagator</i> .....	254
9.4.1	Aspectos da camada <i>satélite</i> .....	256
9.4.2	Algoritmo de treinamento da rede <i>master-propagator</i> .....	257
9.5	Exercícios.....	259
9.6	Projeto prático .....	260
<b>Capítulo 10 – Redes ART (adaptive resonance theory) .....</b>		<b>263</b>
10.1	Introdução .....	263
10.2	Estrutura topológica da rede <i>ART-1</i> .....	265
10.3	Princípio da ressonância adaptativa.....	268
10.4	Aspectos de aprendizado da rede <i>ART-1</i> .....	269
10.5	Algoritmo de treinamento da rede <i>ART-1</i> .....	279
10.6	Aspectos da versão original da rede <i>ART-1</i> .....	281
10.7	Exercícios.....	284
10.8	Projeto prático .....	285
<b>Parte II – Aplicações de redes neurais artificiais em problemas de engenharia e ciências aplicadas .....</b>		<b>287</b>
<b>Capítulo 11 – Estimação da qualidade global de café utilizando o <i>Préprocesse</i> multicamadas .....</b>		<b>289</b>
11.1	Introdução .....	289
11.2	Características da Rede <i>PMC</i> .....	290
11.3	Resultados computacionais.....	292
<b>Capítulo 12 – Análise do tráfego de redes de computadores utilizando protocolo <i>SNMP</i> e rede <i>LJ/Q</i> .....</b>		<b>295</b>
12.1	Introdução .....	295
12.2	Características da rede <i>LJ/Q</i> .....	297
12.3	Resultados computacionais .....	299
<b>Capítulo 13 – Previsão de tendências do mercado de ações utilizando redes recorrentes .....</b>		<b>301</b>
13.1	Introdução .....	301
13.2	Características da rede recorrente .....	303
13.3	Resultados computacionais .....	304
<b>Capítulo 14 – Sistema para diagnóstico de doenças utilizando redes <i>ART</i> .....</b>		<b>309</b>
14.1	Introdução .....	309

14.2	Características da Rede ART.....	311
14.3	Resultados computacionais.....	312
<b>Capítulo 15 – Identificação de padrões de aderentes em pó de café usando mapas de Kohonen.....</b>		<b>315</b>
15.1	Introdução.....	315
15.2	Características da rede de Kohonen.....	316
15.3	Resultados computacionais.....	319
<b>Capítulo 16 – Reconhecimento de distúrbios relacionados à qualidade da energia elétrica utilizando redes PMC.....</b>		<b>321</b>
16.1	Introdução.....	321
16.2	Características da rede PMC.....	325
16.3	Resultados computacionais.....	326
<b>Capítulo 17 – Controle de trajetória de robôs móveis usando sistemas fuzzy e redes PMC.....</b>		<b>329</b>
17.1	Introdução.....	329
17.2	Características da rede PMC.....	331
17.3	Resultados computacionais.....	334
<b>Capítulo 18 – Método para classificação de tomates usando visão computacional e redes PMC.....</b>		<b>339</b>
18.1	Introdução.....	339
18.2	Características da rede neural.....	341
18.3	Resultados computacionais.....	345
<b>Capítulo 19 – Análise de desempenho de redes RBF e PMC em classificação de padrões.....</b>		<b>347</b>
19.1	Introdução.....	347
19.2	Características das redes RBF e PMC.....	348
19.3	Resultados computacionais.....	349
<b>Capítulo 20 – Resolução de problemas de otimização com restrições por redes de Hopfield.....</b>		<b>355</b>
20.1	Introdução.....	355
20.2	Características da rede de Hopfield.....	357
20.3	Mapeamento de problemas de otimização pela rede de Hopfield.....	359
20.4	Resultados computacionais.....	364
<b>Bibliografia.....</b>		<b>371</b>

Apéndice I.....	381
Apéndice II .....	383
Apéndice III.....	385
Apéndice IV.....	391
Apéndice V .....	395
Índice remissivo.....	397



## Prefácio

---

O que são redes neurais artificiais? Para que servem? Quais as suas potencialidades de aplicações práticas? Quais tipos de problemas podem solucionar?

A partir da consideração de tais questionamentos, o livro foi redigido com a preocupação primeira de atender aos diferentes perfis de leitores, que estejam interessados tanto na aquisição de conhecimentos sobre arquiteturas de redes neurais artificiais, como aqueles que estejam mais motivados pelas suas diversas aplicações (de cunho prático) em problemas do mundo real.

A sua audiência com caráter multidisciplinar, conforme poderá ser atestada nos diversos exercícios e exemplos aqui tratados estende-se para diversas áreas do conhecimento, tais como engenharias, ciências de computação, matemática, física, economia, finanças, estatística e neurociências. Adicionalmente, espera-se ainda que o livro possa ser também interessante para diversas outras áreas, cujas aplicações têm sido também foco de redes neurais artificiais, como em medicina, psicologia, química, farmácia, biologia, ecologia, geologia, etc.

Em relação ao perfil acadêmico e ao público-alvo do qual o livro está direcionado, tentou-se elaborar cada um de seus capítulos de maneira a dissecar, passo a passo, os conceitos relacionados em seus assuntos temáticos,

com níveis bem abrangentes de informações técnicas e teóricas. Assim, além de atender aos anseios daqueles profissionais, que desejam começar ou aprofundar os seus estudos sobre as redes neurais artificiais e suas potencialidades de aplicação, espera-se também que o material possa ser adotado como livro texto em disciplinas de graduação e pós-graduação que abordem o tema de redes neurais artificiais em seus conteúdos programáticos.

Complementarmente, buscou-se ainda redigir o texto usando uma linguagem acessível e que pudesse ser apreciado por profissionais, estudantes e pesquisadores, com perfis também autodidatas, como um guia direto e independente no que tange ao aprendizado de temas básicos e avançados relacionados às redes neurais artificiais. Para tanto, as exigências de pré-requisitos para o entendimento dos assuntos do livro são bem básicas, necessitando-se apenas de conhecimentos elementares sobre álgebra linear, algoritmos computacionais e cálculo.

A parte I do livro (capítulos de 01 a 10), direcionada mais para aqueles leitores que buscam iniciar ou aprimorar as suas investigações teóricas em redes neurais artificiais, trata das principais arquiteturas que são passíveis de implementação em vários cenários de aplicação.

Na parte II do livro (capítulos de 11 a 20), concebida especificamente para a apresentação de soluções que utilizam redes neurais artificiais para resolver problemas práticos, decorrentes de diferentes áreas do conhecimento, descrevem-se os vários detalhes de implementação que foram levados em consideração para que se alcançassem os resultados relatados. Tais aspectos contribuem para o amadurecimento e aprimoramento, por parte do leitor, das técnicas experimentais que visam especificar a arquitetura de rede neural artificial que seja mais adequada para determinado escopo de aplicação.

## Organização

O livro foi cuidadosamente concebido com a missão de apresentar um texto objetivo, amistoso, acessível e ilustrativo, cuja preocupação principal estava justamente em seu formato didático. Para tais propósitos, a organização em duas partes, e a redação recheada por mais de 200 ilustrações facilitam a transmissão dos conhecimentos aos diferentes perfis de leitores. A bibliografia, constituída por mais de 160 referências, lustreia os temas tratados, enquadrando-se ainda os assuntos frente a um contexto mais atualizado.

A organização do livro foimeticulosamente repartida em duas partes. A primeira (Parte I), dividida em 10 capítulos, abrange os aspectos teóricos referentes às principais arquiteturas de redes neurais artificiais, incluindo *Perceptrons*, *Adaptive linear element (Adaline)*, *Perceptrons multilaminados*, *Radial basis functions (RBF)*, *Hopfield*, *Kohonen*, *Learning vector quantization (LVQ)*, *Counter-propagation* e *Adaptive resonance theory (ART)*. Em cada capítulo foi inserida uma seção de exercícios, em que o leitor poderá gradativamente testar os seus conhecimentos adquiridos no decorrer dos diversos assuntos tratados no livro.

Complementarmente, estes capítulos, que tratam diretamente das diferentes arquiteturas de redes neurais, são ainda contemplados com seções que envolvem projetos práticos, cujos conteúdos auxiliam o leitor nos aspectos experimentais que visam à modelagem de problemas por redes neurais artificiais. Tais atividades também contribuem para a aquisição de conhecimento de

cunho prático, que podem auxiliar na especificação, parametrização e no ajuste das arquiteturas neurais que sejam mais apropriadas para determinados fins.

A segunda parte (II) aborda várias aplicações, abrangendo também diversas áreas do conhecimento, cujas soluções e implementações são advindas do emprego das arquiteturas de redes neurais que foram exploradas na parte I. As diferentes aplicações refletem o potencial de empregabilidade das redes neurais na solução de problemas em engenharia e ciências aplicadas, apontando-se ainda na direção de prospectar ao leitor as diferentes estratégias de modelagem e mapeamento, por intermédio das redes neurais artificiais, de problemas de naturezas diversificadas.

Diversos materiais didáticos relacionados com o livro, incluindo-se figuras, dicas de exercícios e conjuntos de dados para treinamento das redes neurais (em formato de tabelas), estão também disponibilizados na página da internet: <http://laips.sel.eesc.usp.br/>

Em suma, espera-se que a leitura do livro possa ser realizada de forma bem agradável e prazerosa, contribuindo assim para a aquisição de diversos conhecimentos teóricos e práticos envolvidos com a área de redes neurais artificiais.



## **Agradecimentos**

---

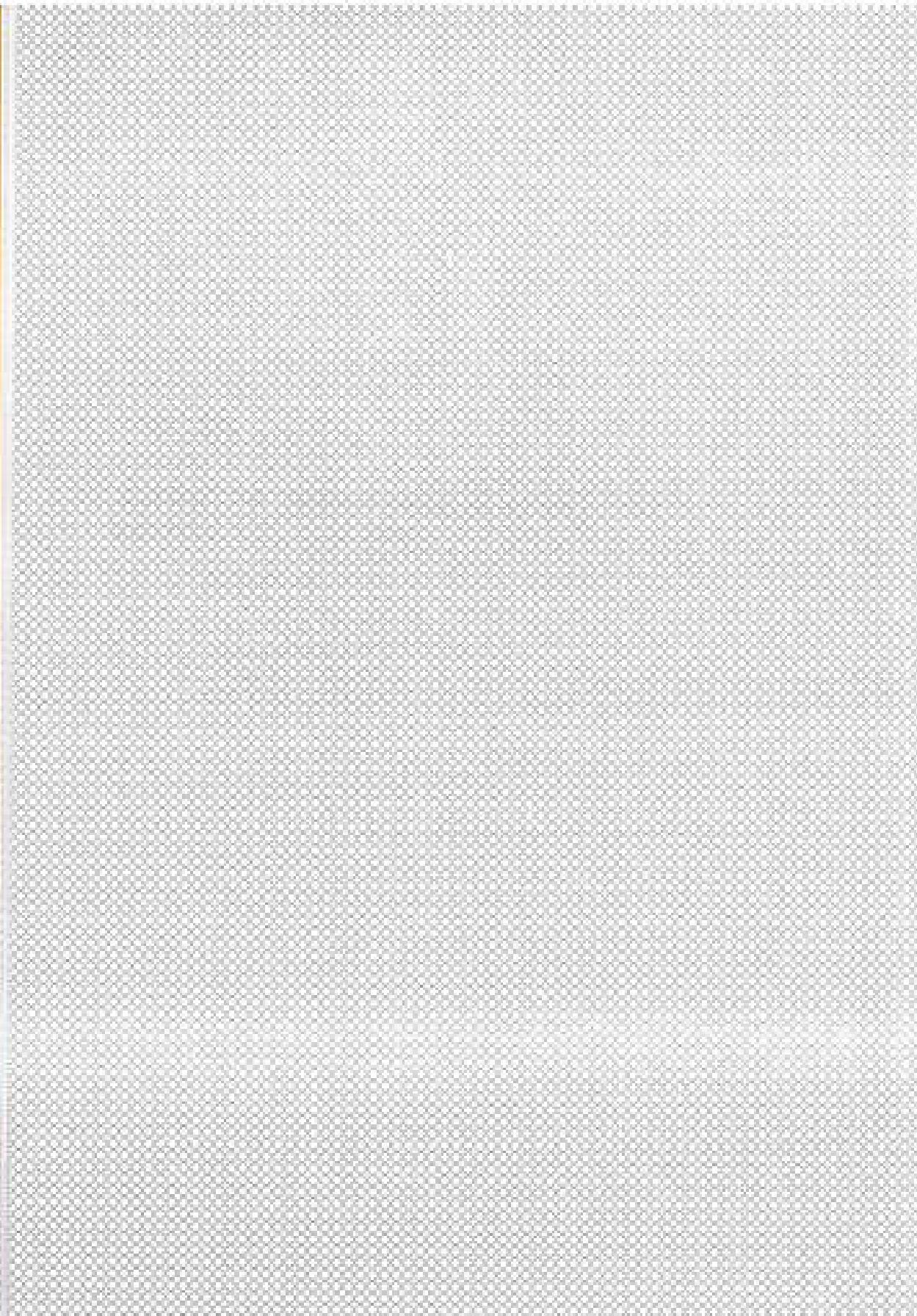
Os autores são imensamente agradecidos aos vários colegas que contribuíram para a realização desta compilação, dispondo de preciosos momentos a fim de auxiliar prontamente nesta importante e nobre empreitada.

Em especial, gostariam de expressar seus agradecimentos aos colegas que colaboraram na formulação da Parte II deste livro: Alexandre C. N. de Oliveira, Anderson da Silva Soares, André L. V. da Silva, Antonio V. Ortega, Débora M. B. S. de Souza, Edison A. Goes, Ednaldo J. Ferreira, Eduardo A. Speranza, Fabiana C. Bertoni, José Alfredo C. Ulson, Juliano C. Miranda, Lucia Valéria R. de Arruda, Marcelo Suetake, Matheus G. Pires, Michelle M. Mendonça, Valmir Ziolkowski, Wagner C. Amaral, Washington L. B. Melo e Wesley F. Usida.



## **Parte I**

### **Arquiteturas de redes neurais artificiais e seus aspectos teóricos**



## Introdução

A ideia de se construir uma máquina ou mecanismo autônomo, que seja dotado de inteligência, se constitui um sonho antigo dos pesquisadores das áreas de ciências e engenharias. Embora os primeiros trabalhos em redes neurais artificiais (RNA) tenham sido publicados há mais de 50 anos, tal tema começou a ser fortemente pesquisado a partir do inicio dos anos 1990, tendo ainda um potencial de pesquisa imenso. As aplicações que envolvem a utilização de sistemas considerados inteligentes são as mais variadas possíveis, por exemplo:

- Avaliação de imagens captadas por satélite;
- Classificação de padrões de escrita e de fala;
- Reconhecimento de faces em visão computacional;
- Controle de trens de grande velocidade;
- Previsão de ações no mercado financeiro;
- Identificação de anomalias em imagens médicas;
- Identificação automática de perfis de crédito para clientes de instituições financeiras;
- Controle de aparelhos eletrônicos e eletrodomésticos, como máquinas de lavar roupa, fornos de microondas, geladeiras, cafeteiras, fritadeiras, filmadoras, etc.

Além dessas aplicações, as potencialidades das redes neurais artificiais permitem a resolução de outros tipos de problemas advindos das mais diferentes áreas do conhecimento, conforme testemunhados em diversos periódicos científicos. É o caso da medicina, em que se utilizam redes neurais artificiais para classificações e previsões de câncer com base no perfil genético do indivíduo [Khan *et alii*, 2001]. Outra aplicação, apresentada em Yan *et alii* (2006), propõe um sistema de apoio à decisão também baseado no uso de redes neurais artificiais para diagnosticar doenças do coração.

Há também registrados trabalhos na área da química, em que se utilizam redes neurais artificiais para obtenção de novos compostos poliméricos [Zhang & Friedrich, 2003]. As redes neurais artificiais são ainda aplicadas em sistemas de controle para tratamento de água, os quais envolvem processos químicos e físicos não-lineares e que são difíceis de serem mapeados por controles convencionais [Zhang & Stanley, 1999].

Na área da biologia é possível encontrar aplicações utilizando redes neurais artificiais com o objetivo de se identificar espécies de morcegos a partir de seus sinais de ecolocalização (biosonar) emitidos durante os voos [Parsons & Jones, 2000]. Outra abordagem neural para classificação de espécies de ratos, usando-se sons produzidos pelos mesmos, foi desenvolvida em Tian & Shang (2006).

Em relação à área de finanças e economia, há também problemas de difícil tratamento, principalmente devido ao comportamento não-linear destes sistemas. As redes neurais artificiais são largamente empregadas em tais situações graças à sua capacidade de tratar as não-linearidades intrínsecas [Coakley & Brown, 2000].

O ramo da ecologia também se beneficia da capacidade das redes neurais artificiais em extrair informações, sendo possível realizar análises entre a influência do clima anual frente à taxa de crescimento de árvores [Zhang *et alii*, 2000].

A habilidade das redes neurais artificiais em classificar padrões pode ser observada até mesmo no ramo da etologia, no qual se busca a diferenciação entre as diversas expressões faciais que caracterizam os sentimentos humanos [Dailey & Cottrell, 2002]. Outra aplicação relacionada à classificação de padrões é dissipada em Fernandes (2009), em que redes neurais artificiais são usadas para classificar fontes de correntes harmônicas em sistemas de distribuição de energia elétrica.

No ramo da farmácia é possível encontrar o emprego de redes neurais artificiais para auxiliar na elaboração de fórmulas, indicando se o medicamento fabricado deverá ser realizado por intermédio de micro-emulsão ou dispersão-sólida [Mendyk & Jachowicz, 2007].

Na acústica também se encontram pesquisas que utilizam redes neurais artificiais para determinar a impedância acústica de um ambiente, fator este importantíssimo para o projeto de salas de cinema e ambientes sensíveis a ruídos externos [Itoo *et alii*, 2007].

A profundidade em que agentes poluentes tendem a penetrar no solo e contaminar lençóis freáticos pode ser, outrossim, estimada por intermédio de redes neurais artificiais, fornecendo subsídios para a implementação de medidas de contenção [Tabach *et alii*, 2007].

A indústria de alimentos também tem sido beneficiada com a utilização de redes neurais artificiais, tais como aquela usada na classificação de diversas variedades existentes de chá [He *et alii*, 2007]. Outro exemplo de aplicação é o trabalho desenvolvido em Silva (2007), em que uma abordagem neural foi implementada para processamento de sinal de ressonância magnética nuclear visando à classificação de amostras de carnes bovinas, possibilitando então a identificação de sexo e raça dos animais. Já em Nazário (2007) foram utilizadas redes neurais artificiais associadas às técnicas de ultra-som objetivando a caracterização de leite líquido em termos de gordura e adulteração por adição de água.

No setor automotivo e aeroespacial encontram-se aplicações de redes neurais artificiais para auxiliar no mapeamento de processos que envolvem estimativas de variáveis de controle e parâmetros de projeto. Como alguns destes exemplos, em Cho *et alii* (2006) foram propostos esquemas de modelagem e estratégias de controle para veículos aéreos não-tripulados. Já em Richter (2009) foi projetada uma arquitetura neural visando sensoriamento virtual de oxigênio em veículos bicompostos. Em Vicente *et alii* (2007) foi proposto um controlador neural de marcha lenta para motores de combustão interna. Outra aplicação interessante do ramo automotivo foi formulada em Ortega & Silva (2008), em que as redes neurais artificiais são responsáveis pela otimização de projetos de *brake-light* automotivos constituídos por diodos emissores de luz.

As redes neurais artificiais estão inseridas dentro da área conhecida como sistemas inteligentes (conexionistas) ou inteligência computacional

[Jang *et alii*, 1997; Zadeh, 1992]. Além das redes neurais artificiais, a área de sistemas inteligentes é composta de várias outras ferramentas, tais como os sistemas *fuzzy* [Pedrycz & Gomide, 2007; Buckley & Siler, 2004; Ross, 2004], computação evolutiva [Michalewicz, 1999; Dasgupta & Michalewicz, 1997; Goldberg, 1989], inteligência coletiva (*swarm intelligence*) [Kennedy & Eberhart, 2001], sistemas imunológicos artificiais [Dasgupta, 2006; Castro & Timmis, 2002] e agentes inteligentes [D'Inverno & Luck, 2004].

Complementarmente, considerando também o ramo do entretenimento contemporâneo, a indústria envolvida com as artes cinematográficas também tem explorado o assunto por meio de vários filmes de ficção científica que acabam abordando a utilização de sistemas inteligentes em máquinas e robôs.

Mais especificamente, as características mais atrativas das redes neurais artificiais, fazendo destas ferramentas poderosas para aplicação em diversos tipos de problemas, consistem de suas elevadas habilidades em mapear sistemas não-lineares, aprendendo os comportamentos envolvidos a partir de informações (medidas, amostras ou padrões) obtidas.

## 1.1 – Conceitos iniciais

Redes neurais artificiais são modelos computacionais inspirados no sistema nervoso de seres vivos. Possuem a capacidade de aquisição e manutenção do conhecimento (baseado em informações) e podem ser definidas como um conjunto de unidades de processamento, caracterizadas por neurônios artificiais, que são interligados por um grande número de interconexões (*sinapses artificiais*), sendo as mesmas representadas aqui por vetores/matrizes de pesos sinápticos.

### 1.1.1 – Características principais

As características mais relevantes envolvidas com aplicação de redes neurais artificiais são:

- a) Adaptação por experiência: as adaptações dos parâmetros internos da rede, tipicamente seus pesos sinápticos, são ajustadas a partir da apresentação sucessiva de exemplos (padrões, amostras, medidas) rela-

cionados ao comportamento do processo, possibilitando a aquisição do conhecimento por experimentação;

b) Capacidade de aprendizado: por intermédio da aplicação de um método de treinamento, a rede consegue extrair o relacionamento existente entre as diversas variáveis que compõem a aplicação;

c) Habilidade de generalização: após o processo de treinamento da rede, essa é capaz de generalizar o conhecimento adquirido, possibilitando estimar soluções que eram até então desconhecidas;

d) Organização de dados: baseada em características intrínsecas envolvendo determinado conjunto de informações a respeito de um processo, a rede é capaz de realizar a sua organização interna visando possibilitar o agrupamento de padrões que apresentam particularidades em comum;

e) Tolerância a falhas: devido ao elevado nível de interconexões entre os neurônios artificiais, a rede neural torna-se um sistema tolerante a falhas quando parte de sua estrutura interna é sensivelmente corrompida;

f) Armazenamento distribuído: o conhecimento a respeito do comportamento de determinado processo dentro de uma arquitetura neural é realizada de forma distribuída entre as diversas sinapses de seus neurônios artificiais, permitindo então um incremento da robustez da arquitetura frente a eventuais neurônios que se tornaram inoperantes;

g) Facilidade de prototipagem: a implementação da maioria das arquiteturas neurais pode ser facilmente, dependendo da especificidade da aplicação, promovida em hardware ou em software, pois, após o processo de treinamento, os seus resultados são normalmente obtidos por algumas operações matemáticas elementares.

### 1.1.2 – Resumo histórico

A primeira publicação relacionada à neurocomputação data de 1943, por meio do artigo elaborado por McCulloch & Pitts (1943). Neste trabalho, os autores realizaram o primeiro modelamento matemático inspirado no neurônio biológico, resultando-se assim na primeira concepção de neurônio artificial.

Em 1949, o primeiro método de treinamento para redes neurais artificiais foi proposto, que se denominou de regra de aprendizado de Hebb, sendo esta baseada em hipóteses e observações de caráter neurofisiológico [Hebb, 1949].

Diversos outros pesquisadores continuaram o trabalho de desenvolvimento de modelos matemáticos fundamentados no neurônio biológico, gerando uma série de topologias (estruturas) e de algoritmos de aprendizado. Entre as linhas de pesquisa que surgiram destaca-se o trabalho de Frank Rosenblatt que, no período compreendido entre 1957 e 1958, desenvolveu o primeiro neurocomputador, denominado *Mark I – Perceptron*, idealizando o modelo básico do *Perceptron* [Rosenblatt, 1958].

O modelo do *Perceptron* despertou interesse devido à sua capacidade em reconhecer padrões simples. Widrow & Hoff (1960) desenvolveram um tipo de rede denominada *Adaline*, que é a abreviatura de *ADaptive LINear Element*. Posteriormente, propôs-se o *Madaline*, a *Adaline* multipla, que é uma rede cujo aprendizado é fundamentado na chamada regra Delta, também conhecida como algoritmo de aprendizado *LMS (least mean square)*.

Em seguida, após esses trabalhos pioneiros, muitos pesquisadores da época ficaram incentivados a realizar pesquisas relacionadas com esta frente de investigação. Porém, em 1969, a neurocomputação sofreu um revés com a publicação do clássico livro *Perceptrons – an introduction to computational geometry* por Minsky & Papert (1969). Os autores demonstraram de forma enfática a limitação de redes neurais artificiais, constituídas de apenas uma única camada, como o *Perceptron* e o *Adaline*, em aprender o relacionamento entre as entradas e saídas de funções lógicas bem simples como o  $X_{\text{OR}}$  (ou-exclusivo). De forma mais específica, nesta publicação houve a demonstração da impossibilidade de as redes realizarem a correta classificação de padrões para classes não linearmente separáveis.

A partir de reflexos desta publicação, teve-se então um período em que pouquíssimas pesquisas eram desenvolvidas, cujos destaques foram a derivação de algoritmos de predição utilizando gradiente reverso [Werbos, 1974], a implementação da rede *ART (adaptive resonance theory)* por Grossberg (1980), a formulação de mapas auto-organizáveis por Kohonen (1982), e a proposição por Hopfield (1982) de redes recorrentes baseadas em funções de energia, sendo que esta última fez com que a área de redes neurais artificiais retomasse o destaque que possuia antes de 1969.

Somente no final dos anos 1980, ainda com o impulso inicial dos trabalhos citados no parágrafo anterior, é que os pesquisadores voltaram a ter significativo interesse nesta área. A retomada definitiva das pesquisas se deve a diversos fatores, tais como o desenvolvimento de computadores com maior capacidade de processamento, a criação de algoritmos de otimização mais eficientes e robustos e, finalmente, as novas descobertas sobre o sistema nervoso biológico. Um dos principais realces naquele período foi a publicação do livro de Rumelhart, Hinton e Williams, intitulado *Parallel distributed processing* [Rumelhart *et al.*, 1986], em que os autores desenvolveram um algoritmo que permitia ajustar os pesos em uma rede com mais de uma camada, solucionando-se inclusive o antigo problema de aprendizado dos padrões da função lógica  $X_{eq}$  (ou-exclusivo). A proposição de tal algoritmo, denominado *backpropagation*, reascendeu e motivou definitivamente as pesquisas em redes neurais artificiais.

Mais recentemente, além de inúmeras aplicações práticas em diferentes ramos do conhecimento, dezenas de novas outras contribuições têm permitido alavancar os desenvolvimentos teóricos associados às redes neurais artificiais. Em especial se podem destacar a proposição de algoritmos de aprendizado baseados no método de Levenberg-Marquardt, permitindo-se incrementar a eficiência do treinamento de redes neurais artificiais em diversas aplicações [Hagan & Menhaj, 1994]; as redes neurais artificiais baseadas em máquinas de vetores suporte (*support vector machines – SVM*), que podem também ser utilizadas em classificação de padrões e regressão linear [Vapnik, 1998]; a implementação de circuitos integrados neurais com diversas configurações de tipologia [Bciu *et al.*, 2003], etc.

Uma descrição extensivamente detalhada sobre diversos outros fatos históricos envolvidos com o percurso evolutivo das redes neurais artificiais, desde os seus primórdios, pode ser examinada em Haykin (1999).

### 1.1.3 – Potenciais áreas de aplicações

As redes neurais artificiais podem ser empregadas em diversos problemas relacionados às engenharias e ciências. As potenciais áreas de aplicabilidade podem ser enquadradas conforme se segue:

- a) Aproximador universal de funções: o objetivo consiste em mapear o relacionamento funcional entre as variáveis (tipicamente reais) de um

sistema a partir de um conjunto conhecido de seus valores representativos. As aplicações são as mais diversas possíveis, sendo que envolvem normalmente o mapeamento de processos cuja modelagem por técnicas convencionais são de difícil obtenção;

b) Controle de processos: o objetivo consiste em identificar ações de controle que permitam o alcance dos requisitos de qualidade, eficiência e segurança do processo. Entre as várias aplicações disponíveis destacam-se os controles empregados em robótica, aeronaves, elevadores, eletrodomésticos, satélites, etc;

c) Reconhecimento/classificação de padrões: o objetivo deste tipo de aplicação consiste de associar um padrão de entrada (amostra) para uma das classes previamente definidas, como acontece em reconhecimento de imagens, voz, escrita, etc. Neste caso, o problema a ser tratado possui um conjunto discreto e conhecido das possíveis saídas desejadas;

d) Agrupamento de dados (clusterização): o objetivo nesta circunstância consiste da identificação e detecção de similaridades e particularidades entre os diversos padrões de entrada a fim de possibilitar seu agrupamento. Como exemplo, pode-se citar os problemas que envolvem identificação automática de classes e garimpagem de dados;

e) Sistemas de previsão: o objetivo consiste em estimar valores futuros de um processo levando-se em consideração diversas medidas prévias observadas em seu domínio. Entre as aplicações disponíveis enquadram-se a previsão de séries temporais, previsões de mercados financeiros, previsões climáticas, etc;

f) Otimização de sistemas: o alvo consiste em minimizar ou maximizar uma função custo (objetivo) obedecendo também eventuais restrições que são impostas para o correto mapeamento do problema. Entre as classes de otimização que podem ser tratadas por redes neurais artificiais destacam-se problemas de otimização restrita, programação dinâmica e otimização combinatorial;

g) Memórias associativas: o objetivo consiste em recuperar padrões corretos mesmo se os seus elementos constituintes forem apresentados de forma incerta ou imprecisa. Entre as aplicações citam-se aquelas relacionadas ao processamento de imagens, à transmissão de sinais, identificação de caracteres manuscritos, etc.

## 1.2 – Neurônio biológico

O processamento de informações no cérebro humano é regido por elementos processadores biológicos que operam em paralelo, tendo como objetivo a produção de ações apropriadas para cada uma de suas funcionalidades, tais como o pensar e o memorizar.

A célula elementar do sistema nervoso cerebral é o neurônio e seu papel se resume a conduzir impulsos (estímulos elétricos advindos de reações físico-químicas) sob determinadas condições de operação. Tal elemento biológico pode ser dividido em três partes principais, isto é, nos dendritos, no corpo celular (também conhecido como soma) e no axônio.

Os dendritos são constituídos por vários finos prolongamentos que formam a árvore dendritica (figura 1.1). A principal função dos dendritos dos neurônios consiste de captar, de forma contínua, os estímulos vindos de diversos outros neurônios (conectores) ou do próprio meio externo onde os mesmos podem estar em contato (neurônios sensitivos).

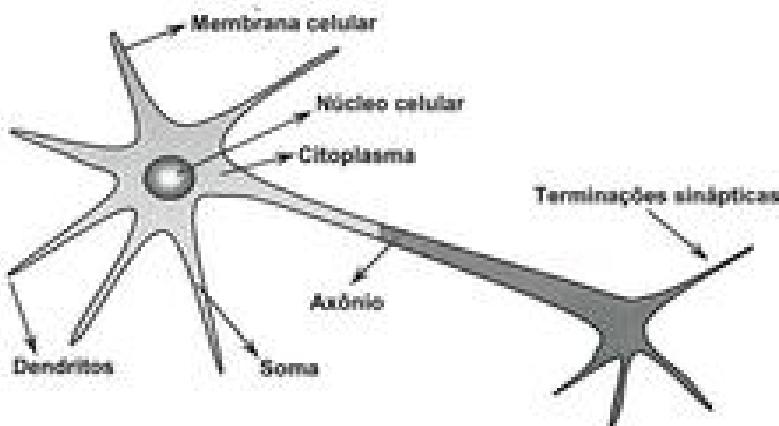


Figura 1.1 – Neurônio biológico

O corpo celular é incumbido de processar todas as informações advindas dos dendritos a fim de produzir um potencial de ativação que indicará se o neurônio poderá disparar um impulso elétrico ao longo de seu axônio. É também no corpo celular que se encontram as principais organelas citoplasmáticas (núcleo, mitocôndria, centriolo, lisossomo, etc.) do neurônio.

O axônio é constituído por um único prolongamento, cuja missão é conduzir os impulsos elétricos para outros neurônios conectores ou para aqueles que se conectam diretamente com o tecido muscular (neurônios efetuadores). A sua terminação é também constituída de ramificações denominadas terminações sinápticas (figura 1.1).

As sinapses se configuram como as conexões que viabilizam a transferência de impulsos elétricos do axônio de um neurônio para os dendritos de outros, conforme ilustração mostrada na figura 1.2. Deve-se mencionar que inexiste contato físico entre os neurônios na junção (fenda) sináptica, sendo que elementos neurotransmissores liberados são os responsáveis por ponderar a transmissão de impulsos elétricos de um neurônio para o outro. De fato, a funcionalidade de um neurônio fica a mercê do comportamento dessas ponderações sinápticas que são dinâmicas e dependentes da química cerebral [Hodgkin & Huxley, 1952].

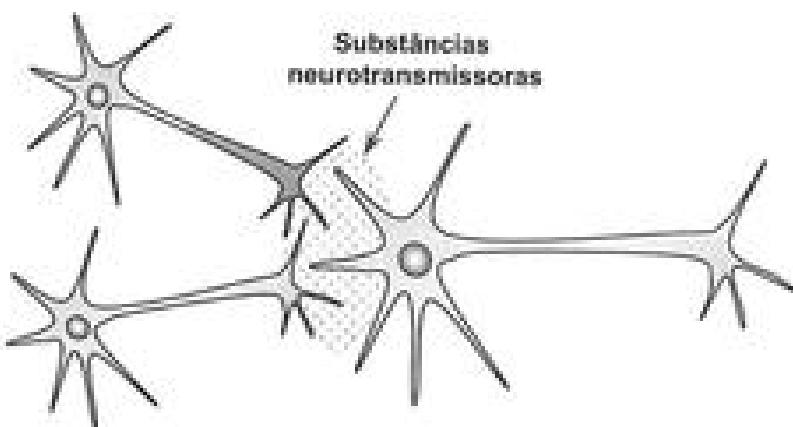


Figura 1.2 – Ilustração de conexões sinápticas entre neurônios

Em suma, embora as atividades relacionadas aos neurônios biológicos parecerem inicialmente bem simples, tais elementos, atuando em conjunto, são os principais responsáveis por todos os processamentos executados e gerenciados pelo cérebro humano. Estima-se que esta rede neural biológica, com características bem excêntricas, seja constituída por cerca de 100 bilhões ( $10^{10}$ ) de neurônios. Cada um destes é interligado por conexões sinápticas (viabilizadas por mais de cinquenta substâncias neurotransmis-

sóras) em média a outros 6.000 neurônios, perfazendo-se então um total de 600 trilhões de sinapses [Shepherd, 1990]. A tabela 1.1 fornece algumas das características físicas relacionadas ao cérebro humano (indivíduo adulto) e aos seus componentes.

**Tabela 1.1 – Características físicas do cérebro humano e seus componentes (em valores médios estimados)**

Característica	Dimensão física
Massa do cérebro	1,5 kg
Energia consumida pelo cérebro	20%
Comprimento do neurônio	100 $\mu$ m
Potencial de repouso da membrana	-70 mV
Potencial de ativação da membrana	-55 mV
Potencial de ação (máximo)	35 mV

Por meio das informações contidas na tabela 1.1, observa-se que o potencial de ação da membrana neural, quando em repouso (polarizada), assume valor negativo, isto é, há uma concentração maior de íons negativos internamente à membrana em relação ao seu exterior.

No momento em que a célula nervosa é estimulada (despolarizada) além de um limiar de ativação (-55 mV), que é ocasionado pela variação de concentração interna de íons de sódio ( $Na^+$ ) e potássio ( $K^+$ ), há então o disparo de um impulso elétrico que será propagado ao longo de todo o seu axônio, cuja amplitude máxima alcançará o valor de 35 mV [Kandel *et alii*, 2000]. A figura 1.3 mostra as diversas fases envolvidas com as variações do potencial de ação dentro do neurônio no decorrer de sua excitação.

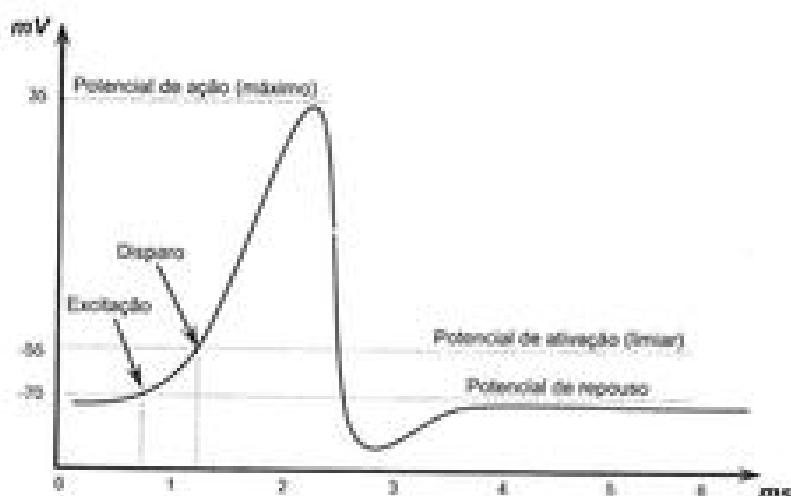


Figura 1.3 – Etapas de variação do potencial de ação do neurônio.

Deve-se ressaltar que, independentemente dos tipos de neurônio (receptor, associativo ou motor), a amplitude de 35 mV, correspondente ao máximo valor de um potencial de ação, é fixa e tende a ser rigorosamente atingida por todos eles quando estimulados, embora a duração deste sinal seja um parâmetro variante. Ao término do processo de excitação, como consequência, haverá a repolarização da membrana, isto é, o potencial de ação do neurônio retornará novamente ao seu patamar de potencial de repouso (-70 mV), conforme ilustrado na figura 1.3.

A Figura 1.4 mostra um registro fotográfico envolvendo neurônios do córtex cerebral.



Figura 1.4 – Registro fotográfico de neurônios do córtex cerebral.

A disposição espacial entre esses neurônios também permite visualizar algumas de suas diversas interligações sinápticas.

### 1.3 – Neurônio artificial

A estrutura das redes neurais artificiais foi desenvolvida a partir de modelos conhecidos de sistemas nervosos biológicos e do próprio cérebro humano. Os elementos computacionais ou unidades processadoras, denominadas neurônios artificiais, são modelos bem simplificados dos neurônios biológicos.

Tais modelos foram inspirados a partir da análise da geração e propagação de impulsos elétricos pela membrana celular dos neurônios [Hodgkin & Huxley, 1952].

Os neurônios artificiais utilizados nos modelos de redes neurais artificiais são não-lineares, fornecem saídas tipicamente contínuas, e realizam funções simples, como coletar os sinais existentes em suas entradas, agregá-los de acordo com sua função operacional e produzir uma resposta, levando em consideração sua função de ativação inherent.

O modelo de neurônio mais simples e que engloba as principais características de uma rede neural biológica, isto é, paralelismo e alta conectividade, foi proposto por McCulloch & Pitts (1943), sendo ainda o modelo mais utilizado nas diferentes arquiteturas de redes neurais artificiais.

Nessa representação, cada neurônio da rede pode ser implementado conforme mostra a figura 1.5. Os diversos sinais de entrada advindos do meio externo (aplicação) são espelhados pelo conjunto  $\{x_1, x_2, x_3, \dots, x_n\}$ , que são análogos aos impulsos elétricos exteros captados pelos dendritos no neurônio biológico.

As ponderações exercidas pelas junções sinápticas do modelo biológico são representadas no neurônio artificial pelo conjunto de pesos sinápticos  $\{w_1, w_2, \dots, w_n\}$ . De forma análoga, a relevância de cada uma das entradas ( $x_i$ ) do neurônio é então executada por meio de suas multiplicações pelos respectivos pesos sinápticos ( $w_i$ ), ponderando-se, portanto, todas as informações externas que chegam ao neurônio. Assim, torna-se possível verificar que a saída do corpo celular artificial, denotado por  $u$ , é a soma ponderada de suas entradas.

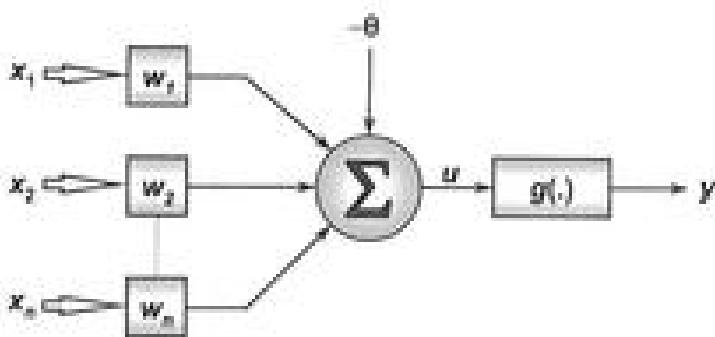


Figura 1.5 – Neurônio artificial

Assim, considerando a figura 1.5, verifica-se que o neurônio artificial é constituído de sete elementos básicos, ou seja:

- Sinais de entrada  $\{x_i, x_j, \dots, x_k\}$

São os sinais ou medidas advindas do meio externo e que representam os valores assumidos pelas variáveis de uma aplicação específica. Os sinal de entrada são usualmente normalizados visando incrementar a eficiência computacional dos algoritmos de aprendizagem;

- Pesos sinápticos  $\{w_i, w_j, \dots, w_n\}$

São os valores que servirão para ponderar cada uma das variáveis de entrada da rede, permitindo-se quantificar as suas relevâncias em relação à funcionalidade do respectivo neurônio;

- Combinador linear  $\{\Sigma\}$

Sua função é agregar todos os sinal de entrada que foram ponderados pelos respectivos pesos sinápticos a fim de produzir um valor de potencial de ativação;

- Limiar de ativação  $\{\theta\}$

É uma variável que especifica qual será o patamar apropriado para que o resultado produzido pelo combinador linear possa gerar um valor de disparo em direção à saída do neurônio;

- Potencial de ativação  $\{u\}$

É o resultado produzido pela diferença do valor produzido entre o combinador linear e o limiar de ativação. Se tal valor é positivo, ou seja, se  $u \geq 0$

então o neurônio produz um potencial excitatório; caso contrário, o potencial será inibitório;

#### f) Função de ativação ( $g$ )

Seu objetivo é limitar a saída do neurônio dentro de um intervalo de valores razoáveis a serem assumidos pela sua própria imagem funcional;

#### g) Sinal de saída ( $y$ )

Consiste do valor final produzido pelo neurônio em relação a um determinado conjunto de sinais de entrada, podendo ser também utilizado por outros neurônios que estão sequencialmente interligados.

As duas expressões seguintes sintetizam o resultado produzido pelo neurônio artificial proposto por McCulloch e Pitts, ou seja:

$$u = \sum_{i=1}^n w_i \cdot x_i - \theta \quad (1.1)$$

$$y = g(u) \quad (1.2)$$

Assim, pode-se resumir o funcionamento de um neurônio artificial por meio dos seguintes passos:

- Apresentação de um conjunto de valores que representam as variáveis de entrada do neurônio;
- Multiplicação de cada entrada do neurônio pelo seu respectivo peso sináptico;
- Obtenção do potencial de ativação produzido pela soma ponderada dos sinais de entrada, subtraindo-se o limiar de ativação;
- Aplicação de uma função de ativação apropriada, tendo-se como objetivo limitar a saída do neurônio;
- Compilação da saída a partir da aplicação da função de ativação neural em relação ao seu potencial de ativação.

As funções de ativação podem ser divididas em dois grupos principais, isto é, funções parcialmente diferenciáveis e funções totalmente diferenciáveis, considerando-se para tanto todo o domínio de definição das mesmas.

### 1.3.1 – Funções de ativação parcialmente diferenciáveis.

As funções de ativação parcialmente diferenciáveis são aquelas que possuem pontos cujas derivadas de primeira ordem são inexistentes. As três principais funções desse grupo são: função degrau, função degrau bipolar e função rampa simétrica.

#### a) Função degrau (*heaviside/band limiter*)

O resultado produzido pela aplicação da função degrau assumirá valores unitários positivos quando o potencial de ativação do neurônio for maior ou igual a zero; caso contrário, o resultado assumirá valores nulos, ou seja:

$$g(u) = \begin{cases} 1, & \text{se } u \geq 0 \\ 0, & \text{se } u < 0 \end{cases} \quad (1.3)$$

A representação gráfica desta função é ilustrada na figura 1.6.

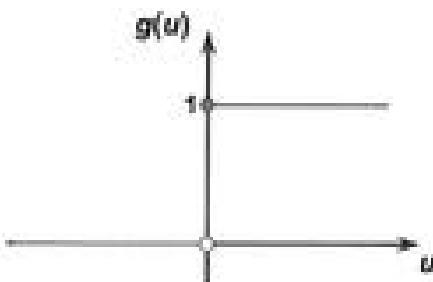


Figura 1.6 – Função de ativação degrau

#### b) Função degrau bipolar ou função sinal (*symmetric band limiter*)

O resultado produzido pela aplicação desta assumirá valores unitários positivos quando o potencial de ativação do neurônio for maior que zero; valor nulo quando o potencial for também nulo; e valores unitários negativos quando o potencial for menor que zero. Em notação matemática, tem-se:

$$g(u) = \begin{cases} 1, & \text{se } u > 0 \\ 0, & \text{se } u = 0 \\ -1, & \text{se } u < 0 \end{cases} \quad (1.4)$$

A representação gráfica desta função é ilustrada na figura 1.7.

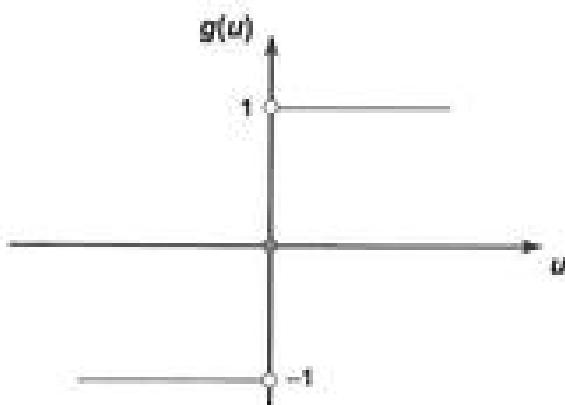


Figura 1.7 – Função de ativação degrau bipolar

Em problemas envolvendo classificação de padrões, a função degrau bipolar pode ser também aproximada pela seguinte expressão:

$$g(u) = \begin{cases} 1, & \text{se } u \geq 0 \\ -1, & \text{se } u < 0 \end{cases} \quad (1.5)$$

Outra alternativa nesta circunstância é manter a saída do neurônio inalterada, ou seja:

$$g(u) = \begin{cases} 1, & \text{se } u > 0 \\ \text{saída anterior, se } u = 0 \\ -1, & \text{se } u < 0 \end{cases} \quad (1.6)$$

### c) Função rampa simétrica

Os valores retornados são iguais aos próprios valores dos potenciais de ativação quando estes estão definidos no intervalo  $[-a, a]$ , restringindo-se aos valores limites em caso contrário. Em notação matemática, tem-se:

$$g(u) = \begin{cases} a, & \text{se } u > a \\ u, & \text{se } -a \leq u \leq a \\ -a, & \text{se } u < -a \end{cases} \quad (1.7)$$

A representação gráfica desta função é ilustrada na figura 1.8.

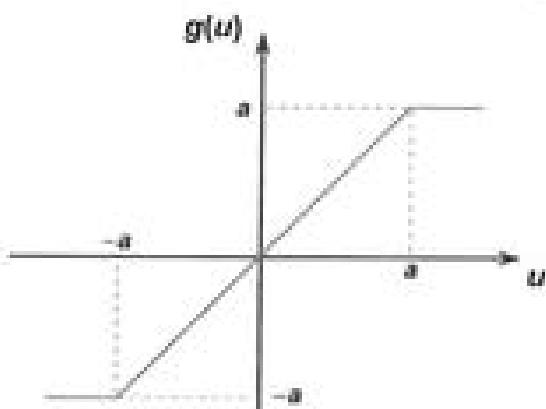


Figura 1.8 – Função de ativação rampa simétrica.

### 1.3.2 – Funções de ativação totalmente diferenciáveis

As funções de ativação totalmente diferenciáveis são aquelas cujas derivadas de primeira ordem existem e são conhecidas em todos os pontos de seu domínio de definição. As quatro principais funções pertencentes a este grupo, e que podem ser empregadas em redes neurais artificiais são a função logística, a tangente hiperbólica, a gaussiana e a função linear.

#### a) Função logística

O resultado de saída produzido pela aplicação da função logística assumirá sempre valores reais entre zero e um, tendo-se sua expressão matemática dada por:

$$g(u) = \frac{1}{1 + e^{-\beta u}} \quad (1.8)$$

onde  $\beta$  é uma constante real associada ao nível de inclinação da função logística frente ao seu ponto de inflexão. A representação gráfica desta função é ilustrada na figura 1.9.

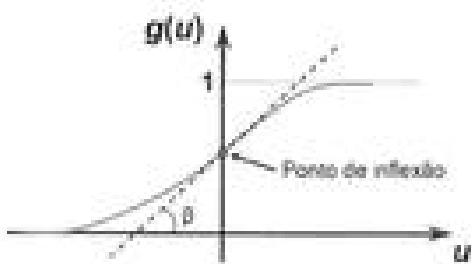


Figura 1.9 – Função de ativação logística

A figura 1.10 mostra o comportamento da função de ativação logística em relação à variação do parâmetro de inclinação  $\beta$ .

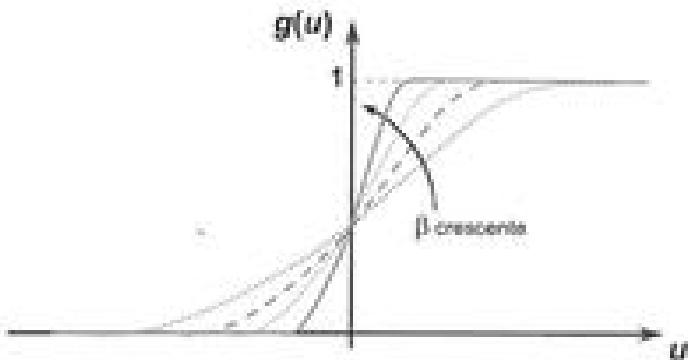


Figura 1.10 – Influência do parâmetro  $\beta$  na função de ativação logística

A partir da análise da figura 1.10 conclui-se que o formato geométrico da função de ativação logística tenderá a ser similar àquele da função degrau (figura 1.6) quando o parâmetro  $\beta$  for muito elevado (tender ao infinito). Entretanto, em contraste com a função degrau, a função logística é totalmente diferenciável em todo o seu domínio de definição.

### b) Função tangente hiperbólica

O resultado de saída, diferentemente da função logística, sempre assumirá valores reais entre -1 e 1, cuja expressão matemática é definida por:

$$g(u) = \frac{1 - e^{-\beta u}}{1 + e^{-\beta u}} \quad (1.9)$$

onde  $\beta$  está também associado ao nível de inclinação da função tangente hiperbólica em relação ao seu ponto de inflexão. A representação gráfica desta função é ilustrada na figura 1.11.

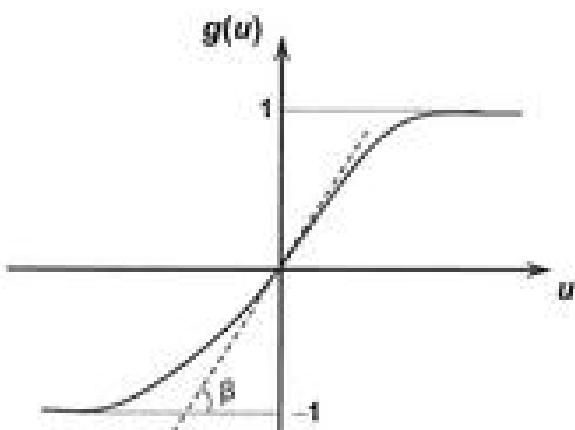


Figura 1.11 – Função de ativação tangente hiperbólica.

A figura 1.12 ilustra também o gráfico da função tangente hiperbólica para diversos valores atribuídos ao parâmetro  $\beta$ .

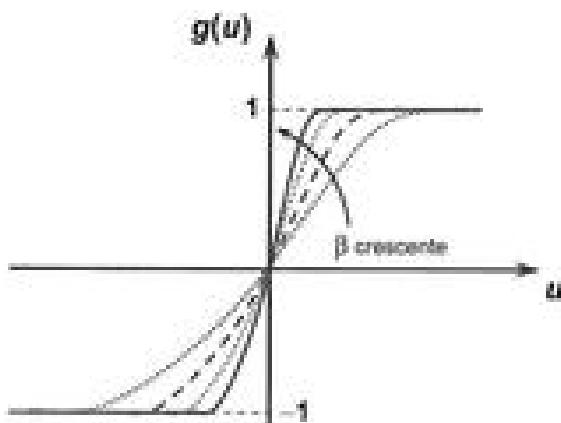


Figura 1.12 – Influência do parâmetro  $\beta$  na função de ativação tangente hiperbólica

Conforme observado na figura 1.12, assim como ocorre para a função logística, quanto maior for o valor do parâmetro  $\beta$ , então mais inclinada estará a função tangente hiperbólica, sendo que se aproximará da função degrau bipolar (figura 1.7) quando o valor de  $\beta$  for também muito elevado.

Tanto a função logística como a função tangente hiperbólica pertencem à família das funções denominadas de sigmoidais.

### c) Função gaussiana

Em relação à utilização da função de ativação gaussiana, a saída do neurônio produzirá resultados iguais para aqueles valores de potencial de ativação ( $u$ ) que estejam posicionados a uma mesma distância de seu centro (média), sendo que a curva é simétrica em relação a este. A função gaussiana é dada por:

$$g(u) = e^{-\frac{(u-c)^2}{2\sigma^2}} \quad (1.10)$$

onde  $c$  é um parâmetro que define o centro da função gaussiana e  $\sigma$  denota o desvio padrão associado à mesma, isto é, o quanto espalhada (dispersada) está a curva em relação ao seu centro. A representação gráfica desta função é ilustrada na figura 1.13.

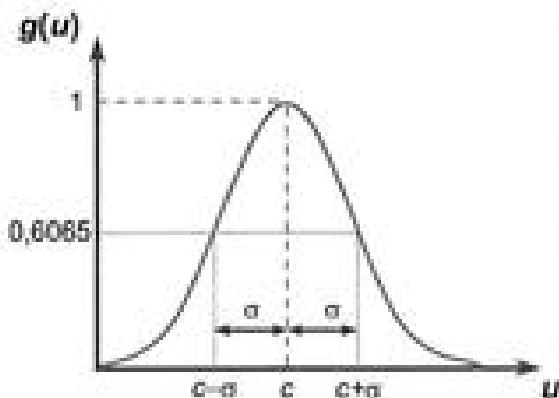


Figura 1.13 – Função de ativação gaussiana.

Contata-se nesta figura que o valor do desvio padrão ( $\sigma$ ) está diretamente associado com o ponto de inflexão da função gaussiana, sendo que  $\sigma^2$  indica a sua respectiva variância.

d) Função linear

A função de ativação linear ou função identidade produz resultados de saída idênticos aos valores do potencial de ativação  $\{u\}$ , tendo sua expressão matemática definida por:

$$g(u) = u \quad (1.11)$$

A representação gráfica desta função é ilustrada na figura 1.14.

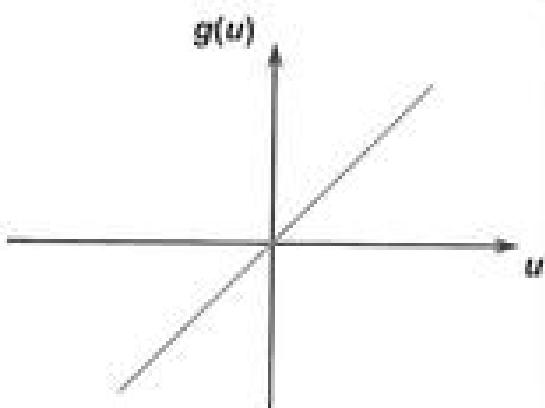


Figura 1.14 – Função de ativação linear

Uma das aplicabilidades da função de ativação linear está na utilização de redes neurais artificiais como aproximators universais de funções, visando-se mapear o comportamento entre as variáveis de entrada e saída de processos, conforme será descrito na seção 5.4.

## 1.4 – Parâmetros de desempenho

Para tecer algumas considerações sobre o funcionamento do neurônio artificial e neurônio biológico, a tabela 1.2 compara alguns parâmetros relativos ao desempenho de ambos.

Tabela 1.2 – Aspectos comparativos entre o neurônio artificial e o biológico

Parâmetro	Neurônio artificial	Neurônio biológico
Eficiência energética (operação/segundo)	$10^{-4}$ J	$10^{-14}$ J
Tempo de processamento (operação/neurônio)	$10^{-9}$ s (clock na ordem de GHz)	$10^{-3}$ s
Mecanismo de processamento	Tipicamente sequencial	Tipicamente paralelo

Pode-se observar que o tempo de processamento do neurônio artificial é muito superior ao neurônio biológico. Mas, o processamento cerebral é infinitas vezes mais rápido que uma rede neural artificial na maioria dos casos, pois os neurônios da rede neural biológica trabalham com alto grau de paralelismo entre si, ao passo que para os neurônios artificiais este nível de paralelismo é bem limitado, pois a maioria dos computadores é constituída de máquinas tipicamente sequenciais [Faggin, 1991; Haykin, 1999].

O parâmetro de velocidade de uma rede neural artificial está basicamente relacionado com o número de operações por segundo dos computadores. Considerando um clock da ordem de gigahertz, então um período de processamento dos neurônios artificiais fica na grandeza de nanosegundos.

## 1.5 – Exercícios

- 1) Explique o funcionamento do neurônio artificial.
- 2) Descreva os objetivos principais das funções de ativação.
- 3) Faça uma analogia entre os elementos constituintes do neurônio artificial e do neurônio biológico.
- 4) Discorra sobre a importância envolvendo o limiar de ativação.

- 5) Em relação às características das redes neurais artificiais, explique em que consiste a adaptação por experiência e a capacidade de generalização.
- 6) Discorra sobre as principais características matemáticas que são verificadas nas funções de ativação logística e tangente hiperbólica.
- 7) Obtenha as expressões analíticas das derivadas de primeira ordem da função de ativação logística e tangente hiperbólica.
- 8) Para um problema específico, há a possibilidade de utilizar como função de ativação tanto a função logística com a tangente hiperbólica. Em termos de implementação em hardware, discorra quais seriam os eventuais aspectos relevantes para a seleção de uma destas.
- 9) Considerando que as operações individuais nos neurônios artificiais são realizadas mais rapidamente em comparação com os neurônios biológicos, explique por que diversas atividades executadas pelo cérebro humano produzem resultados mais rapidamente que um microcomputador.
- 10) Quais os principais tipos de problemas em que as redes neurais artificiais são aplicadas.



Alan Lloyd Hodgkin

## Arquiteturas de redes neurais artificiais e processos de treinamento

### 2.1 – Introdução

A arquitetura de uma rede neural artificial define a forma como os seus diversos neurônios estão arranjados, ou dispostos, uns em relação aos outros. Esses arranjos são essencialmente estruturados através do direcionamento das conexões sinápticas dos neurônios.

Já a topologia de uma rede neural, considerando determinada arquitetura, pode ser definida como sendo as diferentes formas de composições estruturais que esta poderá assumir. Por exemplo, pode-se ter duas topologias pertencentes a uma mesma arquitetura, sendo que uma é composta de 10 neurônios e a outra é de 20 neurônios; ou ainda, uma constituída de neurônios com função de ativação logística, ao passo que os neurônios da outra assumem a tangente hiperbólica como função de ativação.

Por outro lado, o treinamento de uma arquitetura específica consiste da aplicação de um conjunto de passos ordenados com o intuito de ajustar os pesos e os limiares de seus neurônios. Assim, tal processo de ajuste, também conhecido como algoritmo de aprendizagem, visa então sintonizar a rede para que as suas respostas estejam próximas dos valores desejados.

## 2.2 – Principais arquiteturas de redes neurais artificiais

Basicamente, uma rede neural artificial pode ser dividida em três partes, denominadas de camadas, as quais são nomeadas da seguinte forma:

### a) Camada de entrada

É a camada responsável pelo recebimento de informações (dados), sinais, características ou medições advindas do meio externo, sendo que tais entradas (amostras ou padrões) são geralmente normalizadas em relação às faixas de variações dinâmicas (seção 5.6) produzidas pelas funções de ativação. Esta normalização implica numa melhor precisão numérica frente às operações matemáticas realizadas pela rede.

### b) Camadas escondidas, intermediárias, ocultas ou invisíveis

São aquelas compostas de neurônios que possuem a responsabilidade de extrair as características associadas ao processo ou sistema a ser inferido. Quase todo o processamento interno da rede é realizado nessas camadas.

### c) Camada de saída

Esta camada é também constituída de neurônios, sendo responsável pela produção e apresentação dos resultados finais da rede, os quais são advindos dos processamentos efetuados pelos neurônios das camadas anteriores.

As principais arquiteturas de redes neurais artificiais, considerando a disposição de seus neurônios, assim como suas formas de interligação entre eles e a constituição de suas camadas, podem ser divididas em: redes *feedforward* (alimentação à frente) de camada simples, redes *feedforward* de camadas múltiplas, redes recorrentes, e redes reticuladas.

### 2.2.1 – Arquitetura *feedforward* de camada simples

Para este tipo de arquitetura de redes neurais artificiais, tem-se apenas uma camada de entrada e uma única camada de neurônios, que é a própria camada de saída. A figura 2.1 ilustra uma rede *feedforward* de camada simples composta de  $n$  entradas e  $m$  saídas.

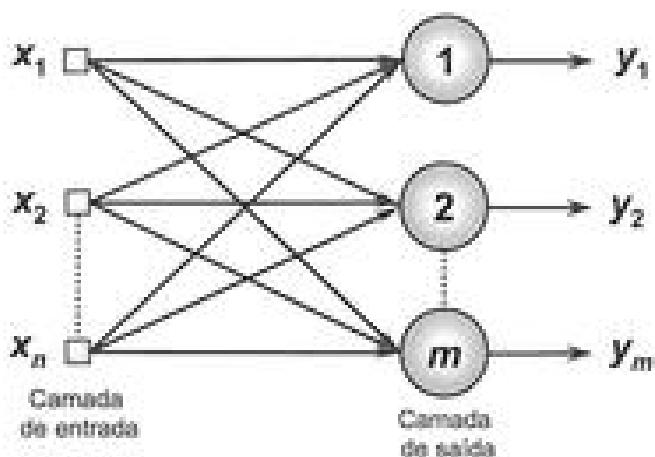


Figura 2.1 – Exemplo de rede *feedforward* de camada única

O fluxo de informações segue sempre numa única direção (unidirecional), ou seja, da camada de entrada em direção à camada de saída. A partir ainda da análise da figura 2.1, observa-se então que a quantidade de saídas nas redes pertencentes à arquitetura sempre coincidirá com o número de neurônios. Essas redes são tipicamente empregadas em problemas envolvendo classificação de padrões e filtragem linear.

Entre os principais tipos de redes tendo arquitetura *feedforward* de camada simples estão o *Perceptron* e o *Adaline*, cujos algoritmos de aprendizado utilizados em seus processos de treinamento são respectivamente baseados na regra de Hebb e na regra Delta, conforme os capítulos posteriores.

### 2.2.2 – Arquitetura *feedforward* de camadas múltiplas

Diferentemente das redes pertencentes à arquitetura anterior, as redes *feedforward* de camadas múltiplas são constituídas pela presença de uma ou mais camadas escondidas de neurônios (figura 2.2). São empregadas na solução de diversos tipos de problemas, tais como aqueles relacionados à aproximação de funções, classificação de padrões, identificação de sistemas, otimização, robótica, ao controle de processos, etc.

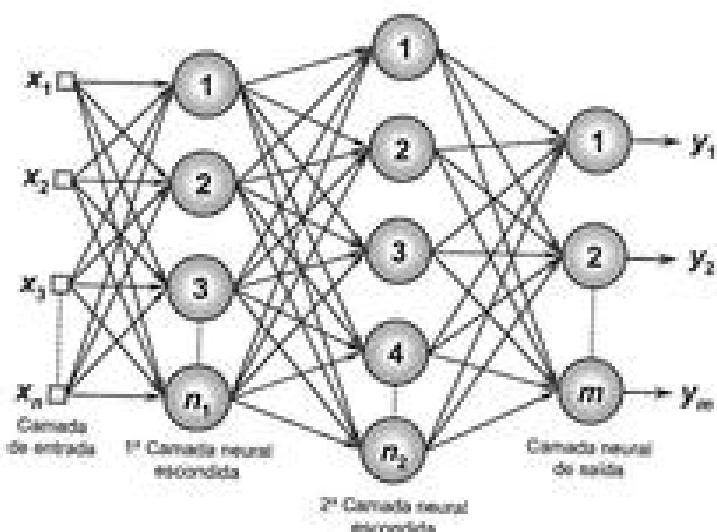


Figura 2.2 – Exemplo de rede *feedforward* de camadas múltiplas

A figura 2.2 ilustra uma rede *feedforward* de camadas múltiplas formada por uma camada de entrada composta de  $n$  sinais, duas camadas neurais escondidas constituídas respectivamente de  $n_1$  e  $n_2$  neurônios e, finalmente, uma camada neural de saída composta de  $m$  neurônios representando os respectivos valores de saída da aplicação.

Entre os principais tipos de redes com arquitetura *feedforward* de camadas múltiplas se encontra o *Perceptror multicamadas (multilayer Perceptron – MLP)* e as redes de base radial (*radial basis function – RBF*), cujos algoritmos de aprendizado utilizados em seus processos de treinamento são respectivamente baseados na regra delta generalizada e na regra delta/competitiva, conforme os capítulos posteriores.

A partir ainda da figura 2.2 é possível abstrair que a quantidade de neurônios que compõem a primeira camada escondida é normalmente diferente do número de sinais que compõem a camada de entrada da rede. De fato, o número de camadas escondidas e seus respectivos neurônios constituintes dependem, sobretudo, do tipo e da complexidade do problema a ser mapeado pela rede, assim como da quantidade e da qualidade dos dados disponíveis sobre o problema. Entretanto, conforme acontece nas redes *feedforward* de camada simples, a quantidade de saídas sempre coincidirá com o número de neurônios daquela respectiva camada.

### 2.2.3 – Arquitetura recorrente ou realimentada

São redes em que as saídas dos neurônios são realimentadas como sinal de entrada para outros neurônios. A característica da realimentação qualifica tais redes para processamento dinâmico de informações, isto é, estas podem ser utilizadas em sistemas variantes em relação ao tempo, como previsão de séries temporais, otimização e identificação de sistemas, controle de processos, etc.

Entre os principais tipos de redes que possuem realimentação estão a rede de Hopfield e a rede *Perceptron* multicamadas com realimentação entre neurônios situados em camadas distintas, cujos algoritmos de aprendizado utilizados em seus processos de treinamento são, respectivamente, baseados na minimização de funções energia e na regra delta generalizada, conforme serão também investigados nos próximos capítulos.

A figura 2.3 ilustra um exemplo de rede *Perceptron* com realimentação, tendo-se um dos sinal da camada de saída retroalimentado à camada intermediária.

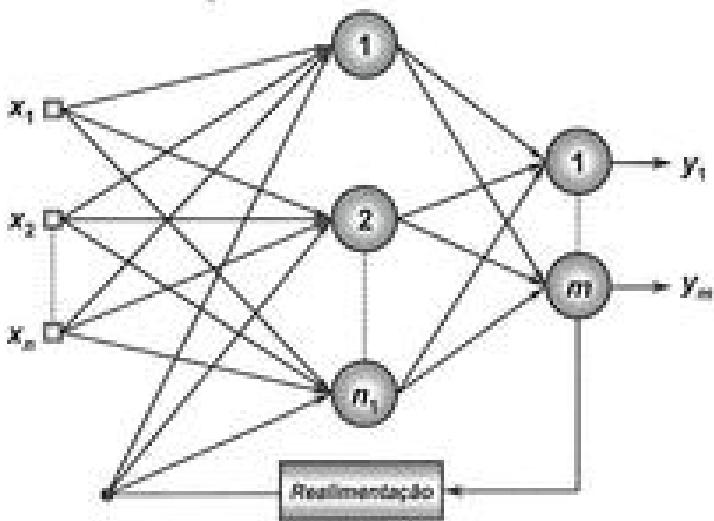


Figura 2.3 – Exemplo de rede recorrente

Assim, por intermédio do processo de realimentação, as redes com este tipo de arquitetura produzem as saídas atuais levando-se também em consideração os valores das saídas anteriores.

### 2.2.4 – Arquitetura em estrutura reticulada

As principais características de redes com estruturas reticuladas estão na consideração da disposição espacial dos neurônios visando propósitos de extração de características, ou seja, a localização espacial dos neurônios está diretamente relacionada com o processo de ajuste de seus pesos e limiares. Suas aplicações são bem diversificadas, tais como em problemas de agrupamento (*clustering*), reconhecimento de padrões, otimização de sistemas, grafos, etc.

A rede de Kohonen é a principal representante da arquitetura em estrutura reticulada, cujo treinamento é realizado por meio de um processo competitivo, conforme os capítulos posteriores. A figura 2.4 ilustra um exemplo de rede de Kohonen em que os neurônios estão dispostos no espaço bidimensional,

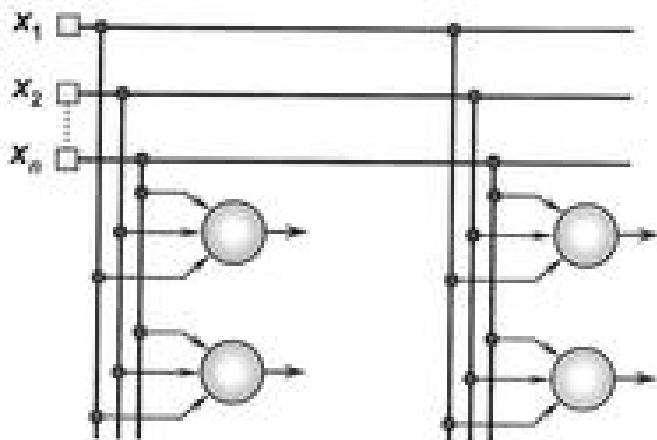


Figura 2.4 – Exemplo de rede com estrutura reticulada

Ainda a partir da figura 2.4, verifica-se que nesta tipologia de rede os diversos sinais de entrada são também inseridos em todos os neurônios da rede.

### 2.3 – Processos de treinamento e aspectos de aprendizado

Um dos destaques mais relevantes das redes neurais artificiais está na capacidade de aprender a partir da apresentação de amostras (padrões) que exprimem o comportamento do sistema, sendo que, em seguida, após a rede ter aprendido o relacionamento entre as entradas e saídas, esta é ca-

paz de generalizar soluções. A rede será então capaz de produzir uma saída próxima daquela esperada (desejada) a partir de quaisquer sinais inseridos em suas entradas.

Portanto, o processo de treinamento de uma rede neural consiste da aplicação de passos ordenados que sejam necessários para sintonização dos pesos sinápticos e limiares de seus neurônios, tendo-se como objetivo final a generalização de soluções a serem produzidas pelas suas saídas, cujas respostas são representativas do sistema físico em que estas estão mapeando.

O conjunto desses passos ordenados visando o treinamento da rede é denominado de algoritmo de aprendizagem. Ao longo de sua aplicação, a rede será então capaz de extrair características discriminantes do sistema a ser mapeado por intermédio de amostras que foram retiradas de seu contexto.

Normalmente, o conjunto total das amostras disponíveis sobre o comportamento do sistema é dividido em dois subconjuntos, os quais são denominados de subconjunto de treinamento e subconjunto de teste. O subconjunto de treinamento, composto aleatoriamente com cerca de 60 a 90% das amostras do conjunto total, será usado essencialmente no processo de aprendizado da rede. Já o subconjunto de teste, cuja composição está entre 10% e 40% do conjunto total de amostras, será utilizado para verificar se os aspectos referentes à generalização de soluções por parte da rede já estão em patamares aceitáveis, possibilitando assim a validação da topologia assumida. Contudo, o dimensionamento desses conjuntos deve também levar em consideração a caracterização estatística dos dados.

Durante o processo de treinamento de redes neurais artificiais, cada apresentação completa das amostras pertencentes ao subconjunto de treinamento, visando, sobretudo, o ajuste dos pesos sinápticos e limiares de seus neurônios, será denominada de época de treinamento.

### 2.3.1 – Treinamento supervisionado

A estratégia de treinamento supervisionado consiste em se ter disponível, considerando cada amostra dos sinais de entrada, as respectivas saídas desejadas, ou seja, cada amostra de treinamento é composta pelos sinais de entradas e suas correspondentes saídas. Desta forma, há então a necessidade de se disponibilizar uma tabela de dados (entradas/saídas) representativa do

processo, também conhecida por tabela atributos/valores, e que contemple inclusive o seu comportamento, pois é a partir de tais informações que as estruturas neurais formularão as "hipóteses" sobre aquilo a ser aprendido.

Neste caso, a aplicação do treinamento supervisionado depende apenas da disponibilidade desta tabela de atributos/valores, sendo que tudo se comporta como se houvesse um "professor" ensinando para a rede qual a resposta correta para cada amostra apresentada em suas entradas.

Os pesos sinápticos e limiares são então continuamente ajustados mediante a aplicação de ações comparativas, executadas pelo próprio algoritmo de aprendizagem, que supervisionam a defasagem entre as respostas produzidas pela rede em relação àquelas desejadas, sendo esta diferença usada no procedimento de ajuste. A rede será considerada treinada quando tal defasagem estiver dentro de valores aceitáveis, levando-se em consideração os propósitos de generalização de soluções.

Na realidade, o treinamento supervisionado é um caso típico de inferência indutiva pura, em que os parâmetros livres da rede são ajustados em função de se conhecer *a priori* quais são as saídas desejadas ao sistema investigado.

A primeira estratégia de treinamento supervisionado foi proposta em 1949 por Donald Hebb, cuja inspiração advém de observações neurofisiológicas [Hebb, 1949].

### 2.3.2 – Treinamento não-supervisionado

Diferentemente do supervisionado, durante a aplicação de um algoritmo de aprendizado baseado em treinamento não-supervisionado inexistem as respectivas saídas desejadas.

Consequentemente, a própria rede deve se auto-organizar em relação às particularidades existentes entre os elementos componentes do conjunto total de amostras, identificando subconjuntos (*clusters*) que contenham similaridades. Os pesos sinápticos e limiares dos neurônios da rede são então ajustados pelo algoritmo de aprendizado de forma a refletir esta representação internamente dentro da própria rede.

Alternativamente, a quantidade máxima desses possíveis *clusters* pode ser especificada (*a priori*) pelo próprio projetista da rede, levando-se em consideração o seu conhecimento a respeito do problema a ser tratado.

### 2.3.3 – Treinamento com reforço

Os métodos baseados em treinamento com reforço têm sido considerados uma variação das técnicas que utilizam treinamento supervisionado, as quais avaliam constantemente a defasagem de valor entre a resposta produzida pela rede em relação à respectiva saída desejada [Sutton & Barto, 1998]. Os algoritmos de aprendizado utilizados no treinamento com reforço ajustam os parâmetros internos dos neurônios baseando-se em quaisquer informações quantitativas ou qualitativas advindas da interação com o sistema (ambiente) que está sendo mapeado, as quais são então utilizadas para medir o desempenho do aprendizado.

Frente a essas situações, o processo de treinamento da rede é realizado tipicamente por tentativa e erro, pois a única resposta disponível para uma determinada entrada é se esta é satisfatória ou não. Se for considerada satisfatória, incrementos nos pesos sinápticos e limiares são então gradualmente efetuados visando reforçar (recompensar) esta condição comportamental envolvida com o sistema.

Diversos algoritmos de aprendizado utilizados no treinamento com reforço são baseados em métodos estocásticos que selecionam probabilisticamente suas ações de ajustes, levando-se em conta um conjunto finito de possíveis situações, as quais são passíveis de serem recompensadas frente às chances de se gerar resultados satisfatórios. No decorrer do processo de treinamento, as probabilidades associadas às ações de ajustes são ainda modificadas visando melhorar o desempenho da rede [Tsoukalas & Uhrig, 1997].

Este mecanismo de ajuste possui também certa similaridade com técnicas aplicadas em programação dinâmica [Bertsekas & Tsitsiklis, 1996; Watkins, 1989].

### 2.3.4 – Aprendizagem usando lote de amostras (off-line)

Na aprendizagem usando lote de amostras (padrões), também denominada de aprendizagem *off-line* ou *batch*, os ajustes efetuados nos vetores de pesos das redes e em seus limiares só são efetivados após a apresentação de todo o conjunto de treinamento, pois cada passo de ajuste leva em consideração o total de desvios observados nas amostras de treinamento frente aos respectivos valores desejados para as suas saídas.

Assim, as redes que utilizam aprendizagem usando lote de padrões necessitam de pelo menos uma época de treinamento para se realizar um passo de ajuste em seus pesos e limiares. Portanto, todas as amostras de treinamento devem estar sempre disponíveis enquanto durar o processo de aprendizagem.

### 2.3.5 - Aprendizagem usando padrão-por-padrão (*on-line*)

Para a aprendizagem usando padrão-por-padrão (*on-line*), ao contrário daquela por lote, os ajustes nos pesos e limiares das redes são efetuados após a apresentação de cada amostra de treinamento. Portanto, após a execução do passo de ajuste, a respectiva amostra pode ser descartada.

A aprendizagem *on-line* nesta configuração é normalmente utilizada quando o comportamento do sistema a ser mapeado varia de forma bastante rápida, sendo quase impraticável a adoção do aprendizado *off-line*, pois as amostras utilizadas em determinado instante podem não mais representar o comportamento do processo nos instantes posteriores.

Entretanto, como os padrões são apresentados um por vez, as ações de ajuste dos pesos e limiares são bem localizadas e pontuais, cujos reflexos remetem à determinada circunstância comportamental do sistema. Consequentemente, a rede apenas começará a fornecer respostas mais precisas após a apresentação de um número significativo de amostras [Reed & Marks II, 1999].

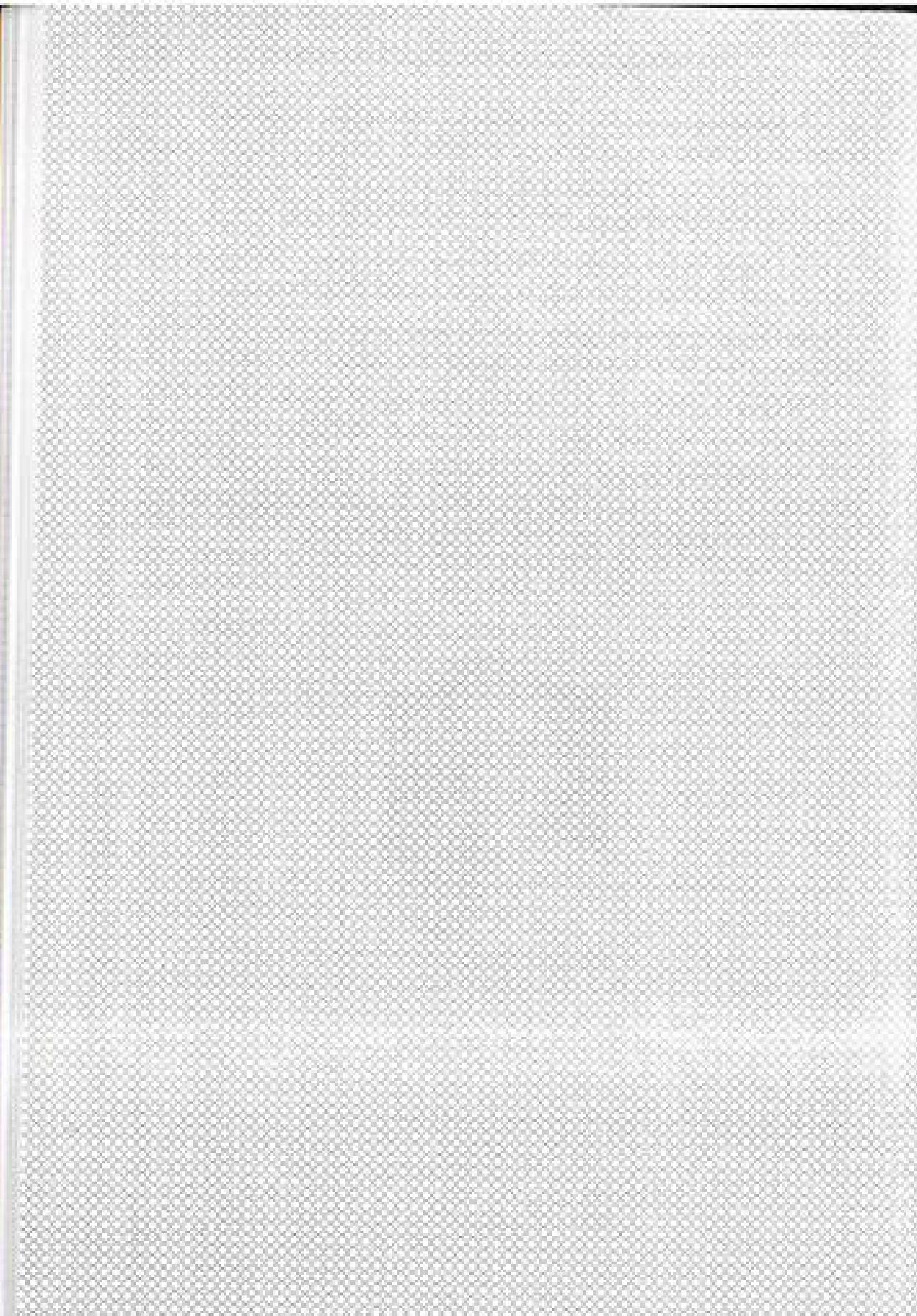
## 2.4 - Exercícios

- 1) Discorra sobre as vantagens e desvantagens envolvidas na aprendizagem usando lote de padrões e aprendizagem usando padrão-por-padrão.
- 2) Considere uma aplicação que possui quatro entradas e duas saídas. O projectista menciona que neste caso a rede *feedforward* de camadas múltiplas a ser implementada deve conter necessariamente quatro neurônios na primeira camada escondida. Discorra se tal informação é pertinente.
- 3) Em relação ao exercício anterior, cite alguns fatores que influenciam na determinação do número de camadas escondidas de uma rede *feedforward* de camadas múltiplas.
- 4) Quais as eventuais diferenças estruturais observadas nas redes com arquitetura recorrente em relação àsquelas com arquitetura *feedforward*.

- 5) Mencione em que tipos de aplicações é essencial a utilização de redes neurais recorrentes.
- 6) Elabore um diagrama de blocos que ilustre o funcionamento do treinamento supervisionado.
- 7) Discorra sobre o conceito de método de treinamento e algoritmo de aprendizado, explicitando-se ainda o conceito de época de treinamento.
- 8) Quais as principais diferenças existentes entre os métodos baseados em treinamento supervisionado e não-supervisionado.
- 9) Quais as principais diferenças existentes entre os métodos baseados em treinamento supervisionado e treinamento com reforço.
- 10) Considerando uma aplicação específica, explice então como poderia ser um critério de desempenho utilizado para o ajuste dos pesos e limiares da rede que empregará método de treinamento com reforço.



Donald Olding Hebb



## Rede Perceptron

### 3.1 – Introdução

O *Perceptron*, idealizado por Rosenblatt (1958), é a forma mais simples de configuração de uma rede neural artificial, cujo propósito focava em implementar um modelo computacional inspirado na retina, objetivando-se então um elemento de percepção eletrônica de sinais. Uma de suas aplicações consistia de identificar padrões geométricos.

A figura 3.1 mostra uma ilustração que exemplifica a concepção inicial do elemento *Perceptron*, em que os sinais elétricos advindos de fotocélulas mapeando padrões geométricos eram ponderados por resistores sintonizáveis, os quais podiam ser ajustados durante o processo de treinamento. Em seguida, um somador efetuava a composição de todos os sinais. Desta forma, o *Perceptron* poderia reconhecer diversos padrões geométricos, tais como letras e números.

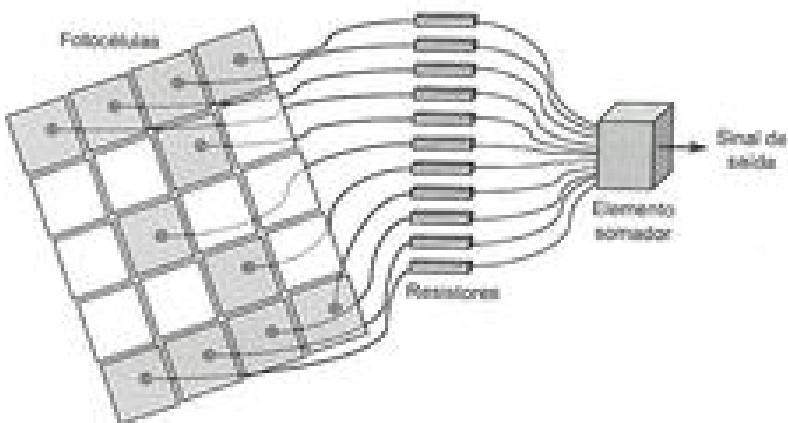


Figura 3.1 – Modelo ilustrativo do *Perceptrón* para reconhecimento de padrões

A simplicidade da rede *Perceptrón* está associada à sua condição de ser constituída de apenas uma camada neural, tendo-se também somente um neurônio artificial nesta única camada.

A figura 3.2 ilustra uma rede *Perceptrón* constituída de  $n$  sinais de entrada, representativas do problema a ser mapeado, e somente uma saída, pois a mesma é composta de um único neurônio.

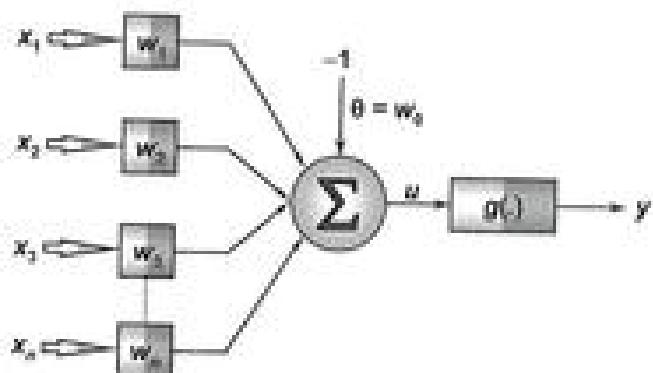


Figura 3.2 – Ilustração da rede *Perceptrón*

Embora seja uma rede simples, o *Perceptrón* teve o potencial de atrair, quando de sua proposição, diversos pesquisadores que aspiravam investigar essa promissora área de pesquisa para a época, recebendo-se ainda especial atenção da comunidade científica que também trabalhava com inteligência artificial.

Visando aspectos de implementação computacional, conforme será abordado na seção 3.4, observa-se na figura 3.2 que o valor do limiar de ativação ( $\theta$ ) foi assumido (sem qualquer perda de interpretabilidade) como sendo também um termo de ponderação ( $w_0$ ), tendo-se então o valor unitário negativo como respectiva entrada.

Com base no exposto no capítulo anterior, a rede *Perceptron* pertence à arquitetura *feedforward* de camada única, pois o fluxo de informações em sua estrutura resiste sempre no sentido da camada de entrada em direção à camada neural de saída, inexistindo-se qualquer tipo de realimentação de valores produzidos pelo seu único neurônio.

### 3.2 - Princípio de funcionamento do Perceptron

Assim como verificado na simplicidade estrutural do *Perceptron*, o seu princípio de funcionamento é também extremamente simples.

A partir da análise da figura 3.2, observa-se que cada uma das entradas ( $x_i$ ), as quais representam informações sobre o comportamento do processo a ser mapeado, será inicialmente ponderada pelos pesos sinápticos ( $w_j$ ) a fim de quantificar a importância de cada uma frente aos objetivos funcionais atribuídos ao neurônio, cujo propósito será então mapear o comportamento entrada/saída do referido processo.

Em seguida, o valor resultante da composição de todas as entradas já devidamente ponderadas pelos seus respectivos pesos, adicionado ainda do limiar de ativação ( $\theta$ ), é repassado como argumento da função de ativação, cujo resultado de retorno será a saída ( $y$ ) produzida pelo *Perceptron*.

Em termos matemáticos, o processamento interno realizado pelo *Perceptron* pode ser descreto pelas seguintes expressões:

$$\begin{cases} u = \sum_{j=1}^n w_j \cdot x_j + \theta \\ y = g(u) \end{cases} \quad (3.1)$$

$$\begin{cases} u = \sum_{j=1}^n w_j \cdot x_j + \theta \\ y = g(u) \end{cases} \quad (3.2)$$

onde  $x_i$  são as entradas da rede,  $w_i$  é o peso (ponderação) associado à  $i$ -ésima entrada,  $\theta$  é o limiar de ativação,  $g(\cdot)$  é a função de ativação e  $y$  é o potencial de ativação.

Tipicamente, devido às suas características estruturais, as funções de ativação normalmente usadas no *Pomphros* são a função degrau ou degrau bipolar, conforme apresentadas na subseção 1.3.1. Assim, independente da função de ativação a ser utilizada, tem-se apenas duas possibilidades de valores a serem produzidos pela sua saída, ou seja, valor 0 ou 1 se for considerada a função de ativação degrau, ou ainda, valor -1 ou 1 se for assumida a função degrau bipolar.

Em relação às entradas  $\{x_i\}$ , estas podem assumir quaisquer valores numéricos, ficando basicamente em função do tipo de problema a ser mapeado pelo *Pomphros*. Na prática, técnicas que normalizam as entradas, considerando os limites numéricos produzidos pelas funções de ativação adotadas, visam melhorar o desempenho computacional do processo de treinamento.

Resumindo, a tabela 3.1 explicita os aspectos característicos dos parâmetros relacionados com a dinâmica de funcionamento do *Pomphros*.

Tabela 3.1 – Aspectos dos parâmetros característicos do *Pomphros*

Parâmetro	Variável representativa	Tipo característico
Entradas	$x_i$ ( $i$ -ésima entrada)	Reais ou binárias (advindas externamente)
Pesos sinápticos	$w_i$ (associado a $x_i$ )	Reais (iniciados aleatoriamente)
Limiar	$\theta$	Real (iniciado aleatoriamente)
Saída	$y$	Binária
Função de ativação	$g(\cdot)$	Degrado ou degrado bipolar
Processo de treinamento	-----	Supervisionado
Regra de aprendizado	-----	Regra de Hebb

O ajuste dos pesos e limiar do *Pomphros* é efetuado utilizando processo de treinamento supervisionado, isto é, para cada amostra dos sinais de entrada se tem a respectiva saída (resposta) desejada. Como o *Pomphros* é tipicamente usado em problemas de reconhecimento de padrões, sendo que sua saída pode assumir somente dois valores possíveis, então cada um de tais valores será associado a uma das duas classes que o *Pomphros* estará identificando.

Mais especificamente, considerando um problema envolvendo a classificação dos sinais de entrada em duas classes possíveis, denominadas de *classe A* e *classe B*, seria então possível (assumindo como exemplo a função de ativação degrau bipolar) atribuir o valor -1 para representar as amostras pertencentes à *classe A*, ao passo que o valor 1 seria usado para aquelas da *classe B*, ou vice-versa.

### 3.3 – Análise matemática do Perceptron

A partir da análise matemática do *Perceptron*, considerando a função de ativação sinal, torna-se possível verificar que tal elemento pode ser considerado um típico caso de discriminador linear. Para mostrar esta situação, assume-se um *Perceptron* com apenas duas entradas, conforme ilustrado na figura 3.3.

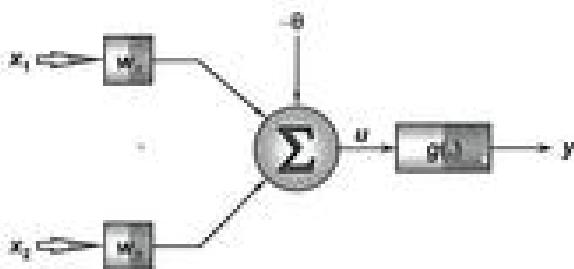


Figura 3.3 – Perceptron constituído de duas entradas

Em termos matemáticos, a saída do *Perceptron*, tendo-se assim como ativação a função sinal definida em (1.5), será dada por:

$$y = \begin{cases} 1, & \text{se } \sum w_i \cdot x_i - b \geq 0 \Leftrightarrow w_1 \cdot x_1 + w_2 \cdot x_2 - b \geq 0 \\ -1, & \text{se } \sum w_i \cdot x_i - b < 0 \Leftrightarrow w_1 \cdot x_1 + w_2 \cdot x_2 - b < 0 \end{cases} \quad (3.3)$$

Sendo as desigualdades em (3.3) representadas por uma expressão de primeiro grau (linear), a fronteira de decisão para esta instância (*Perceptron* de duas entradas) será então uma reta cuja equação é definida por:

$$w_1 \cdot x_1 + w_2 \cdot x_2 - b = 0 \quad (3.4)$$

Portanto, pode-se concluir que o *Perceptron* se comporta como um classificador de padrões cuja função é dividir classes que sejam linearmente separáveis. Para o caso do *Perceptron* de duas entradas, a figura 3.4 ilustra uma reta posicionada na fronteira de separabilidade.

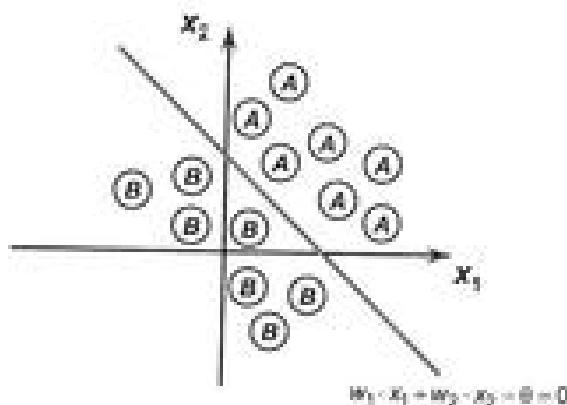


Figura 3.4 – Ilustração de fronteira de separação (neurônio com duas entradas).

Em suma, para a circunstância verificada na figura 3.4, o *Perceptron* consegue então dividir duas classes linearmente separáveis, sendo que quando a saída deste for -1 significa que os padrões (classe A) estão localizados abaixo da fronteira (reta) de separação; caso contrário, quando a saída for 1 indica que os padrões (classe B) estão acima desta fronteira. A figura 3.5 ilustra uma configuração em que as classes não são linearmente separáveis, isto é, uma única reta seria incapaz de separar as duas classes do problema.

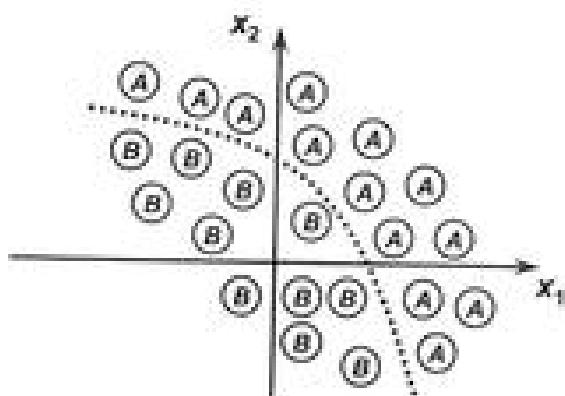


Figura 3.5 – Ilustração de fronteira não linearmente separável

Considerando o fato de o *Perceptron* ser constituído de três entradas (três dimensões), a fronteira de separação seria representada por um plano; sendo que, para dimensões superiores, tais fronteiras seriam hiperplanos.

Finalmente, conclui-se que a condição para que o *Perceptron* de camada simples possa ser utilizado como um classificador de padrões consiste em que as classes do problema a ser mapeado sejam linearmente separáveis. Este princípio condicional foi enunciado como Teorema de Convergência do *Perceptron* [Minsky & Papert, 1969].

### 3.4 – Processo de treinamento do *Perceptron*

O ajuste dos pesos e limiar do *Perceptron*, visando-se propósitos de classificação de padrões que podem pertencer a uma das duas únicas classes possíveis, é realizado por meio da regra de aprendizado de Hebb [Hebb, 1949].

Resumidamente, se a saída produzida pelo *Perceptron* está coincidente com a saída desejada, os pesos sinápticos e limiares da rede serão então incrementados (ajuste excitatório) proporcionalmente aos valores de seus sinais de entrada; caso contrário, ou seja, a saída produzida pela rede é diferente do valor desejado, os pesos sinápticos e limiar serão então decrementados (inibitório). Este processo é repetido, sequencialmente para todas as amostras de treinamento, até que a saída produzida pelo *Perceptron* seja similar à saída desejada de cada amostra. Em termos matemáticos, as regras de ajuste dos pesos sinápticos  $\{w_i\}$  e do limiar  $\{b\}$  do neurônio podem ser expressas, respectivamente, pelas seguintes expressões:

$$\left\{ \begin{array}{l} w_i^{(n+1)} = w_i^{(n)} + \eta \cdot (\sigma^{(n)} - y) \cdot x_i^{(n)} \end{array} \right. \quad (3.5)$$

$$\left\{ \begin{array}{l} b^{(n+1)} = b^{(n)} + \eta \cdot (\sigma^{(n)} - y) \end{array} \right. \quad (3.6)$$

Entretanto, em termos de implementação computacional, torna-se mais conveniente tratar as expressões anteriores em sua forma vetorial. Como a mesma regra de ajuste é aplicada tanto para os pesos sinápticos como para o limiar, pode-se então inserir o valor do limiar  $\{0\}$  dentro do vetor de pesos sinápticos. De fato, o valor do limiar é também uma variável que deve ser ajustada a fim de se realizar o treinamento do *Perceptron*. Portanto, as expressões (3.5) e (3.6) podem ser representadas por uma única expressão vetorial dada por:

$$\mathbf{w}^{atual} = \mathbf{w}^{anterior} + \eta \cdot (\sigma^{(k)} - y) \cdot \mathbf{x}^{(k)} \quad (3.7)$$

Em notação algorítmica, a expressão anterior é equivalente à seguinte:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot (\sigma^{(k)} - y) \cdot \mathbf{x}^{(k)} \quad (3.8)$$

onde:  $\mathbf{w} = [0 \ w_1 \ w_2 \dots w_n]^T$  é o vetor contendo o limiar e os pesos;

$\mathbf{x}^{(k)} = [-1 \ x_1^{(k)} \ x_2^{(k)} \dots x_n^{(k)}]^T$  é a  $k$ -ésima amostra de treinamento;

$\sigma^{(k)}$  é o valor desejado para a  $k$ -ésima amostra de treinamento;

$y$  é valor da saída produzida pelo *Perceptror*;

$\eta$  é uma constante que define a taxa de aprendizagem da rede.

A taxa de aprendizagem  $\{\eta\}$  exprime o quanto rápido o processo de treinamento da rede estará sendo conduzido rumo à sua convergência (estabilização). A escolha de  $\eta$  deve ser realizada com cautela para evitar instabilidades no processo de treinamento, sendo que normalmente se adotam valores pertencentes ao intervalo compreendido em  $0 < \eta < 1$ .

A fim de elucidar a notação utilizada para o vetor  $\mathbf{x}^{(k)}$  representando a  $k$ -ésima amostra de treinamento, assim como o respectivo valor desejado  $\sigma^{(k)}$ , supõe-se que um problema a ser mapeado pelo *Perceptror* tenha três entradas  $\{x_1, x_2, x_3\}$ . Assume-se que o conjunto de treinamento seja composto de apenas quatro amostras, constituídas dos seguintes valores:  $\Omega^a = \{[0,1 \ 0,4 \ 0,7]; [0,3 \ 0,7 \ 0,2]; [0,6 \ 0,9 \ 0,8]; [0,5 \ 0,7 \ 0,1]\}$ . Considerando-se ainda que os respectivos valores de saída para cada uma dessas amostras seja dado por  $\Omega^m = \{[1]; [-1]; [-1]; [1]\}$ , então se pode adotar a seguinte forma matricial para suas representações:

$$\Omega(x) = \begin{bmatrix} x_0 & x^{(1)} & x^{(2)} & x^{(3)} & x^{(4)} \\ x_1 & -1 & -1 & -1 & -1 \\ x_2 & 0,1 & 0,3 & 0,6 & 0,5 \\ x_3 & 0,4 & 0,7 & 0,9 & 0,7 \\ x_4 & 0,7 & 0,2 & 0,8 & 0,1 \end{bmatrix}; \quad \Omega^{(d)} = \begin{bmatrix} d^{(1)} & d^{(2)} & d^{(3)} & d^{(4)} \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Nesta condição, alternativamente, pode-se extrair dessas matrizes cada um dos vetores  $x^{(i)}$ , com seu respectivo valor  $d^{(i)}$ , os quais representarão cada uma das amostras de treinamento, da seguinte forma:

$$x^{(1)} = [-1 \ 0,1 \ 0,4 \ 0,7]^T; \text{ com } d^{(1)} = 1$$

$$x^{(2)} = [-1 \ 0,3 \ 0,7 \ 0,2]^T; \text{ com } d^{(2)} = -1$$

$$x^{(3)} = [-1 \ 0,6 \ 0,9 \ 0,8]^T; \text{ com } d^{(3)} = -1$$

$$x^{(4)} = [-1 \ 0,5 \ 0,7 \ 0,1]^T; \text{ com } d^{(4)} = 1$$

Entretanto, em se tratando de implementações computacionais, a disposição dos dados por meio de variáveis indexadas se torna bem mais adequada para a manipulação dos índices das amostras.

Assim, a sequência passo a passo para o treinamento do *Perceptron* pode ser explicitada por intermédio de algoritmo em pseudocódigo, conforme se segue:

**Início (Algoritmo Perceptron – Fase de Treinamento)**

- <1> Obter o conjunto de amostras de treinamento ( $x^{(k)}$ );
- <2> Associar a saída desejada ( $d^{(k)}$ ) para cada amostra obtida;
- <3> Iniciar o vetor  $w$  com valores aleatórios pequenos;
- <4> Especificar a taxa de aprendizagem ( $\eta$ );
- <5> Iniciar o contador de número de épocas (época  $\leftarrow 0$ );
- <6> Repetir as instruções:
  - <6.1> erro  $\leftarrow$  "inexistente";
  - <6.2> Para todas as amostras de treinamento ( $x^{(k)}, d^{(k)}$ ), fazer:
    - <6.2.1>  $v \leftarrow w^T \cdot x^{(k)}$ ;
    - <6.2.2>  $y \leftarrow \text{sign}(v)$ ;
    - <6.2.3> Se  $y \neq d^{(k)}$ 
      - <6.2.3.1> Então  $w \leftarrow w + \eta \cdot (d^{(k)} - y) \cdot x^{(k)}$
      - erro  $\leftarrow$  "existente"
  - <6.3> época  $\leftarrow$  época + 1;
  - Até que: erro  $\leftarrow$  "inexistente"

**Fim (Algoritmo Perceptron – Fase de Treinamento)**

Analizando o algoritmo envolvido com o treinamento do *Perceptron*, verifica-se que a variável *época* estará incumbida de contabilizar o próprio número de épocas de treinamento, ou seja, quantas vezes serão necessárias apresentar todas as amostras do conjunto de treinamento visando o ajuste do vetor de pesos ( $w$ ). A rede será considerada treinada (ajustada) quando inexistir erro entre os valores desejados em comparação com aqueles produzidos em suas saídas.

Uma vez que esteja treinada, a rede estará apta para proceder com a tarefa de classificação de padrões frente às novas amostras que serão apresentadas em suas entradas. Portanto, as instruções para colocar o *Perceptron* em operação, após a realização de seu treinamento, são sintetizadas no algoritmo seguinte.

**Início {Algoritmo Perceptron – Fase de Operação}**

- {<1> Obter uma amostra a ser classificada ( $x$ );
- <2> Utilizar o vetor  $w$  ajustado durante o treinamento;
- <3> Executar as seguintes instruções:
  - <3.1>  $u \leftarrow w^T \cdot x$ ;
  - <3.2>  $y \leftarrow \text{sign}(u)$ ;
  - <3.3> Se  $y = -1$ 
    - <3.3.1> Então: amostra  $x \in$  /Classe A/
  - <3.4> Se  $y = 1$ 
    - <3.4.1> Então: amostra  $x \in$  /Classe B/

**Fim {Algoritmo Perceptron – Fase de Operação}**

Portanto, pode-se abstrair que o processo de treinamento do *Perceptron* tende a mover continuamente o hiperplano de classificação até que seja alcançada uma fronteira de separação que permite dividir as duas classes.

A figura 3.6 mostra uma ilustração do processo de treinamento do *Perceptron* visando o alcance desta fronteira de separabilidade. Para fins didáticos, considere-se uma rede composta de apenas duas entradas ( $x_1$  e  $x_2$ ).

Após a primeira época de treinamento (1), constata-se que o hiperplano está ainda bem longínquo da fronteira de separabilidade das classes, ao passo que esta distância tende a decrescer cada vez mais na medida em que se transcorre as épocas de treinamento.

Em consequência, quando o *Perceptron* já estiver convergido, significará que tal fronteira foi finalmente alcançada, sendo que as saídas produzidas pelo *Perceptron* a partir deste momento serão todas iguais àquelas desejadas.

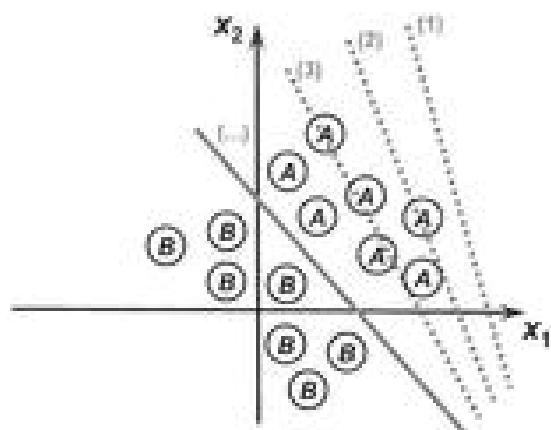


Figura 3.6 – Ilustração do processo de treinamento do *Perceptron*

Analizando a figura 3.6, observa-se ainda a possibilidade de se ter diversas retas que permitam separar as duas classes envolvidas com o problema. A figura 3.7 ilustra um conjunto de eventuais retas que são também passíveis de separar tais classes.

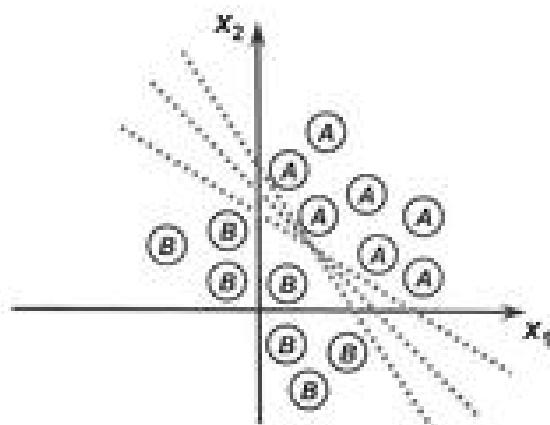


Figura 3.7 – Ilustração de conjunto de retas passíveis de separar as classes

Assim, para este problema de classificação de padrões usando o *Perceptron*, pode-se também abstrair que a reta de separabilidade a ser produzida após o seu treinamento não é única, sendo que, nesses casos, o número de épocas pode também variar.

Na sequência são apresentados alguns aspectos práticos envolvendo o processo de treinamento do *Perceptron*.

- A rede divergirá se o problema for não linearmente separável. A estratégia a ser usada nesta ocorrência é limitar o processo por meio da especificação de um número máximo de épocas;
- Quando a faixa de separabilidade entre as duas classes do problema for muito estreita, o seu processo de treinamento pode implicar em instabilidades. Em tais casos, assumindo-se um valor de taxa de aprendizado ( $\eta$ ) bem pequeno, a instabilidade da convergência pode ser então aliviada;
- A quantidade de épocas necessárias para a convergência do processo de treinamento do *Perceptron* varia em função dos valores iniciais que foram atribuídos ao vetor de pesos ( $w$ ), assim como da disposição espacial das amostras de treinamento e do valor especificado para a taxa de aprendizado ( $\eta$ );
- Quanto mais próxima a superfície de decisão estiver da fronteira de separabilidade, menos épocas para a convergência do *Perceptron* são geralmente necessárias;
- A normalização das entradas para domínios apropriados contribui para incrementar o desempenho do treinamento.

### 3.5 – Exercícios

- 1) Explique como se processa a regra de Hebb no contexto do algoritmo de aprendizado do *Perceptron*.
- 2) Mostre por intermédio de gráficos ilustrativos como pode ocorrer a instabilidade no processo de convergência do *Perceptron* quando da utilização de valores inapropriados para a taxa de aprendizado.
- 3) Explique por que o *Perceptron* somente consegue classificar padrões cuja fronteira de separação entre as classes seja linear.
- 4) Em termos de implementação computacional descreva a importân-

cia de tratarmos o limiar de ativação {0} como um dos elementos do vetor de pesos {w}.

5) Seja um problema de classificação de padrões que se desconhece *a priori* se as suas duas classes são ou não-separáveis linearmente. Elabore uma estratégia para verificar a possível aplicação do *Perceptron* em tal problema.

6) Dois projetistas de instituições diferentes estão aplicando uma rede *Perceptron* para mapear o mesmo problema de classificação de padrões. Discorra se é correto afirmar que ambas as redes convergirão com o mesmo número de épocas.

7) Em relação ao exercício anterior, considere-se que ambas as redes já estão devidamente treinadas. Para um conjunto contendo 10 novas amostras que devem ser identificadas, explique se os resultados produzidos por ambas serão os mesmos.

8) Seja um problema de classificação de padrões que seja linearmente separável composto de 50 amostras. Em determinada época de treinamento observou-se que somente para uma dessas amostras a rede não estava produzindo a resposta desejada. Discorra se é então necessário apresentar novamente todas as 50 amostras na próxima época de treinamento.

9) Considere um problema de classificação de padrões composto de duas entradas  $\{x_1 \text{ e } x_2\}$ , cujo conjunto de treinamento é composto pelas seguintes amostras de treinamento:

$x_1$	$x_2$	Classe
0,75	0,75	A
0,75	0,25	B
0,25	0,75	B
0,25	0,25	A

Mostre se é possível aplicar o *Perceptron* na resolução deste problema.

10) Explique de forma detalhada quais seriam as eventuais limitações do *Perceptron* se considerarmos o seu limiar de ativação nulo.

### 3.6 – Projeto prático

Pela análise de um processo de destilação fracionada de petróleo observou-se que determinado óleo poderia ser classificado em duas classes de pureza ( $P_1$  e  $P_2$ ) a partir da medição de três grandezas ( $x_1$ ,  $x_2$  e  $x_3$ ), que representam algumas de suas propriedades físico-químicas. A equipe de engenheiros e cientistas pretende usar uma rede *Perceptron* para executar a classificação automática das duas classes.

Assim, baseado nas informações coletadas do processo, formou-se o conjunto de treinamento apresentado no apêndice 1, tornando por convenção o valor -1 para óleo pertencente à classe  $P_1$  e o valor 1 para óleo pertencente à classe  $P_2$ .

Para tanto, o neurônio constituinte do *Perceptron* terá então três entradas e uma saída conforme ilustrado na figura 3.8.

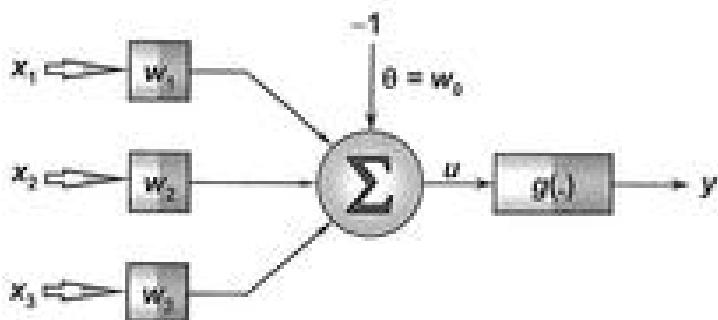


Figura 3.8 – Arquitetura do *Perceptron* para o projeto prático

Utilizando o algoritmo supervisionado de Hebb (regra de Hebb) para classificação de padrões, e assumindo-se a taxa de aprendizagem como 0,01, faça as seguintes atividades:

- 1) Execute cinco treinamentos para a rede *Perceptron*, iniciando-se o vetor de pesos  $\{w\}$  em cada treinamento com valores aleatórios entre zero e um. Se for o caso, reinicie o gerador de números aleatórios em cada treinamento de tal forma que os elementos do vetor de pesos iniciais não sejam os mesmos. O conjunto de treinamento encontra-se no apêndice I.

2) Registre os resultados dos cinco treinamentos na tabela 3.2 apresentada a seguir.

Tabela 3.2 – Resultados dos treinamentos do *Perceptron*

Treinamento:	Vetor de pesos iniciais				Vetor de pesos finais				Número de épocas
	$w_0$	$w_1$	$w_2$	$w_3$	$w_0$	$w_1$	$w_2$	$w_3$	
1º(T1)									
2º(T2)									
3º(T3)									
4º(T4)									
5º(T5)									

3) Após o treinamento do *Perceptron*, coloque o mesmo em operação, aplicando-o na classificação automática das amostras de óleo da tabela 3.3, indicando ainda nesta tabela aqueles resultados das saídas (Classes) referentes aos cinco processos de treinamento realizados no item 1.

Tabela 3.3 – Amostras de óleo para validar a rede *Perceptron*

Amostra	$x_1$	$x_2$	$x_3$	$y$ (T1)	$y$ (T2)	$y$ (T3)	$y$ (T4)	$y$ (T5)
1	-0,3665	0,0620	5,9891					
2	-0,7842	1,1267	5,5912					
3	0,3012	0,5611	5,8234					
4	0,7757	1,0648	8,0677					
5	0,1570	0,8028	6,3040					
6	-0,7014	1,0316	3,6005					
7	0,3748	0,1536	6,1537					
8	-0,6920	0,9404	4,4058					
9	-1,3970	0,7141	4,9263					
10	-1,6842	-0,2805	1,2548					

4) Explique por que o número de épocas de treinamento, em relação a esta aplicação, varia a cada vez que executarmos o treinamento do *Perceptron*.

5) Para a aplicação em questão, discorra se é possível afirmar se as suas classes são linearmente separáveis.



Frank Rosenblatt

## Rede Adaline e regra Delta

### 4.1 – Introdução

O *Adaline* foi idealizado por Widrow e Hoff em 1960, e sua principal aplicação se destinava a sistemas de chaveamento de circuitos telefônicos, sendo esta uma das primeiras aplicações industriais que envolveram efetivamente a utilização das redes neurais artificiais [Widrow & Hoff, 1960].

Embora seja também um tipo de rede bem simples, o *Adaline* promoveu alguns outros avanços que foram de essencial importância para o progresso da área de redes neurais artificiais. Dentre essas contribuições, pode-se então mencionar:

- Desenvolvimento do algoritmo de aprendizado regra Delta;
- Aplicações em diversos problemas práticos envolvendo processamento de sinais analógicos;
- Primeiras aplicações industriais de redes neurais artificiais.

Mais especificamente, a grande contribuição do *Adaline* foi a introdução do algoritmo de aprendizado denominado regra Delta, pois este é considerado o precursor da regra Delta generalizada que é utilizada para treinamento de redes *Perceptrons* de camadas múltiplas (capítulo 5).

Similarmente à configuração estrutural do *Perceptron*, o *Adaline* é também constituído de apenas uma camada neural, sendo composto ainda por somente um único neurônio artificial. A composição de vários *Adalines*, constituindo uma rede única, é denominada de *Madaline* [Widrow & Winter, 1988].

A figura 4.1 ilustra uma rede *Adaline* constituída de  $n$  sinais de entrada e somente uma saída, ressaltando-se que esta é também composta de um único neurônio.

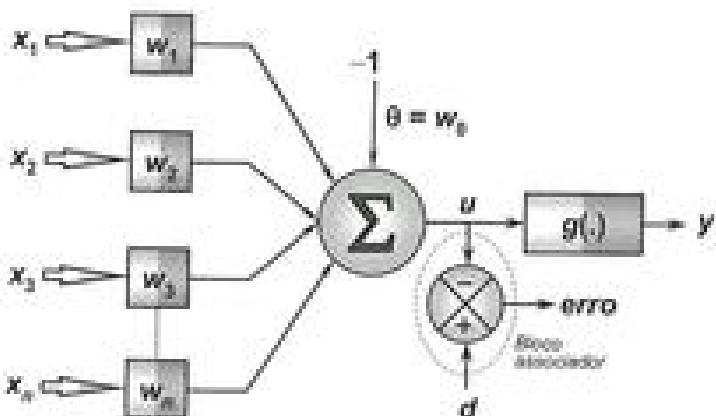


Figura 4.1 – Ilustração da rede *Adaline*

De acordo com a nomenclatura definida no capítulo 2, a rede *Adaline* possui também arquitetura *feedforward* de camada única, pois o fluxo de informações é realizado sempre adiante, isto é, partindo-se das entradas em direção à saída da rede. Observa-se ainda que inexiste qualquer tipo de realimentação de valores produzidos pelo seu único neurônio.

Assim como o *Perceptron*, a rede *Adaline*, devido ainda à sua simplicidade estrutural, tem sido mais utilizada em problemas de classificação de padrões envolvendo apenas duas classes distintas.

## 4.2 – Princípio de funcionamento do *Adaline*

A partir da análise da figura 4.1, verifica-se que cada uma das entradas ( $x$ ) da rede, representando os sinais advindos externamente de determinada aplicação, será inicialmente ponderada pelos respectivos pesos sinápticos que serão ajustados durante o processo de treinamento do *Adaline*.

Em seguida, como também ocorre na rede *Perceptron*, o *Adaline* computa o potencial de ativação ( $u$ ) executando-se a soma das contribuições advindas das multiplicações de todos os sinais  $x$ , por  $w_i$ , incluindo o próprio valor de seu limiar ( $\theta$ ). O passo final para a produção da saída ( $y$ ) do *Adaline* é a aplicação da função de ativação  $g(u)$ , representada tipicamente pela função degrau (1.3) ou degrau bipolar (1.5).

Conforme se pode observar no parágrafo anterior, a obtenção da saída ( $y$ ) do *Adaline* segue a mesma sequência definida para a rede *Perceptron*, sendo que tal processamento será também fornecido pelas seguintes expressões:

$$u = \sum_{i=1}^n w_i \cdot x_i - \theta \text{ ou } u = \sum_{i=0}^n w_i \cdot x_i \quad (4.1)$$

$$y = g(u) \quad (4.2)$$

onde  $x$ , são os sinais de entrada do *Adaline*,  $w_i$  é o peso sináptico associado à  $i$ -ésima entrada,  $\theta$  é o limiar de ativação,  $g(\cdot)$  é a função de ativação e  $u$  é o potencial de ativação.

Como o *Adaline* é ainda utilizado principalmente em aplicações envolvendo reconhecimento de padrões, sua saída somente será capaz de classificar duas classes, sendo que cada uma estará associada a um dos dois valores possíveis da função de ativação adotada (degrau ou degrau bipolar).

Ainda em relação à figura 4.1, observa-se a presença de um bloco associador junto à estrutura do *Adaline*, cuja função é simplesmente auxiliar no processo de treinamento da rede, conforme será detalhado na próxima seção. O sinal de erro obtido pelo bloco é dado por:

$$\text{erro} = d - u \quad (4.3)$$

Em resumo, considerando-se apenas uma amostra de treinamento, o valor da diferença (erro) entre o potencial de ativação ( $u$ ) produzido pela rede e o respectivo valor desejado ( $d$ ) será utilizado para ajustar os pesos ( $w_0, w_1, w_2, \dots, w_n$ ) da rede.

A tabela 4.1 explicita os aspectos característicos dos parâmetros envolvidos com o funcionamento do *Adaline*.

Tabela 4.1 – Aspectos dos parâmetros característicos do *Adaline*

Parâmetro	Variável representativa	Tipo característico
Entradas	$x_i$ ( $i$ -ésima entrada)	Reais ou binárias (advindas externamente)
Pesos sinápticos	$w_i$ (associado a $x_i$ )	Reais (iniciados aleatoriamente)
Límiar	$\theta$	Real (iniciado aleatoriamente)
Saída	$y$	Binária
Função de ativação	$g(\cdot)$	Degrado ou degrado bipolar
Processo de treinamento	-----	Supervisionado
Regra de aprendizado	-----	Regra Delta

Pela inspeção da tabela 4.1, constata-se, portanto, que uma das diferenças principais entre o *Adaline* e o *Perceptron* está principalmente na regra de aprendizado utilizada para os ajustes de pesos e límiar.

A análise matemática efetuada para demonstrar as condições de convergência do *Perceptron* pode ser também aplicada à rede *Adaline*. Em suma, as classes do problema a ser mapeado devem ser linearmente separáveis para que haja a sua completa identificação.

### 4.3 – Processo de treinamento do *Adaline*

O processo de ajuste dos pesos e límiar do *Adaline* é baseado no algoritmo de aprendizado denominado de regra Delta [Widrow & Hoff, 1960] ou regra de aprendizado de Widrow-Hoff, conhecido também como algoritmo *LMS* (*least mean square*) ou método do Gradiente Descendente.

Assumindo-se a disponibilidade de  $p$  amostras de treinamento, a ideia básica envolvida com a aplicação da regra Delta objetivando ajustar os pesos e límiar do neurônio está em minimizar a diferença entre a saída desejada  $\{d\}$  e a resposta do combinador linear  $\{y\}$ , considerando-se para tanto todas as  $p$  amostras.

Mais especificamente, utiliza-se a minimização do erro quadrático entre  $a$  e  $d$  com o intuito de ajustar o vetor de pesos  $\mathbf{w} = [0 \ w_1 \ w_2 \ ... \ w_n]^T$  da rede. Em resumo, o objetivo consiste em obter um  $\mathbf{w}^*$  ótimo tal que o erro quadrático  $\{E(\mathbf{w}^*)\}$  sobre todo o conjunto de amostras seja o mínimo possível. Em termos matemáticos, considerando uma configuração ótima de pesos, tem-se:

$$E(\mathbf{w}^*) \leq E(\mathbf{w}), \quad \text{para } \forall \mathbf{w} \in \mathbb{R}^{n+1} \quad (4.4)$$

A função erro quadrático em relação às  $p$  amostras de treinamento é definida por:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^p (d^{(k)} - a)^2 \quad (4.5)$$

Substituindo o resultado de (4.1) em (4.5), obtém-se:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^p (d^{(k)} - (\sum_{i=1}^n w_i \cdot x_i^{(k)} - \theta))^2 \quad (4.6)$$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^p (d^{(k)} - (\mathbf{w}^T \cdot \mathbf{x}^{(k)} - \theta))^2 \quad (4.7)$$

Assim, a expressão (4.7) totaliza a composição do erro quadrático médio contabilizando-se os  $p$  padrões de treinamento disponibilizados para o processo de aprendizagem do Adaline.

O próximo passo consiste na aplicação do operador gradiente em relação ao vetor  $\mathbf{w}$ , tendo-se como objetivo a busca do valor ótimo para o erro quadrático médio dado em (4.7), ou seja:

$$\nabla E(\mathbf{w}) = \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} \quad (4.8)$$

Considerando o resultado de (4.7) em (4.8), obtém-se:

$$\nabla E(\mathbf{w}) = \sum_{k=1}^p (d^{(k)} - (\mathbf{w}^T \cdot \mathbf{x}^{(k)} - \theta)) \cdot (-\mathbf{x}^{(k)}) \quad (4.9)$$

Retornando o valor de (4.1) em (4.9), tem-se:

$$\nabla E(\mathbf{w}) = -\sum_{k=1}^p (\sigma^{(k)} - u) \cdot \mathbf{x}^{(k)} \quad (4.10)$$

Finalmente, os passos de adaptação do vetor de pesos devem ser efetuados na direção oposta àquela do gradiente, pois o objetivo da otimização é minimizar o erro quadrático médio. Nesta condição, a variação  $\Delta \mathbf{w}$  a ser efectuada no vetor de pesos do Adaline é dada por:

$$\Delta \mathbf{w} = -\eta \cdot \nabla E(\mathbf{w}) \quad (4.11)$$

Inserindo o resultado de (4.10) em (4.11), obtém-se:

$$\Delta \mathbf{w} = \eta \cdot \sum_{k=1}^p (\sigma^{(k)} - u) \cdot \mathbf{x}^{(k)} \quad (4.12)$$

De forma complementar, pode-se expressar (4.12) por:

$$\mathbf{w}^{atual} = \mathbf{w}^{anterior} + \eta \cdot \sum_{k=1}^p (\sigma^{(k)} - u) \cdot \mathbf{x}^{(k)} \quad (4.13)$$

Alternativamente, por razões de simplificação, a atualização de  $\mathbf{w}$  pode ser também realizada discretamente após a apresentação de cada  $k$ -ésimo padrão de treinamento, ou seja:

$$\mathbf{w}^{atual} = \mathbf{w}^{anterior} + \eta \cdot (\sigma^{(k)} - u) \cdot \mathbf{x}^{(k)}, \text{ onde } k = 1, \dots, p \quad (4.14)$$

Em notação algorítmica, a expressão anterior equivale a:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot (\sigma^{(k)} - u) \cdot \mathbf{x}^{(k)} \quad (4.15)$$

onde:  $\mathbf{w} = [0 \ w_1 \ w_2 \dots \ w_n]^T$  é o vetor contendo o limiar e os pesos;

$\mathbf{x}^{(k)} = [-1 \ x_1^{(k)} \ x_2^{(k)} \dots \ x_n^{(k)}]^T$  é a  $k$ -ésima amostra de treinamento;

$d^{(k)}$  é o valor desejado para a  $k$ -ésima amostra de treinamento;  $v$  é o valor de saída do combinador linear; e  $\eta$  é uma constante que define a taxa de aprendizagem da rede.

Assim como no Perceptron, a taxa de aprendizagem  $\{\eta\}$  exprime o quanto rápido o processo de treinamento da rede estará rumando em direção ao ponto de minimização da função erro quadrático médio dada em (4.5), sendo que normalmente se adotam valores pertencentes ao intervalo compreendido em  $0 < \eta < 1$ . A expressão (4.15) é também similar àquela apresentada em (3.8), pois, no caso do Perceptron, o resultado produzido pela própria saída  $\{y\}$  do neurônio será utilizado na regra de ajuste de seus parâmetros internos.

Para elucidar o processo de convergência do Adaline, apresenta-se na figura 4.2 uma ilustração contendo a interpretação geométrica referente aos passos de atualização do vetor  $w$  rumo ao ponto de minimização  $w^*$  da função erro quadrático médio  $\{E(w)\}$ .

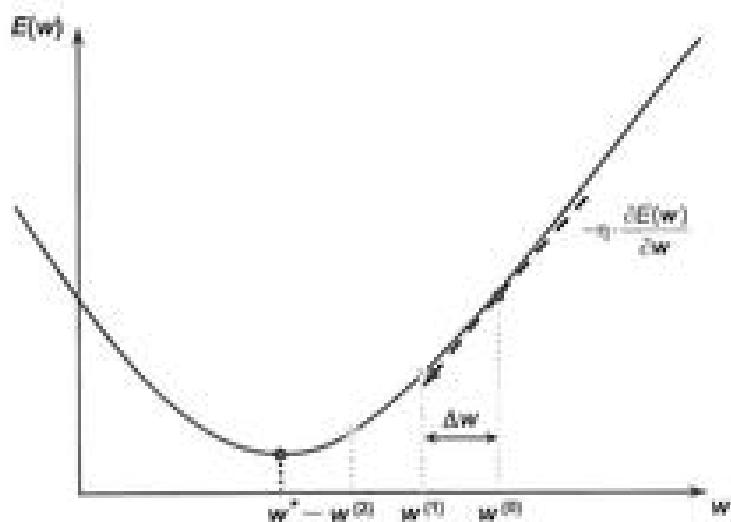


Figura 4.2 – Interpretação geométrica da regra Delta

Assim, a partir da figura 4.2 se observa que, partindo-se de um valor inicial ( $\mathbf{w}^{(0)}$ ), o próximo valor de  $\mathbf{w}$  (representado por  $\mathbf{w}^{(1)}$ ) será obtido considerando-se a direção oposta ao vetor gradiente em relação ao valor de  $\mathbf{w}^{(0)}$ . Para o próximo passo de atualização, o ajuste de  $\mathbf{w}$  (representado agora por  $\mathbf{w}^{(2)}$ ) será realizado considerando-se o valor do gradiente em relação à  $\mathbf{w}^{(1)}$ .

Aplicando-se tais passos sucessivamente, o processo de convergência estará então rumando de forma iterativa em direção ao valor de  $\mathbf{w}^*$ , sendo este a configuração ótima para os parâmetros internos do *Adaline*. Após o processo ter收敛ido para  $\mathbf{w}^*$ , o valor de  $E(\mathbf{w}^*)$  será então sempre menor que quaisquer  $E(\mathbf{w})$  calculados nos passos anteriores.

A sequência passo a passo para o treinamento do *Adaline*, considerando a mesma representação matricial utilizada no treinamento do *Perceptron* (seção 3.4), é explicitada em termos de pseudocódigo conforme se segue. O critério de parada é estipulado em função do erro quadrático médio ( $E_{\text{qm}}(\mathbf{w})$ ) em relação a todas as amostras de treinamento, sendo definido por:

$$E_{\text{qm}}(\mathbf{w}) = \frac{1}{p} \sum_{k=1}^p (d^{(k)} - u)^2 \quad (4.16)$$

O algoritmo converge quando o erro quadrático médio entre duas épocas sucessivas for suficientemente pequeno, ou seja:

$$|E_{\text{qm}}(\mathbf{w}^{\text{atual}}) - E_{\text{qm}}(\mathbf{w}^{\text{anterior}})| \leq \epsilon \quad (4.17)$$

onde  $\epsilon$  é a precisão requerida para o processo de convergência, sendo especificado em função do tipo de aplicação a ser mapeada pela rede *Adaline*.

**Início {Algoritmo Adaline – Fase de Treinamento}**

<1> Obter o conjunto de amostras de treinamento ( $x^{(k)}$ );  
 <2> Associar a saída desejada ( $d^{(k)}$ ) para cada amostra obtida;  
 <3> Iniciar o vetor  $w$  com valores aleatórios pequenos;  
 <4> Especificar taxa de aprendizagem ( $\eta$ ) e precisão requerida ( $\epsilon$ );  
 <5> Iniciar o contador de número de épocas (época  $\leftarrow 0$ );  
 <6> Repetir as instruções:  
 {<6.1>  $E_{qm}^{\text{anterior}} \leftarrow E_{qm}(w)$ ;  
 {<6.2> Para todas as amostras de treinamento ( $x^{(k)}, d^{(k)}$ ), fazer:  
 {<6.2.1>  $u \leftarrow w^T \cdot x^{(k)}$ ;  
 {<6.2.2>  $w \leftarrow w + \eta \cdot (d^{(k)} - u) \cdot x^{(k)}$ ;  
 <6.3> época  $\leftarrow$  época + 1;  
 <6.4>  $E_{qm}^{\text{atual}} \leftarrow E_{qm}(w)$ ;  
 Até que:  $|E_{qm}^{\text{atual}} - E_{qm}^{\text{anterior}}| \leq \epsilon$

**Fim {Algoritmo Adaline – Fase de Treinamento}**

Assim como definido para o treinamento do *Perceptron*, verifica-se que a variável época estará incumbida de contabilizar o próprio número de épocas de treinamento, ou seja, quantas vezes serão necessárias apresentar todas as amostras do conjunto de treinamento visando o ajuste do vetor de pesos ( $w$ ). A rede *Adaline* será considerada treinada (ajustada) quando o erro quadrático médio entre duas épocas sucessivas for inferior à precisão ( $\epsilon$ ) requerida ao problema mapulado pelo *Adaline*. O algoritmo para obtenção do erro quadrático médio ( $E_{qm}(w)$ ) dado por (4.16) pode ser implementado conforme se segue.

**Início {Algoritmo EQM}**

<1> Obter a quantidade de padrões de treinamento ( $p$ );  
 <2> Iniciar a variável  $E_{qm}$  com valor zero ( $E_{qm} \leftarrow 0$ );  
 <3> Para todas as amostras de treinamento ( $x^{(k)}, d^{(k)}$ ), fazer:  
 {<3.1>  $u \leftarrow w^T \cdot x^{(k)}$ ;  
 {<3.2>  $E_{qm} \leftarrow E_{qm} + (d^{(k)} - u)^2$ ;  
 <4>  $E_{qm} \leftarrow \frac{E_{qm}}{p}$ ;

**Fim {Algoritmo EQM}**

Após a finalização do processo de treinamento, o *Adaline* já estará apto para ser aplicado na tarefa de classificação de padrões frente às novas amostras que serão apresentadas em suas entradas. Os passos algorítmicos para esta fase de operação são apresentados a seguir:

**Início {Algoritmo Adaline -- Fase de Operação}**

- <1> Obter uma amostra a ser classificada ( $x$ );
- <2> Utilizar o vetor  $w$  ajustado durante o treinamento;
- <3> Executar as seguintes instruções:
  - <3.1>  $u \leftarrow w^T \cdot x$ ;
  - <3.2>  $y \leftarrow \text{sign}(u)$ ;
  - <3.3> Se  $y = -1$ 
    - <3.3.1> Então: amostra  $x \in$  {Classe A}
  - <3.4> Se  $y = 1$ 
    - <3.4.1> Então: amostra  $x \in$  {Classe B}

**Fim {Algoritmo Adaline – Fase de Operação}**

Assim, conforme mencionado anteriormente, o processo de treinamento do *Adaline* tende a mover continuamente o seu vetor de pesos com o intuito de minimizar o erro quadrático em relação a todas as amostras disponíveis para o aprendizado.

A fim de ilustrar o processo de treinamento do *Adaline* visando o alcance da fronteira de separabilidade entre as classes, a figura 4.3 apresenta duas situações que mostram a sua convergência rumo à estabilização, considerando-se para fins didáticos apenas duas entradas ( $x_1$  e  $x_2$ ).

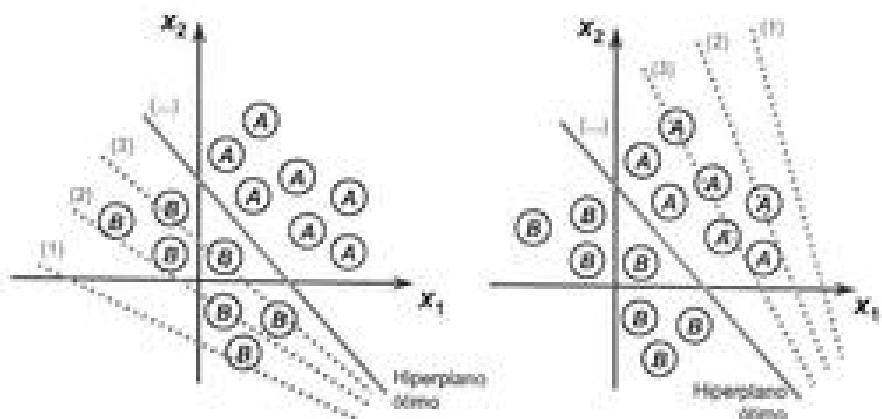


Figura 4.3 – Ilustração do processo de treinamento do *Adaline*

A partir da análise da figura 4.3, observa-se que para as duas situações apresentadas, cujos vetores de pesos iniciais  $w^{(0)}$  foram começados em posições distintas, o processo de convergência da rede encaminha o hiperplano sempre em direção à fronteira de separabilidade ótima, correspondente ao valor de  $w^*$  que minimiza a função erro quadrático (figura 4.2).

Para fins de interpretação, a figura 4.4 mostra o comportamento do erro quadrático médio ( $E_{\text{qr}}$ ) em função do número de épocas de treinamento.

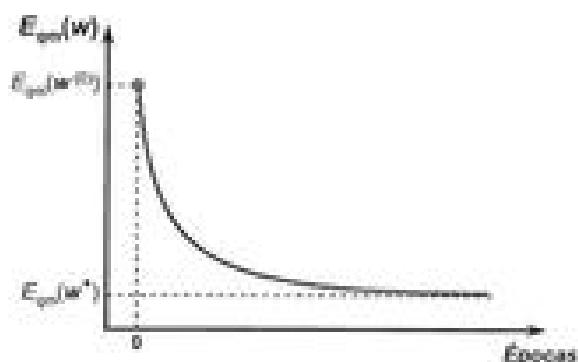


Figura 4.4 – Comportamento do erro quadrático médio em função das épocas de treinamento

Constata-se pela figura 4.4 que a curva do erro quadrático médio para o Adaline é sempre descendente, isto é, a mesma decresce na medida em que as épocas de treinamento vão sendo executadas, estabilizando-se num valor constante quando o ponto de mínimo da função erro quadrático médio (figura 4.2) é alcançado.

#### 4.4 – Comparação entre o processo de treinamento do Adaline e Perceptron

De acordo com a seção anterior, o processo de treinamento visando o ajuste do Adaline é realizado por meio da regra Delta, cujo objetivo consiste em minimizar a diferença entre a saída desejada ( $\sigma$ ) frente à resposta do combinador linear ( $v$ ), considerando para tanto todas as amostras de treinamento disponíveis. Neste caso, independentemente dos valores iniciais atribuídos ao seu vetor de pesos, o hiperplano de separabilidade obtido após a convergência da rede sempre será o mesmo.

Em relação ao processo de treinamento do *Perceptron*, os passos visando o ajuste de seus pesos são realizados por intermédio da regra de Hebb, em que se considera a resposta produzida após a apresentação individual (sinapses locais) de cada amostra de treinamento (seção 3.4). Em tal situação, quaisquer hiperplanos posicionados dentro da faixa de separabilidade entre as classes são considerados soluções apropriadas ao propósito de classificação de padrões pelo *Perceptron*.

Na figura 4.5 são apresentadas duas ilustrações que sintetizam as colocações efetuadas nos dois parágrafos anteriores.

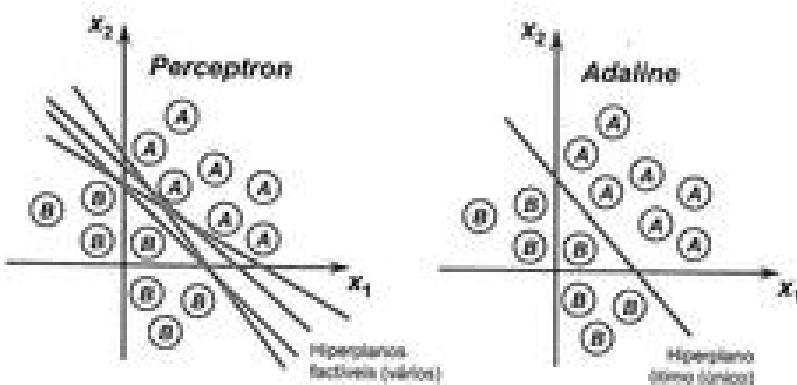


Figura 4.5 – Comparação entre a separação de classes efetuadas pelo *Perceptron* e pelo *Adaline*

A partir desta figura, observa-se que, para o caso do *Perceptron*, o hiperplano que separa as classes pode ter infinitas disposições, pois a configuração final de seu vetor de pesos é fortemente dependente dos valores iniciais que foram aleatoriamente alocados.

Em relação ao *Adaline*, a inclinação do hiperplano é ajustada por intermédio do método dos mínimos quadrados dos erros (*LMS* – *least mean square*), sendo este a própria essência da regra Delta. Portanto, independentemente dos valores iniciais atribuídos para o seu vetor de pesos, a configuração final do hiperplano sempre será a mesma. Tal procedimento faz do *Adaline* uma rede neural com maior tendência de imunidade a eventuais ruidos que podem afetar o processo a ser mapeado.

Para elucidar esta habilidade, ilustra-se na figura 4.6 uma circunstância em que uma nova amostra contendo certa parcela de ruídos será apresentada a ambas as redes, com o propósito de reconhecer em qual das classes será identificada.

Para tanto, considera-se que a amostra possa ter sido afetada por um ruído aditivo atuando sobre a vizinhança representada pela circunferência tracejada.

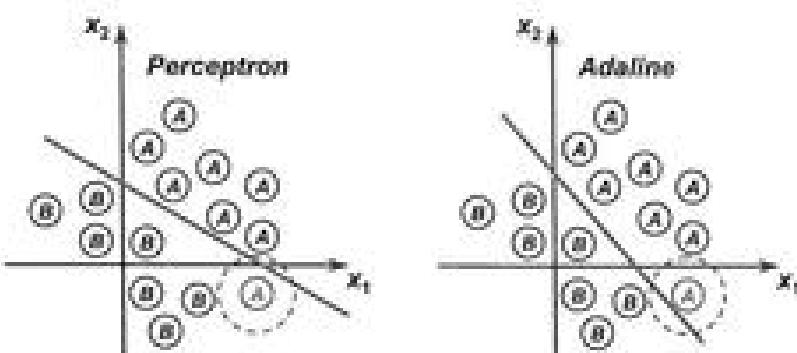


Figura 4.6 – Ilustração da aplicação do *Perceptron* e *Adaline* considerando uma amostra contendo ruídos.

Conforme se pode abstrair pela análise da figura 4.6, considerando-se a condição ruídos presente, a rede *Perceptron* apresentará uma probabilidade maior de classificar incorretamente a amostra; ao passo que a rede *Adaline* terá maiores chances de classificá-la corretamente, pois o seu hiperplano estará posicionado na fronteira de decisão ótima do ponto de vista dos mínimos quadrados dos erros.

Assim, considerando os aspectos envolvidos com o processo de treinamento do *Adaline*, apresentam-se na sequência algumas notas práticas referentes à sua convergência.

- Diferentemente do *Perceptron*, mesmo se as classes envolvidas com o problema a ser mapeado pelo *Adaline* forem não-linearmente separáveis, a regra Delta é passível de convergência em função do patamar de precisão atribuído para medir a diferença dos erros quadráticos médios entre duas épocas sucessivas [Reed & Marks II, 1999];
- O valor da taxa de aprendizado ( $\eta$ ) deve ser cuidadosamente especificado a fim de resguardar instabilidades em torno do ponto de mínimo da função erro quadrático, bem como evitar que o processo de convergência fique demasiadamente lento;
- Assim como ocorre no *Perceptron*, a quantidade de épocas necessárias para a convergência do *Adaline* varia em função dos valores iniciais que foram

atribuídos ao seu vetor de pesos ( $w$ ), da disposição espacial das amostras de treinamento e do valor assumido para a taxa de aprendizado ( $\eta$ );

- A disposição final do hiperplano, obtido após a convergência, confere ao *Adaline* uma maior tendência de robustez frente a eventuais ruidos que podem afetar as amostras a serem classificadas;
- O desempenho do treinamento do *Adaline* pode ser melhorado por intermédio da normalização dos sinais de entradas frente a domínios apropriados, conforme será apresentado em detalhes no próximo capítulo.

#### 4.5 – Exercícios

1) Considerando-se que um problema a ser mapeado pelo *Adaline* não seja linearmente separável, explique então se para esta situação o processo de treinamento (por meio do algoritmo regra Delta) também convergirá.

2) Explique por que o treinamento do *Adaline* se processa normalmente de forma mais rápido que aquele do *Perceptron*. Considere que ambas as redes foram aplicadas no mesmo tipo de problema, tendo-se ainda seus vetores de pesos iniciados com valores iguais.

3) Para o exercício anterior, discorra sobre uma eventual estratégia que se poderia utilizar, fazendo-se uso do *Adaline* já treinado, para verificar se as classes do problema mapeado são linearmente separáveis.

4) Explique as principais diferenças existentes entre o *Perceptron* e o *Adaline*.

5) Considerando-se a aplicação de redes neurais artificiais em um processo de classificação de padrões que necessite de treinamento *on-line*, explique que tipo de rede (*Perceptron* ou *Adaline*) seria a mais apropriada para tal condição.

6) Baseando-se no processo de treinamento utilizando a regra Delta, explique as eventuais instabilidades que se podem verificar quando da adoção de valores elevados para a taxa de aprendizado. Explique também se há eventuais inconvenientes quando se assumem valores extremamente pequenos para a taxa de aprendizagem.

7) Considerando-se os procedimentos de derivação do processo de aprendizagem do *Adaline*, explique se seria possível utilizar na expressão (4.5) a saída do neurônio ( $y$ ) ao invés do valor do combinador linear ( $u$ ).

8) Discorra se a afirmação seguinte é verdadeira ou falsa. Independentemente

mente dos valores iniciais assumidos para o vetor de pesos do *Adaline*, uma mesma configuração final para  $\mathbf{w}^*$  será sempre obtida após a sua convergência.

9) Explique, considerando a questão anterior, se o número de épocas de treinamento será também igual, independentemente do seu vetor de pesos iniciais.

10) Em relação ao critério de parada para o processo de convergência do *Adaline*, fornecido em (4.17), discorra se há realmente a necessidade de aplicação do operador módulo sobre a diferença do erro quadrático médio entre duas épocas sucessivas.

#### 4.6 – Projeto prático

Um sistema de gerenciamento autônomo de duas válvulas, situado a 500 metros de um processo industrial, envia um sinal codificado constituído de quatro grandezas  $\{x_1, x_2, x_3 \text{ e } x_4\}$ , as quais são necessárias para seus acionamentos. Conforme mostra a figura 4.7, uma mesma via de comunicação é utilizada para acionar ambas as válvulas, sendo que o comutador localizado próximo a estas deve decidir se o sinal é para a válvula A ou B.

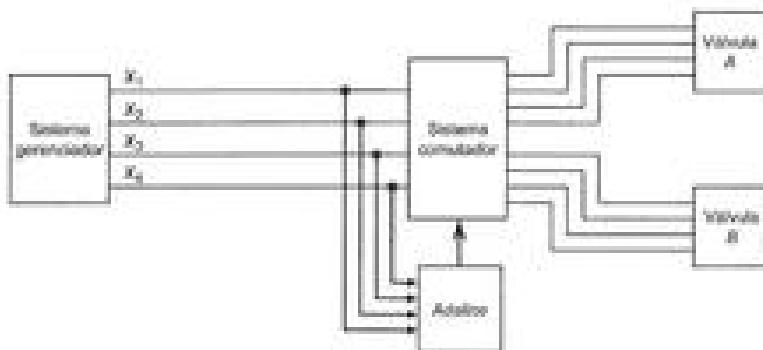


Figura 4.7 – Estrutura esquemática do sistema de acionamento de válvulas

Entretanto, durante a comunicação, os sinais sofrem interferências que alteram o conteúdo das informações originalmente transmitidas. Para contornar este problema, a equipe de engenheiros e cientistas pretende treinar um *Adaline* para classificar os sinais ruidosos, cujo objetivo é então garantir ao sistema comutador se os dados devem ser encaminhados para o comando de ajuste da válvula A ou B.

Assim, fundamentado nas medições de alguns sinais já com ruídos, compilou-se o conjunto de treinamento apresentado no apêndice II, tornando-se por convenção o valor -1 para os sinais que devem ser encaminhados para o ajuste da válvula A e o valor +1 se os mesmos devem ser enviados para a válvula B. Para tanto, a estrutura do *Adaline* é ilustrada na figura seguinte.

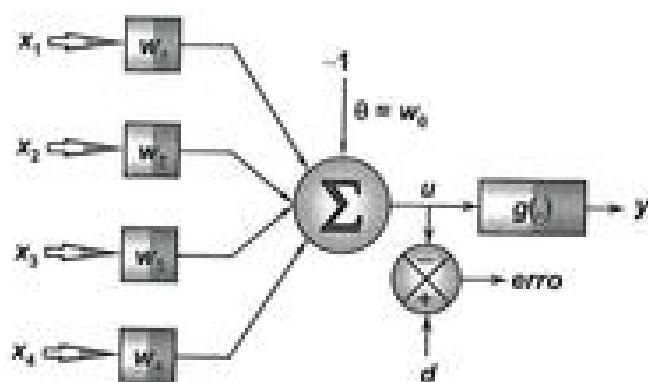


Figura 4.8 – Topologia do *Adaline* aplicada ao projeto prático

Utilizando-se o algoritmo de aprendizado regra Delta visando classificação de padrões pelo *Adaline*, realize as seguintes atividades:

- 1) Execute cinco treinamentos para o *Adaline*, iniciando-se o vetor de pesos em cada treinamento com valores aleatórios entre zero e um. Se for o caso, reinicie o gerador de números aleatórios em cada treinamento de tal forma que os elementos do vetor de pesos iniciais não sejam os mesmos. Utilize um valor de taxa de aprendizado ( $\eta$ ) igual a 0,0025 e uma precisão ( $\epsilon$ ) igual a  $10^{-4}$ . O conjunto de treinamento encontra-se no apêndice II.
- 2) Registre os resultados dos cinco treinamentos na tabela seguinte.

Tabela 4.2 – Resultado dos treinamentos do *Adaline*

Treinamento	Vetor de pesos iniciais					Vetor de pesos finais					Número de épocas
	$w_0$	$w_1$	$w_2$	$w_3$	$w_4$	$w_0$	$w_1$	$w_2$	$w_3$	$w_4$	
1º(T1)											
2º(T2)											
3º(T3)											
4º(T4)											
5º(T5)											

3) Trace para os dois primeiros treinamentos realizados os respectivos gráficos dos valores de erro quadrático médio em função de cada época de treinamento, analisando também o comportamento de ambos. Discorra ainda se as classes envolvidas com o problema podem ser consideradas linearmente separáveis.

4) Para todos os treinamentos realizados anteriormente, aplique então ao *Adaline* (já ajustado) os sinais registrados na tabela 4.3, visando-se classificar (indicar ao comutador) se os referidos sinais devem ser encaminhados para a válvula A ou B.

Tabela 4.3 – Amostras de sinais para classificação pelo *Adaline*

Amostra	$x_1$	$x_2$	$x_3$	$x_4$	$\hat{y}$ (T1)	$\hat{y}$ (T2)	$\hat{y}$ (T3)	$\hat{y}$ (T4)	$\hat{y}$ (T5)
1	0,9694	0,6909	0,4334	3,4965					
2	0,5427	1,3832	0,6390	4,0352					
3	0,6081	-0,9196	0,5925	0,1016					
4	-0,1618	0,4694	0,2030	3,0117					
5	0,1870	-0,2578	0,6124	1,7749					
6	0,4891	-0,5276	0,4378	0,6439					
7	0,3777	2,0149	0,7423	3,3932					
8	1,1498	-0,4067	0,2469	1,5666					
9	0,9325	1,0950	1,0399	3,3591					
10	0,5060	1,3317	0,9222	3,7174					
11	0,0497	-2,0656	0,6124	-0,6585					
12	0,4004	3,5389	0,9766	5,3632					
13	-0,1874	1,3343	0,5374	3,2189					
14	0,5060	1,3317	0,9222	3,7174					
15	1,6375	-0,7911	0,7537	0,5616					

5) Embora os números de épocas de cada treinamento realizado no Item 2 possam ser diferentes, explique por que os valores dos pesos continuam praticamente inalterados.



Warren Sturgis McCulloch

## ***Redes Perceptron multicamadas***

### **5.1 – Introdução**

As redes *Perceptrons* de múltiplas camadas (PMC) são caracterizadas pela presença de pelo menos uma camada intermediária (escondida) de neurônios, situada entre a camada de entrada e a respectiva camada neural de saída. Consequentemente, as redes PMC possuem no mínimo duas camadas de neurônios, os quais estarão distribuídos entre as camadas intermediárias e a camada de saída.

As redes PMC são ainda caracterizadas pelas elevadas possibilidades de aplicações em diversos tipos de problemas relacionados com as mais diferentes áreas do conhecimento, sendo também consideradas uma das arquiteturas mais versáteis quanto à aplicabilidade. Entre essas potenciais áreas, têm-se os seguintes destaques:

- Aproximação universal de funções;
- Reconhecimento de padrões;
- Identificação e controle de processos;
- Previsão de séries temporais;
- Otimização de sistemas.

De acordo com a classificação exposta no capítulo 2, a rede PMC pertence, portanto, à arquitetura *feedforward* de camadas múltiplas, cujo treinamento é efetuado de forma supervisionada. Conforme observado na figura 5.1, o fluxo de informações na estrutura da rede se inicia na camada de entrada, percorre em seguida as camadas intermediárias, sendo então finalizado na camada neural de saída. Observa-se ainda que na rede PMC convencional inexiste qualquer tipo de realimentação de valores produzidos pela camada neural de saídas ou pelas próprias camadas neurais intermediárias.

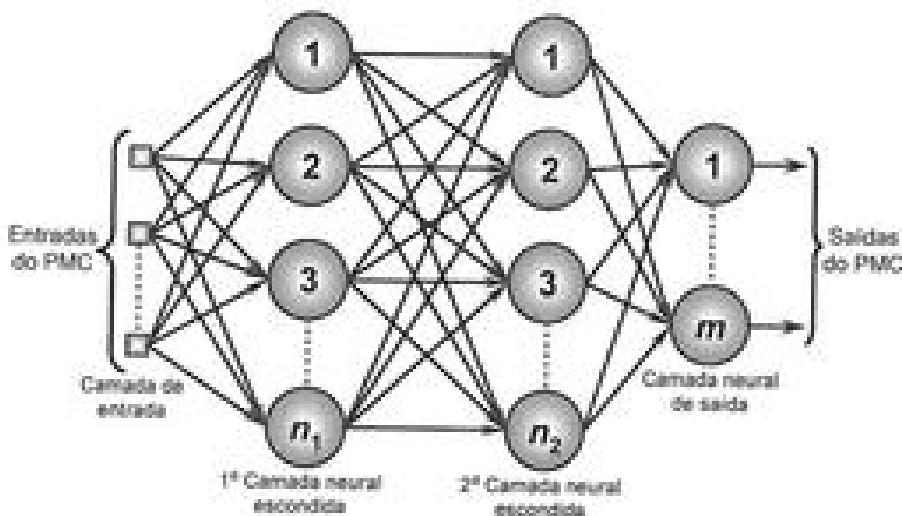


Figura 5.1 – Ilustração de rede *Perceptron* multicamadas

O início da grande popularidade e das extensas aplicabilidades das redes PMC se deram a partir do fim dos anos 1980, sendo atribuídos tais efeitos principalmente à publicação do livro *Parallel/Distributed Processing* [Rumelhart *et al.*, 1986], no qual foi consistentemente explicitado o algoritmo de aprendizagem denominado *backpropagation*, permitindo a sua implementação no processo de treinamento dessas redes.

## 5.2 – Princípio de funcionamento do Perceptron multicamadas

Por intermédio da figura 5.1, observa-se que cada uma das entradas da rede, representando os sinais advindos de determinada aplicação, será propagada uma a uma em direção à camada neural de saída do PMC. Neste caso, as saídas dos neurônios da primeira camada neural escondida serão as próprias

entradas daqueles neurônios pertencentes à segunda camada neural escondida. Para a situação da rede ilustrada na figura 5.1, as saídas dos neurônios da segunda camada neural escondida serão as respectivas entradas dos neurônios pertencentes à sua camada neural de saída.

Assim, a propagação dos sinais de entradas da rede PMC, independentemente da quantidade de camadas intermediárias, é sempre realizada num único sentido, ou seja, da camada de entrada em direção à camada neural de saída.

Diferentemente da rede *Perceptron* simples e da rede *Adaline*, além da presença de camadas escondidas na topologia do PMC, observa-se também, por intermédio da ilustração apresentada na figura 5.1, que a camada neural de saída pode ser composta por diversos neurônios, sendo que cada um destes representaria uma das saídas do processo a ser mapeado. Assim, se tal processo consistisse de  $m$  saídas, a rede PMC teria também  $m$  neurônios em sua última camada neural.

Em resumo, em contraste ainda ao *Perceptron* e *Adaline*, em que um único neurônio era responsável pelo mapeamento integral (concentrado) de todo processo, o conhecimento relacionado ao comportamento entrada/saída do sistema será distribuído por todos os neurônios constituintes do PMC. Os estímulos ou sinais são apresentados à rede em sua camada de entrada. As camadas intermediárias, por sua vez, extraem a maioria das informações referentes ao seu comportamento e as codificam por meio dos pesos sinápticos e limiares de seus neurônios, formando assim uma representação própria do ambiente em que está inserido o referido sistema a ser tratado. Finalmente, os neurônios da camada de saída recebem os estímulos advindos dos neurônios da última camada intermediária, produzindo um padrão de resposta que será a saída disponibilizada pela rede.

A especificação da configuração topológica de uma rede PMC, tais como a quantidade de camadas intermediárias e seus respectivos números de neurônios, depende de diversos fatores que serão pautados ao longo deste capítulo. Mais especificamente, a classe de problema a ser tratada pelo PMC, a disposição espacial das amostras de treinamento e os valores iniciais atribuídos tanto aos parâmetros de treinamento como às matrizes de pesos são elementos que auxiliam na definição de sua topologia.

Conforme mencionado anteriormente, o ajuste dos pesos e do limiar de cada um dos neurônios da rede PMC é efetuado utilizando-se o processo de treinamento supervisionado, isto é, para cada amostra dos dados de entrada obtém-se a respectiva saída (resposta) desejada.

O algoritmo de aprendizado aplicado no decorrer do processo de treinamento de redes PNC é denominado *backpropagation* ou algoritmo de retro-propagação do erro.

### 5.3 – Processo de treinamento do Perceptron multicamadas

O processo de treinamento de redes PNC utilizando o algoritmo *backpropagation*, conhecido também como regra Delta generalizada, é comumente realizado mediante as aplicações sucessivas de duas fases bem específicas. A ilustração de tais fases é mostrada na figura 5.2, em que é considerada uma configuração de PNC constituída de duas camadas escondidas, composta de  $n$  sinais em sua camada de entrada, tendo ainda  $n_1$  neurônios na primeira camada neural escondida,  $n_2$  neurônios na segunda camada neural escondida e  $n_3$  sinais associados à camada neural de saída (terceira camada neural).

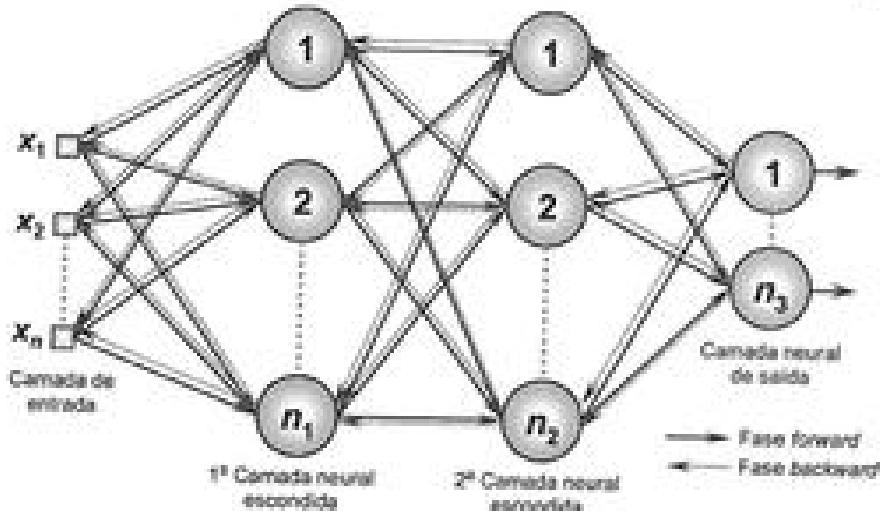


Figura 5.2 – Ilustração das duas fases de treinamento da rede PNC.

A primeira fase a ser aplicada é denominada de "propagação adiante" (*forward*), na qual os sinais  $\{x_1, x_2, \dots, x_n\}$  de uma amostra do conjunto de treinamento são inseridos nas entradas da rede e são propagados camada a camada até a produção das respectivas saídas. Portanto, a aplicação desta fase visa não somente obter as respostas da rede, levando-se em consideração apenas valores atuais de pesos sinápticos e limiares de seus neurônios, os quais permanecerão inalterados durante cada execução desta fase.

Logo em seguida, as respostas produzidas pelas saídas da rede são com-

paradas com as respectivas respostas desejadas que estejam disponíveis, pois, conforme mencionado anteriormente, trata-se de um processo de aprendizado supervisionado. Para ressaltar, considerando-se uma rede PMC constituída de  $n_j$  neurônios em sua camada de saída, os respectivos  $\delta_j$ , desvios (erros) entre as respostas desejadas e aquelas produzidas pelos neurônios de saída são então calculados (subseção 5.3.1), os quais serão subsequentemente utilizados para ajustar os pesos e limiares de todos os seus neurônios.

Assim, em função desses valores de erros, aplica-se, em seguida, a segunda fase do método *backpropagation*, denominada de "propagação reversa" (*backward*). Diferentemente da anterior, as alterações (ajustes) dos pesos sinápticos e limiares de todos os neurônios da rede são executadas no decorrer desta fase.

Em suma, as aplicações sucessivas das fases *forward* e *backward* fazem com que os pesos sinápticos e limiares dos neurônios se ajustem automaticamente em cada iteração, implicando-se na gradativa diminuição da soma dos erros produzidos pelas respostas da rede frente àquelas desejadas.

### 5.3.1 – Derivação do algoritmo *backpropagation*

Para um melhor entendimento do princípio de funcionamento envolvido com o algoritmo *backpropagation*, há a necessidade de se definir *a priori* diversas variáveis e parâmetros auxiliares que serão usados para tal propósito. Baseando-se na topologia de PMC ilustrada na figura 5.2, apresenta-se na figura 5.3 um conjunto de variáveis que norteariam a derivação do algoritmo.

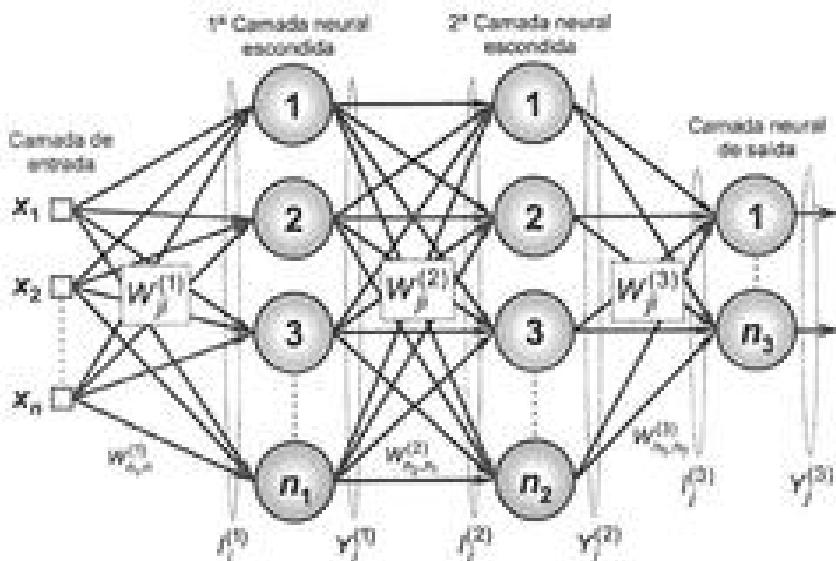


Figura 5.3 – Notação para derivação do algoritmo *backpropagation*

Cada um dos neurônios ( $j$ ) pertencentes a uma das camadas ( $L$ ) da topologia ilustrada na figura 5.3 pode ser configurado conforme a terminologia adotada na figura 5.4, onde  $g(\cdot)$  representa uma função de ativação que deve ser contínua e diferenciável em todo o seu domínio, tais como aquelas representadas pela função de ativação logística ou tangente hiperbólica.

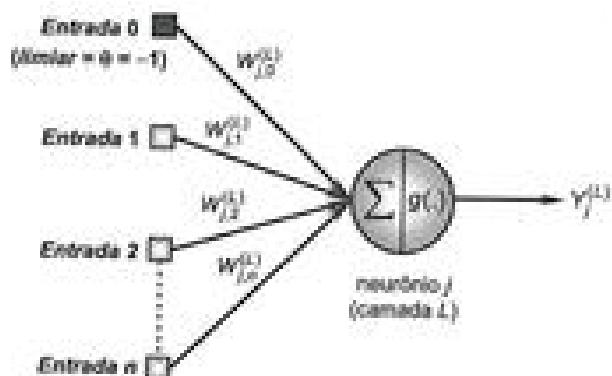


Figura 5.4 – Configuração de neurônio utilizado na derivação do algoritmo *backpropagation*

A partir das figuras 5.3 e 5.4, assume-se então a seguinte terminologia para os seus parâmetros constituintes:

- $W_j^{(l)}$  são matrizes de pesos cujos elementos denotam o valor do peso sináptico conectando o  $j$ -ésimo neurônio da camada ( $L$ ) ao  $i$ -ésimo neurônio da camada ( $L-1$ ). Para a topologia ilustrada na figura 5.3, tem-se:
  - $W_j^{(2)}$  é o peso sináptico conectando o  $j$ -ésimo neurônio da camada de saída ao  $i$ -ésimo neurônio da camada 2;
  - $W_j^{(1)}$  é o peso sináptico conectando o  $j$ -ésimo neurônio da camada escondida 2 ao  $i$ -ésimo neurônio da camada 1;
  - $W_j^{(0)}$  é o peso sináptico conectando o  $j$ -ésimo neurônio da camada 1 ao  $i$ -ésimo sinal da camada de entrada.
- $I_j^{(l)}$  são vetores cujos elementos denotam a entrada ponderada em relação ao  $j$ -ésimo neurônio da camada  $L$ , os quais são definidos por:

$$I_j^{(l)} = \sum_{i=0}^n W_j^{(l)} \cdot x_i \Leftrightarrow I_j^{(l)} = W_{j,0}^{(l)} \cdot x_0 + W_{j,1}^{(l)} \cdot x_1 + \dots + W_{j,n}^{(l)} \cdot x_n \quad (5.1)$$

$$f_j^{(2)} = \sum_{i=0}^{n_2} W_{j,i}^{(2)} \cdot Y_i^{(1)} \Leftrightarrow f_j^{(2)} = W_{j,0}^{(2)} \cdot Y_0^{(1)} + W_{j,1}^{(2)} \cdot Y_1^{(1)} + \dots + W_{j,n_2}^{(2)} \cdot Y_{n_2}^{(1)} \quad (5.2)$$

$$f_j^{(3)} = \sum_{i=0}^{n_3} W_{j,i}^{(3)} \cdot Y_i^{(2)} \Leftrightarrow f_j^{(3)} = W_{j,0}^{(3)} \cdot Y_0^{(2)} + W_{j,1}^{(3)} \cdot Y_1^{(2)} + \dots + W_{j,n_3}^{(3)} \cdot Y_{n_3}^{(2)} \quad (5.3)$$

\*  $Y_i^{(l)}$  são vetores cujos elementos denotam a saída do  $j$ -ésimo neurônio em relação à camada  $L$ , os quais são definidos por:

$$Y_j^{(l)} = g(f_j^{(l)}) \quad (5.4)$$

$$Y_j^{(2)} = g(f_j^{(2)}) \quad (5.5)$$

$$Y_j^{(3)} = g(f_j^{(3)}) \quad (5.6)$$

Como exemplo desta terminologia adotada, considera-se o PMC representado na figura 5.5, composto de duas entradas  $x_0$  e  $x_1$  ( $n = 2$ ), três neurônios na primeira camada escondida ( $n_1 = 3$ ), dois neurônios na segunda camada escondida ( $n_2 = 2$ ) e um neurônio de saída ( $n_3 = 1$ ). Considera-se também que a tangente hiperbólica será assumida como função de ativação para todos os neurônios.

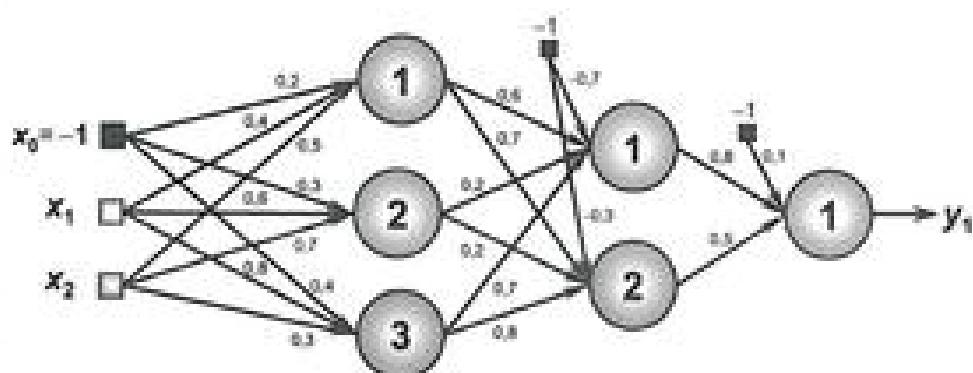


Figura 5.5 – Exemplo de Perceptrons multicamadas

Neste caso, as configurações das matrizes de pesos, considerando-se os valores expostos na figura 5.5, seriam definidas por:

$$W_j^{(1)} = \begin{bmatrix} 0.2 & 0.4 & 0.5 \\ 0.3 & 0.6 & 0.7 \\ 0.4 & 0.8 & 0.3 \end{bmatrix}; W_j^{(2)} = \begin{bmatrix} -0.7 & 0.6 & 0.2 & 0.7 \\ -0.3 & 0.7 & 0.2 & 0.8 \end{bmatrix}; W_j^{(3)} = \begin{bmatrix} 0.1 & 0.6 & 0.5 \end{bmatrix}$$

Assumindo-se um sinal de entrada definido por  $x_0 = 0,3$  e  $x_1 = 0,7$ , os vetores  $I_j^{(0)}$  e  $Y_j^{(0)}$  seriam então representados por:

$$I_j^{(0)} = \begin{bmatrix} I_1^{(0)} \\ I_2^{(0)} \\ I_3^{(0)} \end{bmatrix} = \begin{bmatrix} W_{10}^{(0)} \cdot x_0 + W_{11}^{(0)} \cdot x_1 + W_{12}^{(0)} \cdot x_2 \\ W_{20}^{(0)} \cdot x_0 + W_{21}^{(0)} \cdot x_1 + W_{22}^{(0)} \cdot x_2 \\ W_{30}^{(0)} \cdot x_0 + W_{31}^{(0)} \cdot x_1 + W_{32}^{(0)} \cdot x_2 \end{bmatrix} = \begin{bmatrix} 0.2 \cdot (-1) + 0.4 \cdot 0.3 + 0.5 \cdot 0.7 \\ 0.3 \cdot (-1) + 0.6 \cdot 0.3 + 0.7 \cdot 0.7 \\ 0.4 \cdot (-1) + 0.8 \cdot 0.3 + 0.3 \cdot 0.7 \end{bmatrix} = \begin{bmatrix} 0.27 \\ 0.37 \\ 0.05 \end{bmatrix}$$

$$Y_j^{(0)} = \begin{bmatrix} Y_1^{(0)} \\ Y_2^{(0)} \\ Y_3^{(0)} \end{bmatrix} = \begin{bmatrix} g(I_1^{(0)}) \\ g(I_2^{(0)}) \\ g(I_3^{(0)}) \end{bmatrix} = \begin{bmatrix} \tanh(0,27) \\ \tanh(0,37) \\ \tanh(0,05) \end{bmatrix} = \begin{bmatrix} 0.26 \\ 0.35 \\ 0.06 \end{bmatrix} \xrightarrow{Y_0^{(0)} = 1} Y_j^{(0)} = \begin{bmatrix} Y_0^{(0)} \\ Y_1^{(0)} \\ Y_2^{(0)} \\ Y_3^{(0)} \end{bmatrix} = \begin{bmatrix} 1 \\ 0.26 \\ 0.35 \\ 0.05 \end{bmatrix}$$

onde os argumentos usados na função tangente hiperbólica ( $\tanh$ ) estão em radianos.

Os vetores  $I_j^{(1)}$  e  $Y_j^{(1)}$  referentes à segunda camada neural seriam representados por:

$$I_j^{(1)} = \begin{bmatrix} I_1^{(1)} \\ I_2^{(1)} \end{bmatrix} = \begin{bmatrix} W_{10}^{(1)} \cdot Y_0^{(0)} + W_{11}^{(1)} \cdot Y_1^{(0)} + W_{12}^{(1)} \cdot Y_2^{(0)} + W_{13}^{(1)} \cdot Y_3^{(0)} \\ W_{20}^{(1)} \cdot Y_0^{(0)} + W_{21}^{(1)} \cdot Y_1^{(0)} + W_{22}^{(1)} \cdot Y_2^{(0)} + W_{23}^{(1)} \cdot Y_3^{(0)} \end{bmatrix} = \begin{bmatrix} 0.96 \\ 0.69 \end{bmatrix}$$

$$Y_j^{(1)} = \begin{bmatrix} Y_1^{(1)} \\ Y_2^{(1)} \end{bmatrix} = \begin{bmatrix} g(I_1^{(1)}) \\ g(I_2^{(1)}) \end{bmatrix} = \begin{bmatrix} \tanh(0.96) \\ \tanh(0.69) \end{bmatrix} = \begin{bmatrix} 0.74 \\ 0.63 \end{bmatrix} \xrightarrow{Y_0^{(1)} = 1} Y_j^{(1)} = \begin{bmatrix} Y_0^{(1)} \\ Y_1^{(1)} \\ Y_2^{(1)} \end{bmatrix} = \begin{bmatrix} 1 \\ 0.74 \\ 0.63 \end{bmatrix}$$

Finalmente, os vetores  $I_j^{(2)}$  e  $Y_j^{(2)}$  referentes à terceira camada neural seriam representados por:

$$f_1^{(3)} = [l_1^{(3)}] = [W_{00}^{(3)} \cdot Y_0^{(2)} + W_{10}^{(3)} \cdot Y_1^{(2)} + W_{20}^{(3)} \cdot Y_2^{(2)}] = [0,76]$$

$$Y_1^{(3)} = [0,76] = [g(f_1^{(3)})] = [\tanh(0,76)] = [0,64]$$

Nesta última expressão, dispensa-se a inserção do termo  $Y_0^{(2)} = -1$ , pois já se trata da última camada neural, sendo que o valor de  $Y_i^{(k)}$  é a própria saída  $y_i$  produzida por esta rede.

O próximo passo para o inicio da derivação do algoritmo *backpropagation* consiste em definir a função representativa do erro de aproximação, cuja incumbência será medir o desvio entre as respostas produzidas pelos neurônios de saída da rede em relação aos respectivos valores desejados. Assim, considerando a  $k$ -ésima amostra de treinamento para a topologia ilustrada na figura 5.3, assume-se a função erro quadrático como aquela a ser utilizada para medir o desempenho local associado aos resultados produzidos pelos neurônios de saída frente à referida amostra, ou seja:

$$E(k) = \frac{1}{2} \sum_{j=1}^n (d_j(k) - Y_j^{(k)}(k))^2 \quad (5.7)$$

onde  $Y_j^{(k)}(k)$  é o valor produzido pelo  $j$ -ésimo neurônio de saída da rede considerando-se a  $k$ -ésima amostra de treinamento, enquanto que  $d_j(k)$  é o seu respeitivo valor desejado.

Consequentemente, assumindo um conjunto de treinamento composto por  $p$  amostras, a medição da evolução do desempenho global do algoritmo *backpropagation* pode ser efetuada por meio da avaliação do "erro quadrático médio", definido por:

$$E_M = \frac{1}{p} \sum_{k=1}^p E(k) \quad (5.8)$$

onde  $E(k)$  é o erro quadrático obtido em (5.7).

Visando um melhor entendimento dos passos necessários para a compreensão do algoritmo *backpropagation*, divide-se a sua descrição em duas partes, sendo a primeira destinada ao ajuste da matriz de pesos sinápticos  $W_j^{(k)}$ , referente à camada neural de saída. A segunda parte refere-se aos procedi-

mentos de ajuste das matrizes de pesos associadas às camadas intermediárias que, para a topologia de rede PMC ilustrada na figura 5.3, são constituídas pelas matrizes  $W_j^{(2)}$  e  $W_j^{(3)}$ .

Utilizar-se-á neste livro o processo de aprendizado usando lote de padrões ou *off-line* (subseção 2.3.4) em relação à expressão (5.8), fazendo-se também uso do método baseado no gradiente da função erro quadrático dada em (5.7).

## Parte I – Ajuste dos pesos sinápticos da camada de saída

O objectivo do processo de treinamento para a camada neural de saída consiste de ajustar a matriz de pesos  $W_j^{(3)}$  a fim de minimizar o erro entre a saída produzida pela rede em relação à respectiva saída desejada. Neste caso, considerando-se o erro dado em (5.7) em relação à  $k$ -ésima amostra de treinamento referente ao  $j$ -ésimo neurónio da camada de saída, a regra de ajuste se torna similar àquela aplicada ao *Adaline*. Então, empregando a definição de gradiente e explorando a regra de diferenciação em cadeia, tem-se:

$$\nabla E^{(3)} = \frac{\partial E}{\partial W_j^{(3)}} = \frac{\partial E}{\partial Y_j^{(3)}} \cdot \frac{\partial Y_j^{(3)}}{\partial f_j^{(3)}} \cdot \frac{\partial f_j^{(3)}}{\partial W_j^{(3)}} \quad (5.9)$$

Por intermédio das definições anteriores, tem-se:

$$\frac{\partial f_j^{(3)}}{\partial W_j^{(3)}} = Y_i^{(2)} \quad \{ \text{Obtido a partir de (5.3)} \} \quad (5.10)$$

$$\frac{\partial Y_j^{(3)}}{\partial f_j^{(3)}} = g'(f_j^{(3)}) \quad \{ \text{Obtido a partir de (5.6)} \} \quad (5.11)$$

$$\frac{\partial E}{\partial Y_j^{(3)}} = -(d_i - Y_j^{(3)}) \quad \{ \text{Obtido a partir de (5.7)} \} \quad (5.12)$$

onde  $g'(\cdot)$  denota a derivada de primeira ordem da função de ativação considerada. Substituindo (5.10), (5.11) e (5.12) em (5.9), obtém-se:

$$\frac{\partial E}{\partial W_j^{(3)}} = -(d_j - Y_j^{(3)}) \cdot g'(Y_j^{(3)}) \cdot Y_i^{(2)} \quad (5.13)$$

Logo, o ajuste da matriz de pesos  $W_j^{(3)}$  deve ser efetuado em direção oposta ao gradiente a fim de minimizar o erro, ou seja:

$$\Delta W_j^{(3)} = -\eta \cdot \frac{\partial E}{\partial W_j^{(3)}} \quad \Leftrightarrow \quad \Delta W_j^{(3)} = \eta \cdot \delta_j^{(3)} \cdot Y_i^{(2)} \quad (5.14)$$

onde  $\delta_j^{(3)}$  é definido como o gradiente local em relação ao  $j$ -ésimo neurônio da camada de saída, sendo o mesmo dado por:

$$\delta_j^{(3)} = (d_j - Y_j^{(3)}) \cdot g'(Y_j^{(3)}) \quad (5.15)$$

De forma complementar, a expressão (5.14) pode ser também convertida no seguinte procedimento iterativo:

$$W_j^{(3)}(t+1) = W_j^{(3)}(t) + \eta \cdot \delta_j^{(3)} \cdot Y_i^{(2)} \quad (5.16)$$

onde  $\eta$  é a taxa de aprendizagem do algoritmo *backpropagation*. Em notação algorítmica, esta expressão é equivalente à seguinte:

$$W_j^{(3)} \leftarrow W_j^{(3)} + \eta \cdot \delta_j^{(3)} \cdot Y_i^{(2)} \quad (5.17)$$

Portanto, a expressão (5.17) ajusta os pesos dos neurônios da camada de saída da rede levando-se em consideração a diferença observada entre as respostas produzidas por suas saídas em relação aos seus respectivos valores desejados.

## Parte II – Ajuste dos pesos sinápticos de camadas intermediárias

Diferentemente dos neurônios pertencentes à camada de saída do PMC, para os neurônios das camadas intermediárias não se tem acesso de forma

direta aos valores desejados para as suas saídas. Nesta situação, os ajustes de seus pesos sinápticos são efetuados por intermédio de estimativas dos erros de saída produzidos por aqueles neurônios da camada imediatamente posterior, os quais já foram previamente ajustados.

Como exemplo, seguindo a sequência de ajustes impetrada pela fórmula *backpropagation* para a topologia ilustrada na figura 5.3, somente após os ajustes dos neurônios da camada neural de saída é que se iniciam as correções dos pesos para aqueles da segunda camada intermediária. Nesta condição específica, indispõem-se de valores desejados para as saídas de tais neurônios, sendo então que seus ajustes serão baseados nos valores ponderados daqueles pesos sinápticos que já foram ajustados para os neurônios da camada de saída.

Assim, é justamente neste aspecto que se encontra a essência do algoritmo de retropropagação do erro (*backpropagation*), pois, em primeira instância, tem-se ajustado os pesos sinápticos dos neurônios da camada de saída mediante valores verdadeiros dos desvios observados entre suas respostas produzidas e os respectivos valores desejados. Em segunda instância, retropropaga-se este erro para os neurônios das camadas anteriores, ponderando-se os mesmos pelos valores de pesos sinápticos que já foram previamente ajustados em todas as camadas posteriores. Consequentemente, a resposta desejada para um neurônio de camada escondida deve ser então determinada em função dos neurônios (da camada imediatamente posterior) que estão diretamente conectados a este e que já foram previamente ajustados no passo anterior.

#### (A) Ajuste dos pesos sinápticos da segunda camada escondida

O objetivo do processo de treinamento para a segunda camada neural escondida consiste em ajustar a matriz de pesos  $W^{(2)}$  a fim de minimizar o erro entre a saída produzida pela rede em relação à retropropagação do erro advindo dos ajustes dos neurônios da camada neural de saída. Assim, tem-se:

$$\nabla E^{(2)} = \frac{\partial E}{\partial W_j^{(2)}} = \frac{\partial E}{\partial Y_j^{(2)}} \cdot \frac{\partial Y_j^{(2)}}{\partial I_j^{(2)}} \cdot \frac{\partial I_j^{(2)}}{\partial W_j^{(2)}} \quad (5.18)$$

Por intermédio das definições anteriores, tem-se:

$$\frac{\partial \hat{Y}_j^{(2)}}{\partial W_{ij}^{(2)}} = Y_i^{(1)} \quad \text{(Obtido a partir de (5.2))} \quad (5.19)$$

$$\frac{\partial Y_j^{(2)}}{\partial f_j^{(2)}} = g'(f_j^{(2)}) \quad \text{(Obtido a partir de (5.5))} \quad (5.20)$$

$$\frac{\partial E}{\partial Y_j^{(2)}} = \sum_{k=1}^{n_2} \frac{\partial E}{\partial f_k^{(3)}} \cdot \frac{\partial f_k^{(3)}}{\partial Y_j^{(2)}} = \underbrace{\sum_{k=1}^{n_2} \frac{\partial E}{\partial f_k^{(3)}}}_{\text{Parcela } (i)} \cdot \underbrace{\frac{\partial (\sum_{k=1}^{n_2} W_{kj}^{(3)} \cdot Y_k^{(2)})}{\partial Y_j^{(2)}}}_{\text{Parcela } (ii)} \quad (5.21)$$

onde o valor da derivada parcial do argumento da parcela (i) em relação à  $Y_j^{(2)}$  é o próprio valor de  $W_{kj}^{(3)}$ , ou seja:

$$\frac{\partial E}{\partial Y_j^{(2)}} = \sum_{k=1}^{n_2} \frac{\partial E}{\partial f_k^{(3)}} \cdot \underbrace{\frac{W_{kj}^{(3)}}{\text{parcela } (ii)}}_{\text{parcela } (i)} \quad (5.22)$$

Torna-se importante observar que o valor da parcela (i) da expressão (5.22) refere-se aos pesos sinápticos de todos os neurônios da camada de saída que estão interligados a um determinado neurônio  $j$  da segunda camada intermediária.

Ressalta-se aqui novamente a essência do método *backpropagation*, pois todos os pesos  $W_{ij}^{(2)}$  já foram ajustados no passo anterior com base em valores reais de erro, sendo que estes serão então utilizados para o ajuste dos pesos da segunda camada neural intermediária. Para fins de ilustração, a figura 5.6 mostra essa representação tomando-se como referência o neurônio  $j$  desta camada.

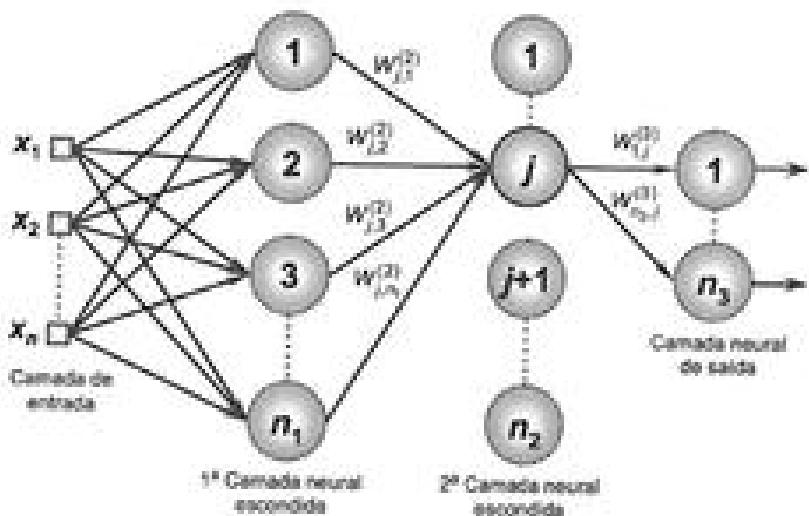


Figura 5.6 – Representação de interligação neural visando correção dos pesos sinápticos de neurônios da segunda camada escondida

Em relação à obtenção de valores para a parcela  $(i)$  da expressão (5.22), observa-se que o seu resultado pode ser fornecido por intermédio da multiplicação de (5.11) por (5.12), a qual resulta no próprio valor absoluto de  $\delta_j^{(2)}$  explicitado em (5.15). Assim, levando-se em consideração tais substituições, a expressão (5.22) pode então ser representada por:

$$\frac{\partial E}{\partial y_j^{(2)}} = - \sum_{k=1}^{n_2} \delta_k^{(3)} \cdot w_{kj}^{(3)} \quad (5.23)$$

Por conseguinte, substituindo (5.19), (5.20) e (5.23) em (5.18), tem-se:

$$\frac{\partial E}{\partial w_{j\ell}^{(2)}} = - \left( \sum_{k=1}^{n_2} \delta_k^{(3)} \cdot w_{kj}^{(3)} \right) \cdot g'(y_j^{(2)}) \cdot y_\ell^{(1)} \quad (5.24)$$

Logo, o ajuste da matriz de pesos  $W_j^{(2)}$  deve ser efetuado em direção oposta ao gradiente a fim de minimizar o erro, ou seja:

$$\Delta w_{j\ell}^{(2)} = -\eta \cdot \frac{\partial E}{\partial w_{j\ell}^{(2)}} \Leftrightarrow \Delta w_{j\ell}^{(2)} = \eta \cdot \delta_j^{(2)} \cdot y_\ell^{(1)} \quad (5.25)$$

onde  $\delta_j^{(2)}$  é definido como o gradiente local em relação ao  $j$ -ésimo neurônio da segunda camada intermediária, isto é:

$$\delta_j^{(2)} = \left( \sum_{k=1}^m \delta_k^{(2)} \cdot w_{kj}^{(2)} \right) \cdot g'(y_j^{(2)}) \quad (5.26)$$

De forma complementar, a expressão (5.25) pode ser também convertida no seguinte procedimento iterativo:

$$w_j^{(2)}(t+1) = w_j^{(2)}(t) + \eta \cdot \delta_j^{(2)} \cdot y_j^{(1)} \quad (5.27)$$

Em notação algorítmica, esta expressão é equivalente à seguinte:

$$w_j^{(2)} \leftarrow w_j^{(2)} + \eta \cdot \delta_j^{(2)} \cdot y_j^{(1)} \quad (5.28)$$

Portanto, a expressão (5.28) ajusta os pesos dos neurônios da segunda camada escondida, levando-se em conta a retropropagação do erro advinda a partir dos neurônios da camada de saída.

### (B) Ajuste dos pesos sinápticos da primeira camada escondida

Em relação à primeira camada escondida, o objetivo do processo de treinamento consiste de ajustar a matriz de pesos  $W^{(1)}$  a fim de minimizar o erro entre a saída produzida pela rede em função da retropropagação do erro advindo dos ajustes dos neurônios da segunda camada escondida. Assim, tem-se:

$$\nabla E^{(1)} = \frac{\partial E}{\partial w_j^{(1)}} = \frac{\partial E}{\partial y_j^{(1)}} \frac{\partial y_j^{(1)}}{\partial r_j^{(1)}} \frac{\partial r_j^{(1)}}{\partial w_j^{(1)}} \quad (5.29)$$

Por intermédio das definições anteriores, tem-se:

$$\frac{\partial r_j^{(1)}}{\partial w_j^{(1)}} = x_i \quad (\text{Obtido a partir de (5.1)}) \quad (5.30)$$

$$\frac{\partial Y_j^{(1)}}{\partial Y_j^{(0)}} = g'(Y_j^{(0)}) \quad \text{[Obtido a partir de (5.4)]} \quad (5.31)$$

$$\frac{\partial E}{\partial Y_j^{(1)}} = \sum_{k=1}^{n_2} \frac{\partial E}{\partial f_k^{(2)}} \cdot \frac{\partial f_k^{(2)}}{\partial Y_j^{(1)}} = \sum_{k=1}^{n_2} \underbrace{\frac{\partial E}{\partial f_k^{(2)}}}_{\text{parcela } (i)} \cdot \underbrace{\frac{\partial \left( \sum_{k=1}^{n_2} W_{kj}^{(2)} \cdot Y_j^{(1)} \right)}{\partial Y_j^{(1)}}}_{\text{parcela } (j)} \quad (5.32)$$

De forma similar à (5.22), o valor da derivada parcial do argumento da parcela (ii) de (5.32) em relação à  $Y_j^{(1)}$  é o próprio valor de  $W_{kj}^{(2)}$ , ou seja:

$$\frac{\partial E}{\partial Y_j^{(1)}} = \sum_{k=1}^{n_2} \underbrace{\frac{\partial E}{\partial f_k^{(2)}}}_{\text{parcela } (i)} \cdot \underbrace{\frac{W_{kj}^{(2)}}{\text{parcela } (j)}}_{(j)} \quad (5.33)$$

A mesma análise realizada para os ajustes dos pesos sinápticos da segunda camada neural intermediária pode também aqui ser utilizada para a primeira camada intermediária. Neste caso, o valor da parcela (ii) de (5.33) se refere aos pesos sinápticos de todos os neurônios da segunda camada intermediária que estão interligados a um determinado neurônio  $j$  da primeira camada intermediária.

Ressalta-se ainda que todos os pesos  $W_{kj}^{(2)}$  já foram ajustados no passo anterior com base em valores retropropagados de erro que tiveram como base o ajuste dos pesos  $W_{kj}^{(1)}$ , os quais por sua vez foram ajustados baseando-se em valores reais de erro. Para fins de ilustração, a figura 5.7 mostra essa representação tomando-se também como referência o neurônio  $j$  da primeira camada escondida.

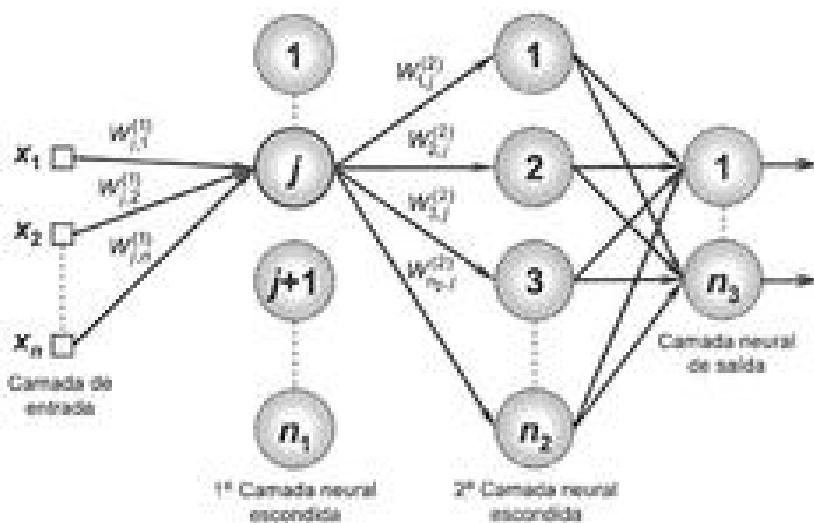


Figura 5.7 – Representação de interligação ocorrida visando correção dos pesos sinápticos de neurônios da primeira camada escondida

Em relação à obtenção de valores para a parcela (i) da expressão (5.33), observa-se que o resultado desta pode ser calculado por intermédio da multiplicação de (5.20) por (5.21), na qual resulta no próprio valor absoluto de  $\delta_j^{(2)}$  explicitado em (5.26). Assim, levando-se em consideração tais substituições, a expressão (5.32) pode então ser representada por:

$$\frac{\partial E}{\partial Y_j^{(1)}} = - \sum_{k=1}^{n_2} \delta_k^{(2)} \cdot W_{kj}^{(2)} \quad (5.34)$$

Por conseguinte, substituindo (5.30), (5.31) e (5.34) em (5.29), tem-se:

$$\frac{\partial E}{\partial W_{ji}^{(1)}} = - \left( \sum_{k=1}^{n_2} \delta_k^{(2)} \cdot W_{kj}^{(2)} \right) \cdot g'(Y_j^{(1)}) \cdot x_i \quad (5.35)$$

Logo, o ajuste da matriz de pesos  $W_{ji}^{(1)}$  deve ser efetuado em direção oposta ao gradiente visando propósitos de minimização do erro, ou seja:

$$\Delta W_{ji}^{(1)} = -\eta \cdot \frac{\partial E}{\partial W_{ji}^{(1)}} \Leftrightarrow \Delta W_{ji}^{(1)} = \eta \cdot \delta_j^{(1)} \cdot x_i \quad (5.36)$$

onde  $\delta_j^{(1)}$  é definido como o gradiente local em relação ao  $j$ -ésimo neurônio da primeira camada intermediária, isto é:

$$\delta_j^{(1)} = \left( \sum_{k=1}^m \delta_k^{(2)} \cdot W_{kj}^{(2)} \right) \cdot g'(t_j^{(1)}) \quad (5.37)$$

De maneira complementar, a expressão (5.36) pode ser também convertida no seguinte procedimento iterativo:

$$W_{ji}^{(1)}(t+1) = W_{ji}^{(1)}(t) + \eta \cdot \delta_j^{(1)} \cdot x_i \quad (5.38)$$

Em notação algorítmica, resume-se em:

$$W_{ji}^{(1)} \leftarrow W_{ji}^{(1)} + \eta \cdot \delta_j^{(1)} \cdot x_i \quad (5.39)$$

Portanto, a expressão (5.39) ajusta os pesos dos neurônios da primeira camada neural intermediária, levando-se em consideração a retropropagação do erro advinda a partir dos neurônios da segunda camada intermediária.

Os procedimentos de ajuste das matrizes de pesos sinápticos apresentados nas expressões anteriores podem ser generalizados para quaisquer topologias de redes PMC, independentemente da quantidade de camadas intermediárias.

### 5.3.2 – Implementação do algoritmo *backpropagation*

Diferentemente do *Perceptron* simples e do *Adaline*, a camada de saída do PMC pode ser constituída por mais de um neurônio, sendo que tal quantidade é especificada em função do número de saídas do sistema a ser mapeado.

Assim, a fim de elucidar a notação utilizada para o vetor  $x^{(k)}$ , representando a  $k$ -ésima amostra de treinamento, assim como para os respectivos valores desejados, que são agora inseridos no vetor  $d^{(k)}$ , assume-se então como exemplo um problema a ser mapeado pelo PMC que seja constituído de três entradas  $\{x_1, x_2, x_3\}$ . Assume-se também que o conjunto de treinamento seja

composto de apenas quatro amostras, formadas pelos seguintes valores de entrada:  $\Omega^{(x)} = \{ [0,2 \ 0,9 \ 0,4]; [0,1 \ 0,3 \ 0,5]; [0,9 \ 0,7 \ 0,8]; [0,6 \ 0,4 \ 0,3] \}$ . Neste caso, considera-se ainda que o PMC esteja resolvendo um problema de aproximação funcional que exige duas variáveis de saída, as quais serão então representadas por dois neurônios pertencentes à sua camada de saída, cujos respectivos valores desejados  $\{d_x, d_y\}$  para cada uma das quatro amostras são fornecidos por:  $\Omega^{(d)} = \{ [0,7 \ 0,3]; [0,6 \ 0,4]; [0,9 \ 0,5]; [0,2 \ 0,8] \}$ . A representação matricial de tais valores pode ser dada por:

$$\Omega^{(x)} = \begin{matrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{matrix} \left[ \begin{array}{cccc} x^{(1)} & x^{(2)} & x^{(3)} & x^{(4)} \\ \hline -1 & -1 & -1 & -1 \\ 0,2 & 0,1 & 0,9 & 0,6 \\ 0,9 & 0,3 & 0,7 & 0,4 \\ 0,4 & 0,5 & 0,8 & 0,3 \end{array} \right]; \quad \Omega^{(d)} = \begin{matrix} d_0 \\ d_1 \end{matrix} \left[ \begin{array}{cccc} d^{(1)} & d^{(2)} & d^{(3)} & d^{(4)} \\ \hline 0,7 & 0,6 & 0,9 & 0,2 \\ 0,3 & 0,4 & 0,5 & 0,8 \end{array} \right]$$

Alternativamente, pode-se extrair dessas matrizes cada um dos vetores  $x^{(i)}$ , com seu respectivo vetor  $d^{(i)}$ , os quais representarão cada uma das amostras de treinamento, isto é:

$$x^{(1)} = [-1 \ 0,2 \ 0,9 \ 0,4]^T; \text{ com } d^{(1)} = [0,7 \ 0,3]^T$$

$$x^{(2)} = [-1 \ 0,1 \ 0,3 \ 0,5]^T; \text{ com } d^{(2)} = [0,6 \ 0,4]^T$$

$$x^{(3)} = [-1 \ 0,9 \ 0,7 \ 0,8]^T; \text{ com } d^{(3)} = [0,9 \ 0,5]^T$$

$$x^{(4)} = [-1 \ 0,6 \ 0,4 \ 0,3]^T; \text{ com } d^{(4)} = [0,2 \ 0,8]^T$$

O critério de parada do processo fica estipulado em função do erro quadrático médio calculado por meio de (5.8), levando-se em consideração todas as amostras de treinamento disponíveis. O algoritmo converge quando o erro quadrático médio entre duas épocas sucessivas for suficientemente pequeno, ou seja:

$$|E_M^{\text{final}} - E_M^{\text{anterior}}| \leq \epsilon \quad (5.40)$$

onde  $\epsilon$  é a precisão requerida para o processo de convergência, sendo especificado em função do tipo de aplicação a ser mapeada pela rede.

A sequência de procedimentos computacionais visando o processo de treinamento de redes PMC é explicitada, em termos de pseudocódigo, conforme se segue:

#### Início {Algoritmo PMC – Fase de Treinamento}

- <1> Obter o conjunto de amostras de treinamento  $\{x^{(k)}\}$ ;
- <2> Associar o vetor de saída desejada ( $d^{(k)}$ ) para cada amostra;
- <3> Iniciar  $W_1^{(1)}$ ,  $W_2^{(2)}$  e  $W_3^{(3)}$  com valores aleatórios pequenos;
- <4> Especificar taxa de aprendizagem ( $\eta$ ) e precisão requerida ( $\epsilon$ );
- <5> Iniciar o contador de número de épocas {época  $\leftarrow 0$ };
- <6> Repetir as instruções:
  - <6.1>  $E_M^{\text{atual}} \leftarrow E_M$ ; {conforme (5.8)}
  - <6.2> Para todas as amostras de treinamento ( $x^{(k)}, d^{(k)}$ ), fazer:
    - <6.2.1> Obter  $f_1^{(1)}$  e  $y_1^{(1)}$ ; {conforme (5.1) e (5.4)}
    - <6.2.2> Obter  $f_2^{(2)}$  e  $y_2^{(2)}$ ; {conforme (5.2) e (5.5)}
    - <6.2.3> Obter  $f_3^{(3)}$  e  $y_3^{(3)}$ ; {conforme (5.3) e (5.6)}
    - <6.2.4> Determinar  $s_1^{(1)}$ ; {conforme (5.15)}
    - <6.2.5> Ajustar  $W_1^{(1)}$ ; {conforme (5.17)}
    - <6.2.6> Determinar  $s_2^{(2)}$ ; {conforme (5.20)}
    - <6.2.7> Ajustar  $W_2^{(2)}$ ; {conforme (5.28)}
    - <6.2.8> Determinar  $s_3^{(3)}$ ; {conforme (5.37)}
    - <6.2.9> Ajustar  $W_3^{(3)}$ ; {conforme (5.39)}
  - <6.3> Obter  $y^{(k)}$  ajustado; {conforme <6.2.1>, <6.2.2> e <6.2.3>}
  - <6.4>  $E_M^{\text{atual}} \leftarrow E_M$ ; {conforme (5.8)}
  - <6.5> época  $\leftarrow$  época + 1;
- Até que:  $|E_M^{\text{atual}} - E_M^{\text{anterior}}| \leq \epsilon$

#### Fim {Algoritmo PMC – Fase de Treinamento}

Após o processo de treinamento do PMC, a variável época conterá o próprio número de épocas que foram necessárias para a efetivação do treinamento da rede, isto é, quantas vezes foram necessárias apresentar todas as amostras do conjunto de treinamento visando o ajuste das matrizes de pesos.

Para tanto, o PMC será considerado totalmente treinado (ajustado) quando o erro quadrático médio ( $E_m$ ) entre duas épocas sucessivas for inferior à precisão ( $\epsilon$ ), requerida ao problema a ser mapeado.

A variável época pode ser também utilizada como critério de parada para o PMC, em situações em que a precisão especificada para o problema se torna inalcançável. Para tal propósito, basta limitar o processo de treinamento quando a quantidade de épocas já tenha alcançado um valor pré-especificado.

Por conseguinte, após o término do treinamento do PMC, pode-se então utilizá-lo para estimar as saídas do sistema que foi mapeado frente às novas amostras que serão apresentadas em suas entradas. Para esta fase de operação, a sequência de passos é apresentada conforme se segue.

#### Início (Algoritmo PMC – Fase de Operação)

- { <1> Obter uma amostra ( $x$ );
- <2> Assumir  $W_j^{(1)}$ ,  $W_j^{(2)}$  e  $W_j^{(3)}$  já ajustadas no treinamento;
- <3> Execute as seguintes instruções:
  - <3.1> Obter  $I_j^{(1)}$  e  $Y_j^{(1)}$ ; (conforme (5.1) e (5.4))
  - <3.2> Obter  $I_j^{(2)}$  e  $Y_j^{(2)}$ ; (conforme (5.2) e (5.5))
  - <3.3> Obter  $I_j^{(3)}$  e  $Y_j^{(3)}$ ; (conforme (5.3) e (5.6))
- <4> Disponibilizar as saídas da rede, as quais são dadas pelos elementos contidos em  $Y_j^{(3)}$

Passo Forward

#### Fim (Algoritmo PMC – Fase de Operação)

Torna-se relevante ressaltar que os ajustes das matrizes de pesos são realizados somente na fase de treinamento da rede, em que se aplicam os passos *forward* e *backward* a fim de proceder eventuais correções sinápticas necessárias. Já na fase de operação, nenhum tipo de ajuste é efetuado nos parâmetros internos da rede, sendo que, para esta ocasião, somente a fase *forward* é processada com o objetivo de gerar as saídas da rede.

#### 5.3.3 – Versões aperfeiçoadas do algoritmo *backpropagation*

Diversas variações do método *backpropagation* têm sido propostas com o objetivo de tornar o processo de convergência mais eficiente. Entre tais

aperfeiçoamentos, tem-se o método de inserção do termo de *momentum*, o *backpropagation* e o Levenberg-Marquardt.

### (A) Método de inserção do termo de *momentum*

A inserção do termo de *momentum* se configura como uma das variações mais simples de ser efetuada no algoritmo *backpropagation*, pois, basta inserir um único parâmetro visando ponderar o quão as matrizes sinápticas foram alteradas entre duas iterações anteriores e sucessivas. Formalmente, considerando os neurônios pertencentes à  $L$ -ésima camada, tem-se:

$$W_j^{(L)}(t+1) = W_j^{(L)}(t) + \underbrace{\alpha \cdot (W_j^{(L)}(t) - W_j^{(L)}(t-1))}_{\text{Termo de momentum}} + \underbrace{\eta \cdot \delta_j^{(L)} \cdot Y^{(L-1)}}_{\text{Termo de aprendizagem}} \quad (5.41)$$

onde  $\alpha$  é definida como taxa de *momentum* e seu valor está compreendido entre zero e um.

Conforme se pode abstrair de (5.41), quando o valor da taxa de *momentum* for igual a zero, a expressão se torna então equivalente àquela do *backpropagation* convencional. Por outro lado, para valores diferentes de zero, o termo de *momentum* passa a ser relevante, sendo que tal contribuição afetará positivamente o processo de convergência.

Mais especificamente, quando a solução atual (refletida por suas matrizes de peso) estiver longe da solução final (mínimo da função erro), a variação na direção oposta ao gradiente da função erro quadrático entre duas iterações sucessivas será também grande. Isto implica que a diferença entre as matrizes de pesos dessas duas iterações será bem considerável e, nesta situação, pode-se imprimir um passo maior de incremento para  $W^{(L)}$  em direção ao mínimo da função erro. A execução de tal tarefa ficará então a cargo do termo de *momentum*, pois o mesmo é responsável pela medição desta variação.

Entretanto, quando a solução atual estiver bem próxima da solução final, as variações nas matrizes de pesos serão então bem infimas, pois a variação do erro quadrático entre duas iterações sucessivas será baixa e, consequentemente, a contribuição do termo de *momentum* para o processo de convergência é bem

pequena. A partir deste instante, todos os ajustes nas matrizes de peso acabam sendo conduzidos (quase que na totalidade) apenas pelo termo de aprendizagem, como ocorre também no backpropagation convencional.

A figura 5.8 ilustra a contribuição do termo de *momentum* ( $TM$ ) e do termo de aprendizagem ( $TA$ ) visando a convergência em direção ao mínimo  $W^*$  da função erro quadrático.

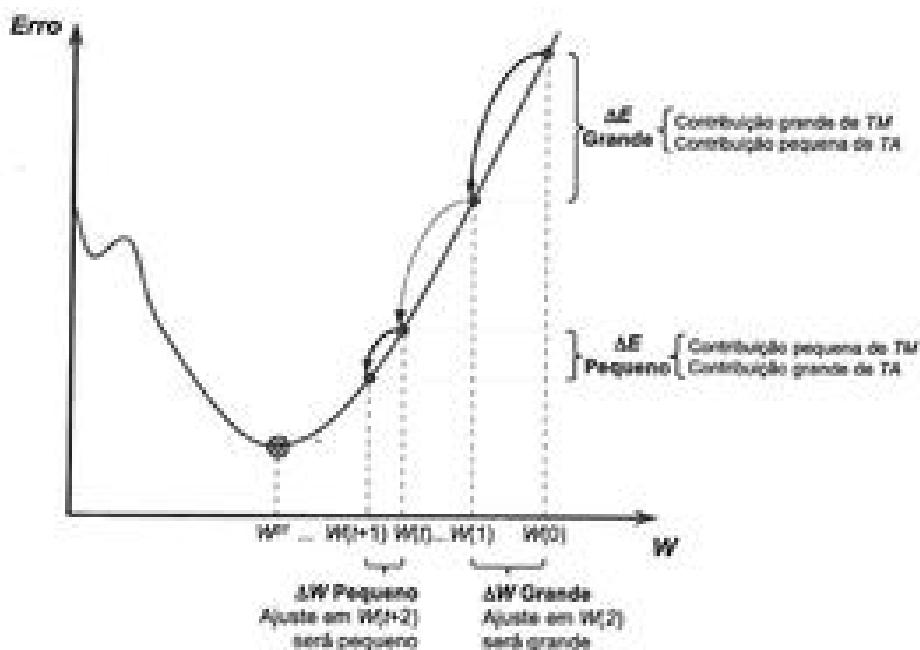


Figura 5.8 – Ilustração do processo de treinamento utilizando o método de inserção do termo de *momentum*.

Assim, por intermédio da inserção do termo de *momentum*, o processo de convergência da rede se torna bem mais eficiente, pois se leva em consideração o critério de quão afastada está a solução atual da solução final (ótima). O uso do termo de *momentum* implica acelerar a convergência da rede à razão de  $\eta/(1-\alpha)$ , conforme análises efetuadas em Reed & Marks II (1999). Valores compreendidos entre  $(0,05 \leq \eta \leq 0,75)$  e  $(0 \leq \alpha \leq 0,9)$  são normalmente recomendados para treinamento de redes PMC [Rumelhart *et al.*, 1986].

### (B) Método resilient-propagation

As funções de ativação do tipo logística, ou tangente hiperbólica, usadas nos neurônios das redes P-MC, frente aos seus domínios de definição, produzem valores limites 0 ou 1 (no caso da logística), ou -1 a 1 (no caso da tangente hiperbólica), para a maioria dos pontos, conforme se pode observar pela ilustração da figura 5.9. Tal circunstância implica em saturar as saídas dos neurônios para esses valores limites se seus potenciais de ativação ( $u$ ) possuírem valores elevados. Além disso, as derivadas parciais  $g'(u)$  produziriam ainda valores próximos de zero, que implicariam na lentidão do processo de treinamento, pois, de acordo com as expressões (5.11), (5.20) e (5.31), o back-propagation depende também do cálculo de  $g'(u)$ .

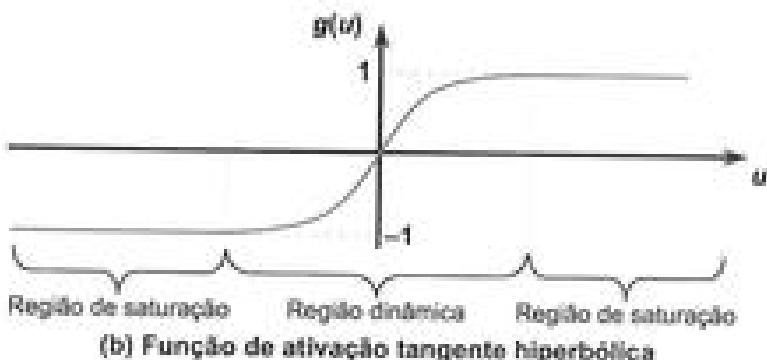
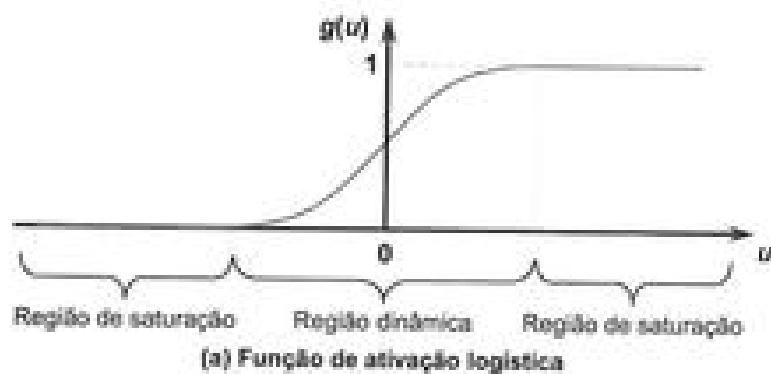


Figura 5.9 – Ilustração dos intervalos de saturação e de variação dinâmica das funções logística e tangente hiperbólica

Nesta condição, variações muito pequenas do gradiente da função erro em (5.7), combinado com as faixas de saturação das funções de ativação,

fazem com que o processo de convergência do algoritmo *backpropagation* se torne bem lento, pois haverá um gasto adicional de esforço computacional a fim de conduzir os valores das matrizes de pesos do PMC para as regiões de variação dinâmica dessas funções de ativação.

Assim, o objetivo do método *resilient-propagation*, ao invés de considerar as variações das magnitudes do gradiente da função erro, levará somente em conta a variação de seu sinal [Riedmiller & Braun, 1993]. Desta forma, a taxa de aprendizagem do método se torna dinâmica, pois quando os sinais do gradiente forem os mesmos, considerando duas iterações sucessivas, significa que se pode incrementar a taxa de aprendizado em virtude de se estar relativamente distante de um ponto de mínimo (gradiente nulo) da função erro. Caso contrário, se os sinais do gradiente forem diferentes, significa então que o ponto de mínimo da função foi ultrapassado, e isto implica reduzir a taxa de aprendizagem a fim de se convergir suavemente para o mesmo, levando-se também em conta a precisão requerida ao problema.

Uma ilustração do processo de convergência envolvido com o método *resilient-propagation* é apresentada na figura 5.10, em que os passos (I), (II), (III) e (V) implicam em incrementos positivos (crescentes) na taxa de aprendizado, pois possuem variações de sinais de gradiente também positivos; enquanto que os passos (IV), (VI) e (VII) implicam em incrementos negativos (decrescentes), pois suas variações de sinais são igualmente negativas.

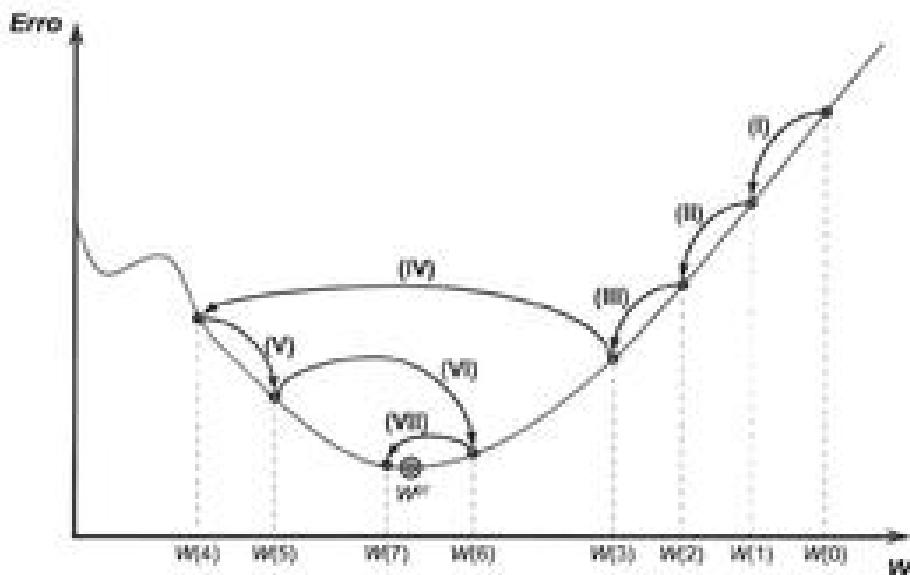


Figura 5.10 – Ilustração do mecanismo de convergência do método *resilient-propagation*

Em termos matemáticos, a avaliação da condição de mudança de sinal do gradiente é dada pelas seguintes expressões:

$$\Lambda_j^{(L)}(t) = \begin{cases} \eta^+ \cdot \Lambda_j^{(L)}(t-1), & \text{se } \frac{\partial E(t-1)}{\partial W_j^{(L)}} \cdot \frac{\partial E(t)}{\partial W_j^{(L)}} > 0 \\ \eta^- \cdot \Lambda_j^{(L)}(t-1), & \text{se } \frac{\partial E(t-1)}{\partial W_j^{(L)}} \cdot \frac{\partial E(t)}{\partial W_j^{(L)}} < 0 \\ \Lambda_j^{(L)}(t-1), & \text{caso contrário} \end{cases} \quad (5.42)$$

onde  $\Lambda_j^{(L)}$  são consideradas as taxas de aprendizado individuais associadas a cada um dos elementos  $W_j^{(L)}$  referentes às  $L$ -ésimas matrizes de pesos do PMC, sendo que ( $0 < \eta^- < 1$ ) e ( $\eta^+ > 1$ ) são constantes responsáveis por incrementar ou decrementar as taxas de aprendizado individuais de cada um dos pesos.

Finalmente, as matrizes de pesos do PMC são em tal situação alteradas da seguinte forma:

$$\Delta W_j^{(L)}(t) = \begin{cases} -\Lambda_j^{(L)}(t), & \text{se } \frac{\partial E(t)}{\partial W_j^{(L)}} > 0 \\ +\Lambda_j^{(L)}(t), & \text{se } \frac{\partial E(t)}{\partial W_j^{(L)}} < 0 \\ 0, & \text{caso contrário} \end{cases} \quad (5.43)$$

Portanto, observa-se nesta expressão anterior que a alteração nas matrizes de pesos da rede fica somente em virtude das variações dos valores dos sinais das derivadas parciais, desconsiderando-se os valores de suas magnitudes.

### (C) Método de Levenberg-Marquardt

Como descrito anteriormente, o algoritmo *backpropagation* ajusta os valores das matrizes de pesos da rede PMC em relação à direção oposta do gradiente da função erro quadrático. Entretanto, a utilização deste algoritmo na prática tende a convergir muito lentamente, exigindo-se assim um eleva-

do esforço computacional. Para contornar este inconveniente, várias técnicas de otimização têm sido incorporadas ao algoritmo *backpropagation* a fim de reduzir o seu tempo de convergência e diminuir o esforço computacional requerido. Dentre as técnicas de otimização mais utilizadas para este propósito destaca-se o algoritmo de Levenberg-Marquardt [Hagan & Menhaj, 1994].

O algoritmo de Levenberg-Marquardt é um método gradiente de segunda ordem, baseado no método dos mínimos quadrados para modelos não-lineares, que pode ser incorporado ao algoritmo *backpropagation* a fim de potencializar a eficiência do processo de treinamento. Para este algoritmo, as funções erro quadrático e erro quadrático médio, fornecidas respectivamente nas expressões (5.7) e (5.8), podem ser expressas conjuntamente por:

$$\begin{aligned} V &= \frac{1}{2p} \cdot \sum_{k=1}^p \sum_{j=1}^{n_j} (d_j(k) - Y_j^{(3)}(k))^2 \\ V &= \frac{1}{2p} \cdot \sum_{k=1}^p (d(k) - Y^{(3)}(k))^T (d(k) - Y^{(3)}(k)) \\ V &= \frac{1}{2p} \cdot \sum_{k=1}^p E^T(k) \cdot E(k) \end{aligned} \quad (5.44)$$

onde o termo  $\{E(k) = d(k) - Y^{(3)}(k)\}$  denota o vetor erro em relação à  $k$ -ésima amostra de treinamento. Para uma amostra  $k$  específica, o erro é obtido por:

$$V = \frac{1}{2} \cdot E^T(k) \cdot E(k) \quad (5.45)$$

Enquanto que o algoritmo *backpropagation* é um método de descida no gradiente da função erro quadrático a fim de minimizá-la, o algoritmo de Levenberg-Marquardt é uma aproximação do método de Newton [Battiti, 1992; Foressee & Hagan, 1997]. Por sua vez, a minimização de uma função  $V(z)$  em relação a um vetor paramétrico  $z$  é dada pelo seguinte procedimento iterativo:

$$\Delta z = -(\nabla^2 V(z))^{-1} \cdot \nabla V(z) \quad (5.46)$$

onde  $\nabla^2 V(\mathbf{z})$  denota a matriz Hessiana (matriz de derivadas de segunda ordem) e  $\nabla V(\mathbf{z})$  é a matriz Jacobiana (matriz de derivadas de primeira ordem) de  $V(\mathbf{z})$ . Assumindo-se que  $V(\mathbf{z})$  é uma função que executa a soma de  $m$  funções quadráticas, como aquelas representadas em (5.44), para um vetor paramétrico  $\mathbf{z}$  composto por  $q$  elementos, tem-se então a seguinte expressão:

$$V(\mathbf{z}) = \sum_{i=1}^m \phi_i^2(\mathbf{z}) \quad (5.47)$$

Assim, a partir da equação anterior, pode ser mostrado que:

$$\nabla V(\mathbf{z}) = J^T(\mathbf{z}) \cdot \phi(\mathbf{z}) \quad (5.48)$$

$$\nabla^2 V(\mathbf{z}) = J^T(\mathbf{z}) \cdot J(\mathbf{z}) + \mu \cdot I \quad (5.49)$$

onde  $I$  é a matriz identidade,  $\mu$  é um parâmetro que ajusta a taxa de convergência do algoritmo de Levenberg-Marquardt e  $J(\mathbf{z})$  é a matriz Jacobiana, a qual é definida por:

$$J(\mathbf{z}) = \begin{bmatrix} \frac{\partial \phi_1(\mathbf{z})}{\partial z_1} & \frac{\partial \phi_1(\mathbf{z})}{\partial z_2} & \cdots & \frac{\partial \phi_1(\mathbf{z})}{\partial z_q} \\ \frac{\partial \phi_2(\mathbf{z})}{\partial z_1} & \frac{\partial \phi_2(\mathbf{z})}{\partial z_2} & \cdots & \frac{\partial \phi_2(\mathbf{z})}{\partial z_q} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \phi_m(\mathbf{z})}{\partial z_1} & \frac{\partial \phi_m(\mathbf{z})}{\partial z_2} & \cdots & \frac{\partial \phi_m(\mathbf{z})}{\partial z_q} \end{bmatrix} \quad (5.50)$$

Inserindo os resultados de (5.48) e (5.49) em (5.46), obtém a expressão iterativa do método de Levenberg-Marquardt, isto é:

$$\Delta \mathbf{z} = (J^T(\mathbf{z}) \cdot J(\mathbf{z}) + \mu \cdot I)^{-1} \cdot J^T(\mathbf{z}) \cdot \phi(\mathbf{z}) \quad (5.51)$$

Portanto, a característica principal deste algoritmo é a computação da matriz Jacobiana. Para o processo de treinamento das redes PMC, conforme ilustração da figura 5.3, esta matriz Jacobiana (5.50) passa a ser reescrita em função das matrizes sinápticas da rede, ou seja:

$$J(W) = [J(W^{(1)}) \mid J(W^{(2)}) \mid J(W^{(3)})] \quad (5.52)$$

sendo que  $J(W) \in \mathbb{R}^{(p) \times ((n+1)n_1 + (n_1+1)n_2 + (n_2+1)n_3)}$ . Neste caso,  $W$  é composta por:

$$W = [W^{(1)} \mid W^{(2)} \mid W^{(3)}] = \\ \begin{bmatrix} W_{1,0}^{(1)} \dots W_{1,n_1}^{(1)} & W_{2,0}^{(1)} \dots W_{2,n_1}^{(1)} & \dots & W_{n_1,0}^{(1)} \dots W_{n_1,n_1}^{(1)} & | \\ W_{1,0}^{(2)} \dots W_{1,n_2}^{(2)} & W_{2,0}^{(2)} \dots W_{2,n_2}^{(2)} & \dots & W_{n_2,0}^{(2)} \dots W_{n_2,n_2}^{(2)} & | \\ W_{1,0}^{(3)} \dots W_{1,n_3}^{(3)} & W_{2,0}^{(3)} \dots W_{2,n_3}^{(3)} & \dots & W_{n_3,0}^{(3)} \dots W_{n_3,n_3}^{(3)} & ]^T \end{bmatrix} \quad (5.53)$$

onde  $W \in \mathbb{R}^{((n+1)n_1 + (n_1+1)n_2 + (n_2+1)n_3)}$ .

As matrizes  $J(W^{(1)})$ ,  $J(W^{(2)})$  e  $J(W^{(3)})$  são então, por sua vez, definidas como:

$$J(W^{(1)}) = \begin{bmatrix} \frac{\partial E(1)}{\partial W_{1,1}^{(1)}} \dots \frac{\partial E(1)}{\partial W_{1,n_1}^{(1)}} & \frac{\partial E(1)}{\partial W_{2,1}^{(1)}} \dots \frac{\partial E(1)}{\partial W_{2,n_1}^{(1)}} & \dots & \frac{\partial E(1)}{\partial W_{n_1,1}^{(1)}} \dots \frac{\partial E(1)}{\partial W_{n_1,n_1}^{(1)}} \\ \frac{\partial E(2)}{\partial W_{1,1}^{(1)}} \dots \frac{\partial E(2)}{\partial W_{1,n_1}^{(1)}} & \frac{\partial E(2)}{\partial W_{2,1}^{(1)}} \dots \frac{\partial E(2)}{\partial W_{2,n_1}^{(1)}} & \dots & \frac{\partial E(2)}{\partial W_{n_1,1}^{(1)}} \dots \frac{\partial E(2)}{\partial W_{n_1,n_1}^{(1)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E(p)}{\partial W_{1,1}^{(1)}} \dots \frac{\partial E(p)}{\partial W_{1,n_1}^{(1)}} & \frac{\partial E(p)}{\partial W_{2,1}^{(1)}} \dots \frac{\partial E(p)}{\partial W_{2,n_1}^{(1)}} & \dots & \frac{\partial E(p)}{\partial W_{n_1,1}^{(1)}} \dots \frac{\partial E(p)}{\partial W_{n_1,n_1}^{(1)}} \end{bmatrix} \quad (5.54)$$

$$J(W^{(2)}) = \begin{bmatrix} \frac{\partial E(1)}{\partial W_{1,1}^{(2)}} \dots \frac{\partial E(1)}{\partial W_{1,n_1}^{(2)}} & \frac{\partial E(1)}{\partial W_{2,1}^{(2)}} \dots \frac{\partial E(1)}{\partial W_{2,n_1}^{(2)}} & \dots & \frac{\partial E(1)}{\partial W_{n_2,1}^{(2)}} \dots \frac{\partial E(1)}{\partial W_{n_2,n_1}^{(2)}} \\ \frac{\partial E(2)}{\partial W_{1,1}^{(2)}} \dots \frac{\partial E(2)}{\partial W_{1,n_1}^{(2)}} & \frac{\partial E(2)}{\partial W_{2,1}^{(2)}} \dots \frac{\partial E(2)}{\partial W_{2,n_1}^{(2)}} & \dots & \frac{\partial E(2)}{\partial W_{n_2,1}^{(2)}} \dots \frac{\partial E(2)}{\partial W_{n_2,n_1}^{(2)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E(p)}{\partial W_{1,1}^{(2)}} \dots \frac{\partial E(p)}{\partial W_{1,n_1}^{(2)}} & \frac{\partial E(p)}{\partial W_{2,1}^{(2)}} \dots \frac{\partial E(p)}{\partial W_{2,n_1}^{(2)}} & \dots & \frac{\partial E(p)}{\partial W_{n_2,1}^{(2)}} \dots \frac{\partial E(p)}{\partial W_{n_2,n_1}^{(2)}} \end{bmatrix} \quad (5.55)$$

$$J(W^{(3)}) = \begin{bmatrix} \frac{\partial E(1)}{\partial W_{11}^{(3)}} & \frac{\partial E(1)}{\partial W_{1, n_2}^{(3)}} & \frac{\partial E(1)}{\partial W_{21}^{(3)}} & \frac{\partial E(1)}{\partial W_{2, n_2}^{(3)}} & \cdots & \frac{\partial E(1)}{\partial W_{n_1, 1}^{(3)}} & \frac{\partial E(1)}{\partial W_{n_1, n_2}^{(3)}} \\ \frac{\partial E(2)}{\partial W_{11}^{(3)}} & \frac{\partial E(2)}{\partial W_{1, n_2}^{(3)}} & \frac{\partial E(2)}{\partial W_{21}^{(3)}} & \frac{\partial E(2)}{\partial W_{2, n_2}^{(3)}} & \cdots & \frac{\partial E(2)}{\partial W_{n_1, 1}^{(3)}} & \frac{\partial E(2)}{\partial W_{n_1, n_2}^{(3)}} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial E(\rho)}{\partial W_{11}^{(3)}} & \frac{\partial E(\rho)}{\partial W_{1, n_2}^{(3)}} & \frac{\partial E(\rho)}{\partial W_{21}^{(3)}} & \frac{\partial E(\rho)}{\partial W_{2, n_2}^{(3)}} & \cdots & \frac{\partial E(\rho)}{\partial W_{n_1, 1}^{(3)}} & \frac{\partial E(\rho)}{\partial W_{n_1, n_2}^{(3)}} \end{bmatrix} \quad (5.56)$$

onde  $J(W^{(1)}) \in \mathbb{R}^{(\rho) \times (n_1 n_2)}$ ,  $J(W^{(2)}) \in \mathbb{R}^{(\rho) \times (n_1 n_2)}$  e  $J(W^{(3)}) \in \mathbb{R}^{(\rho) \times (n_2 n_3)}$ .

A partir de (5.51), a expressão iterativa do método de Levenberg-Marquardt visando ajustar as matrizes de pesos do PMC passa a ser redefinida por:

$$\Delta W = (J^T(W) \cdot J(W) + \mu \cdot I)^{-1} \cdot J^T(W) \cdot E \quad (5.57)$$

onde  $E = [E(1) \ E(2) \dots E(\rho)]^T$  é o vetor erro referente às  $\rho$  amostras de treinamento.

Finalmente, os elementos das matrizes  $J(W^{(1)})$ ,  $J(W^{(2)})$  e  $J(W^{(3)})$  são obtidos, sequencialmente, a partir das fases *forward* e *backward* empregadas no algoritmo *backpropagation* convencional, o qual foi apresentado na subseção anterior. Por intermédio da implementação destas modificações, comprova-se que o método de Levenberg-Marquardt consegue conduzir o treinamento de redes PMC na ordem de 10 a 100 vezes mais rápido que o algoritmo *backpropagation* convencional [Hagan & Menhaj, 1994]. Entretanto, problemas de convergência podem ocorrer no caso de a matriz  $J(z)$ , usada em (5.51), ser mal condicionada.

## 5.4 – Aplicabilidade das redes *Perceptron* multicamadas

As redes *Perceptron* multicamadas podem ser consideradas as mais utilizadas na solução de problemas advindos de áreas relacionadas às ciências e às engenharias. De fato, constatam-se aplicações de redes PMC nas mais variadas áreas do conhecimento, tais como medicina, biologia, química, física, economia, geologia, ecologia e psicologia, além de vastas empregabilidades nas diferentes temáticas envolvendo as engenharias como um todo.

Considerando os leques de aplicabilidades em que as redes PMC são passíveis de serem utilizadas, destacam-se três classes de problemas que acabam concentrando grande parte de suas aplicações, ou seja, os problemas que envolvem a classificação (reconhecimento) de padrões, os problemas relacionados à aproximação de funções e aqueles direcionados para sistemas dinâmicos (variantes no tempo). Devido às suas relevâncias, essas três classes de problemas serão tratadas separadamente nas subseções seguintes.

#### 5.4.1 – Problemas envolvendo classificação de padrões

Conforme destacado no capítulo 1, um problema de classificação de padrões consiste em associar um padrão de entrada (amostra) para uma das classes que foram previamente definidas. Como exemplo, pode-se ter uma aplicação em que o PMC seja treinado para reconhecer sinais de vozes a fim de permitir acesso de pessoas à ambientes restritos. Nessa situação, considerando-se que o treinamento foi realizado com vocábulos de apenas três pessoas, a resposta da rede, frente a um sinal de voz inserido em suas entradas, trataria então de informar de quem seria o referido sinal.

Outro aspecto relevante que pode ser abstraido, a partir deste simples exemplo, é que as saídas associadas aos problemas de classificação de padrões estão sempre relacionadas com grandezas discretas (enumeráveis). As situações mais elementares seriam aquelas das saídas binárias, em que se têm apenas duas classes como possíveis respostas, sendo que as mesmas poderiam estar representando, por exemplo, a "presença" ou "ausência" de determinado atributo em uma amostra apresentada à rede. Assim, considerando-se que a saída da rede fornece respostas numéricas, uma possível codificação seria assinalar o valor 0 ao atributo "presença", ao passo que o valor 1 estaria rotulando o atributo "ausência". Conforme apresentado mais adiante, uma sistemática similar a esta poderia ser também utilizada para problemas multi-classes (três ou mais classes).

Ainda em relação aos problemas com saídas binárias, de acordo com o capítulo 2, o *Perceptron* simples (camada única) somente conseguiria convergir se as duas classes envolvidas com o problema a ser mapeado fossem linearmente separáveis. Caso contrário, o *Perceptron* simples jamais conseguiria convergir a fim de posicionar o seu hiperplano na faixa delimitada pela fronteira de separabilidade entre as classes. Um caso clássico de tal fato é encontra-

do no problema do ou-exclusivo (porta  $X_{\text{OR}}$ ), envolvendo a lógica booleana, como ilustrado na figura 5.11.

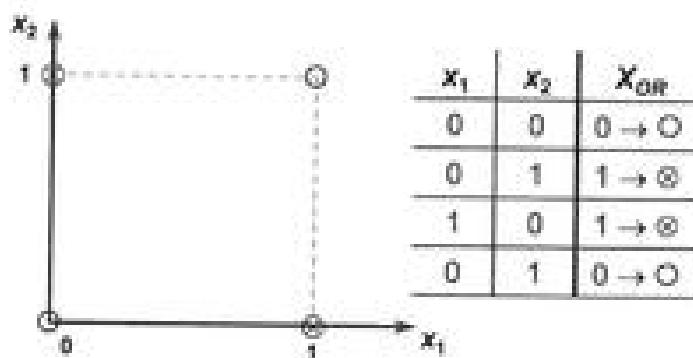


Figura 5.11 – Ilustração do problema do ou-exclusivo

Fazendo uso da representação gráfica da figura 5.11, constata-se que seria impossível posicionar uma única reta que permitiria separar as duas classes resultantes do problema do ou-exclusivo.

Outras situações similares somente podem ser resolvidas por intermédio de uma rede PNC de duas camadas neurais, tal como esta representada na figura 5.12.

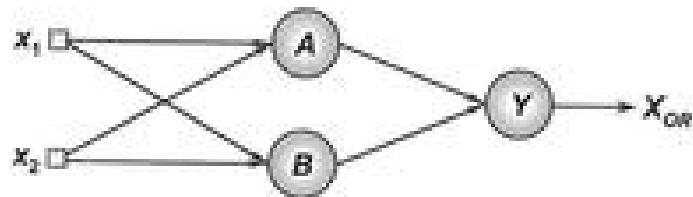


Figura 5.12 – Rede PNC aplicada no problema do ou-exclusivo

Para se compreender os mecanismos envolvidos com essa classificação de padrões, ilustra-se na figura 5.13 uma configuração de retas (de separabilidade) que serão implementadas pelos neurônios A e B, considerando-se a configuração topológica da figura 5.12, quando da aplicação do algoritmo *backpropagation*.

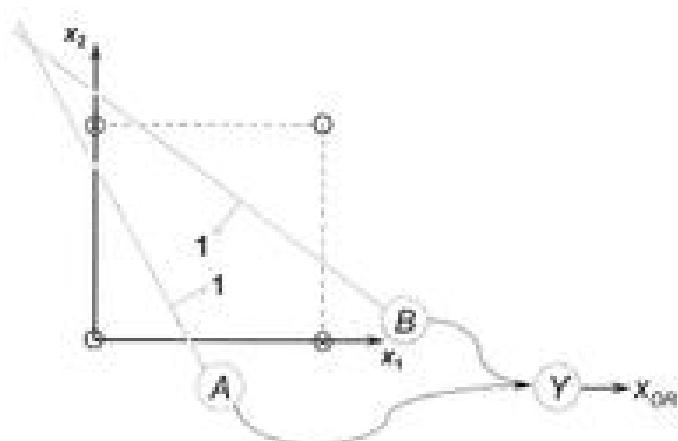


Figura 5.13 – Separabilidade no problema do ou-exclusivo

Em tal exemplo, assumindo a função logística em cada um dos neurônios do PMC da figura 5.12, observa-se que o neurônio *A* terá sua saída igual a 1 apenas para aqueles padrões que estejam acima de sua reta, enquanto que o neurônio *B* fornecerá o valor 1 para todos os padrões que estejam abaixo. Consequentemente, o neurônio *Y* produzirá em sua saída o valor igual a 1 somente para os casos em que as respostas dos neurônios *A* e *B* forem também iguais a 1; caso contrário, o valor de sua saída será igual a 0. Portanto, torna-se ainda interessante assinalar que o neurônio *Y* está simplesmente realizando uma operação de conjunção booleana (porta *AND*).

A figura 5.14 ilustra a superfície de separabilidade proporcionada pela função logística do neurônio *A*, em que se verifica a produção do valor 1 somente para  $(x_1 = 0) \wedge (x_2 = 1)$ , assim como para  $(x_1 = 1) \wedge (x_2 = 0)$ .

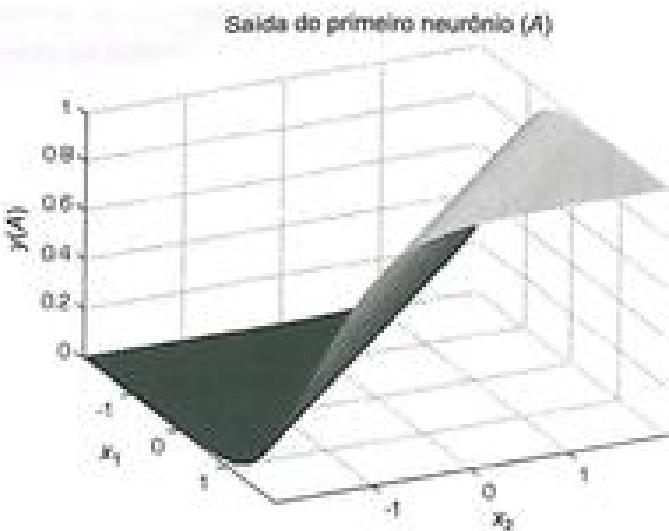


Figura 5.14 – Ilustração da saída proporcionada pela função logística do neurônio A

Já a figura 5.15 ilustra a combinação das duas funções logísticas efetuadas pelo neurônio Y, na qual se verifica que o mesmo inverteu a função logística relacionada ao neurônio B a fim de classificar corretamente as classes do problema do ou-exclusivo.

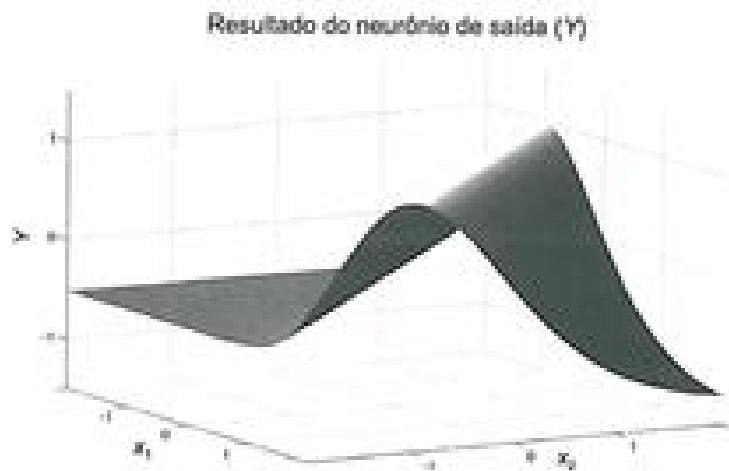


Figura 5.15 – Ilustração da saída do neurônio Y que combina as duas funções logísticas advindas dos neurônios A e B

Finalmente, a figura 5.16 mostra as fronteiras de classificação associadas ao problema do ou-exclusivo, as quais são delimitadas por duas retas, quando se executa a intersecção da superfície ilustrada na figura 5.15 próximo ao plano  $y = 0$ .

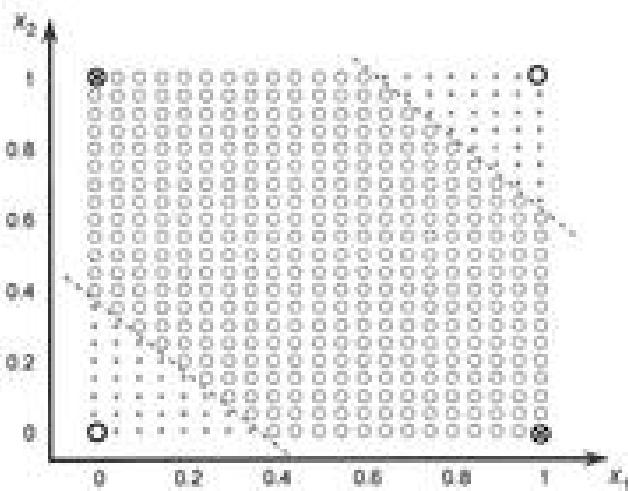


Figura 5.16 – Fronteiras de classificação para o problema do ou-exclusivo

De forma similar, pode-se então deduzir que um PMC de duas camadas neurais, sendo uma delas a camada escondida e a outra a própria camada de saída, é capaz de mapear qualquer problema de classificação de padrões cujos elementos estejam dentro de uma região convexa, tais como aquelas ilustradas na figura 5.17.

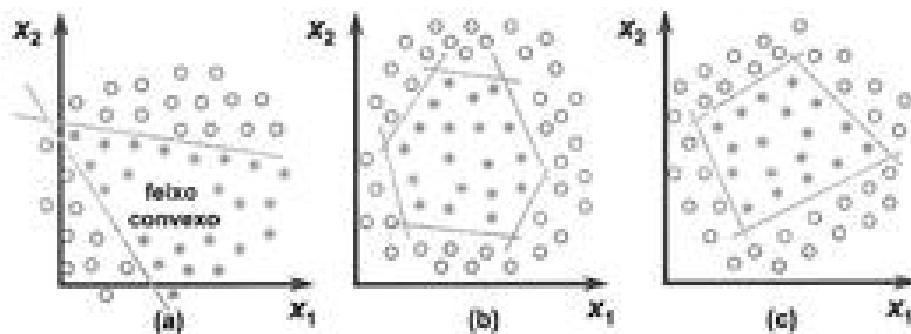


Figura 5.17 – Exemplos de regiões convexas para o problema de classificação de padrões

Em tais exemplos, a classificação das amostras representadas pela figura 5.17(a) necessitaria de dois neurônios na camada escondida do PMC, enquanto que para aquelas amostras das figuras 5.17(b) e 5.17(c) haveria então a necessidade de se utilizar seis e quatro neurônios nas respectivas camadas escondidas.

Do ponto de vista geométrico, uma região é considerada convexa se, e somente se, todos os pontos contidos em quaisquer segmentos de reta, os quais estão também definidos por quaisquer dois pontos delimitados pelo respectivo domínio, estiverem ainda dentro desta. A figura 5.18(a) apresenta uma ilustração de região convexa, enquanto a figura 5.18(b) mostra uma região não-convexa.

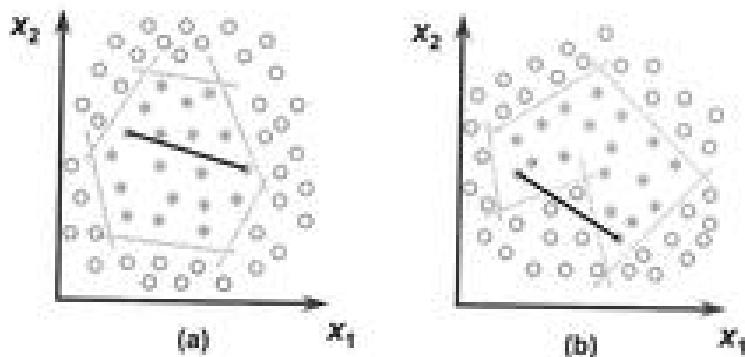


Figura 5.18 – Ilustração de região convexa e região não-convexa

Assim, considerando que redes PMC de apenas uma camada escondida conseguem classificar padrões que estejam dispostos em regiões convexas, pode-se também deduzir que redes PMC de duas camadas escondidas são capazes de classificar padrões que estejam em quaisquer tipos de regiões geométricas [Lippmann, 1987], inclusive com formato não-convexo como esta da figura 5.19.

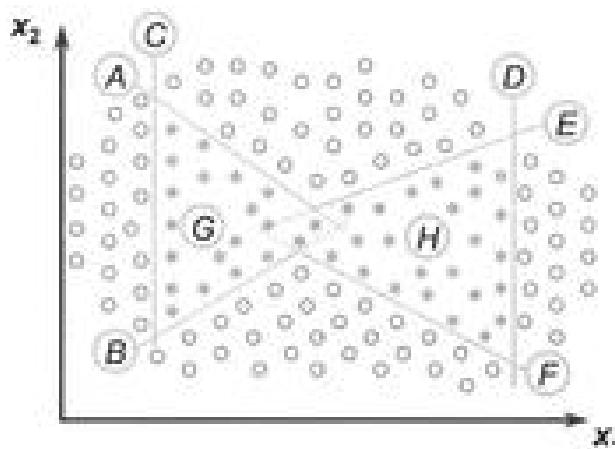


Figura 5.19 – Separabilidade em problemas com fronteiras não-convexas

Para tanto, a configuração de rede PMC mostrada na figura 5.20 representa uma topologia passível de implementar a classificação de padrões envolvida com a ilustração da figura 5.19.

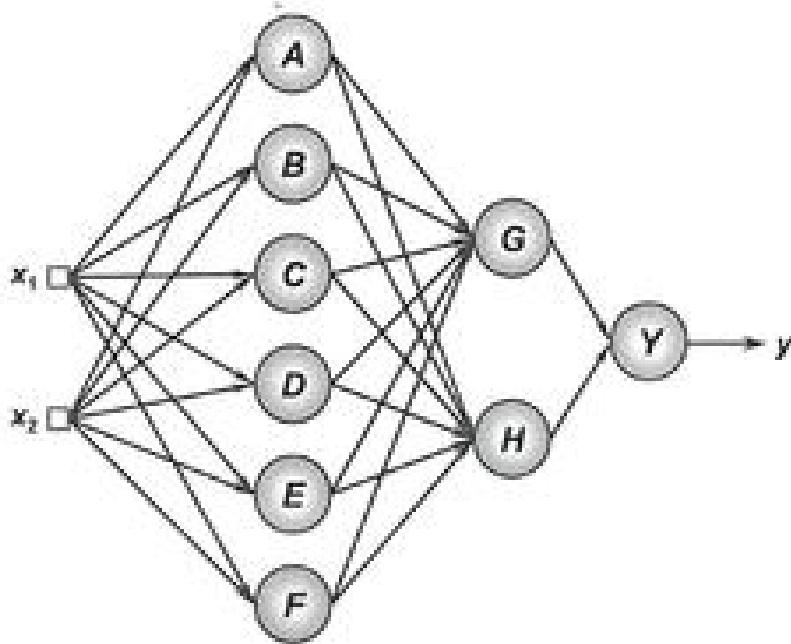


Figura 5.20 – Exemplo de rede PMC aplicada em problemas de classificação de padrões com fronteiras não-convexas

Neste caso, pode-se considerar, por exemplo, que os neurônios A, B e C da primeira camada intermediária seriam responsáveis pela delimitação da região convexa (triângulo) da esquerda, enquanto que os neurônios D, E e F estariam envolvidos com a região convexa (triângulo) da direita. Por sua vez, o neurônio G da segunda camada intermediária poderia ser responsável por combinar as saídas advindas dos neurônios A, B e C a fim de enquadrar a parcela dos padrões que pertencem à região convexa da esquerda, sendo que nesta condição a sua saída seria igual a 1. De maneira similar, o neurônio H produziria valor igual a 1 com o objetivo de representar a região convexa da direita, cuja geometria cercaria o restante dos demais padrões pertencentes à mesma classe.

Finalmente, o neurônio Y da figura 5.20 estaria então incumbido de efectuar uma operação de disjunção booleana (porta OR), pois se qualquer uma das saídas produzidas pelos neurônios G ou H for igual a 1, a sua resposta final y seria também igual a 1.

Adicionalmente, o PMC com duas camadas escondidas poderia ainda mapear outros tipos de regiões geométricas, tais como aquelas formadas por conjuntos disjuntos (desconexos), conforme ilustradas na figura 5.21.

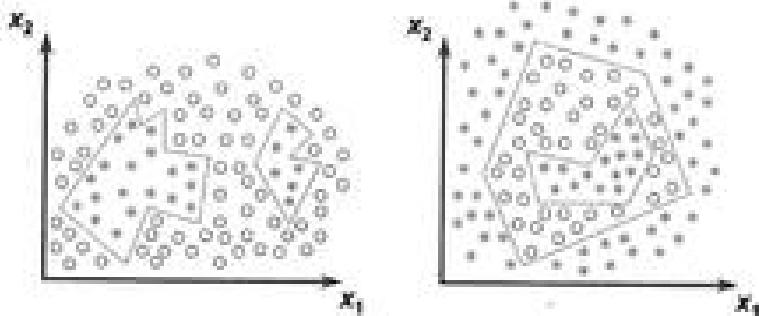


Figura 5.21 – Exemplos de regiões disjuntas em classificação de padrões

Por outro lado, embora um PMC constituído de duas camadas escondidas seja suficiente para reconhecer padrões que estejam delimitados por quaisquer regiões geométricas, há situações em que se utilizam mais de duas camadas escondidas, pois tais configurações podem ser apropriadas tanto para o propósito de incrementar o desempenho do processo de treinamento como de reduzir a topologia estrutural da rede.

Além disso, há outras situações particulares, conforme relatadas em

Makhoul *et alii* (1989) e Lai (1990), em que redes PMC com apenas uma camada escondida seriam ainda capazes de mapear problemas cujos padrões estariam também dispostos em regiões disjuntas ou não-convexas.

Em suma, pode-se concluir que os neurônios de saída das redes PMC, quando aplicadas em problemas de classificação de padrões, realizam combinações lógicas, tais como AND e OR, das regiões que foram definidas pelos neurônios das camadas anteriores, independentemente da dimensão associada aos padrões de entrada. Tais operações lógicas somente são passíveis de mapeamento em virtude de serem linearmente separáveis. A figura 5.22 ilustra algumas dessas operações lógicas (linearmente separáveis) em que estes neurônios de saída poderiam estar implementando. Assim, considerando duas variáveis lógicas  $x_1$  e  $x_2$ , tem-se então a possibilidade de se implementar 16 operações booleanas; dentre as quais, apenas duas (ou-exclusivo e seu respectivo complemento) deixam de ser linearmente separáveis.

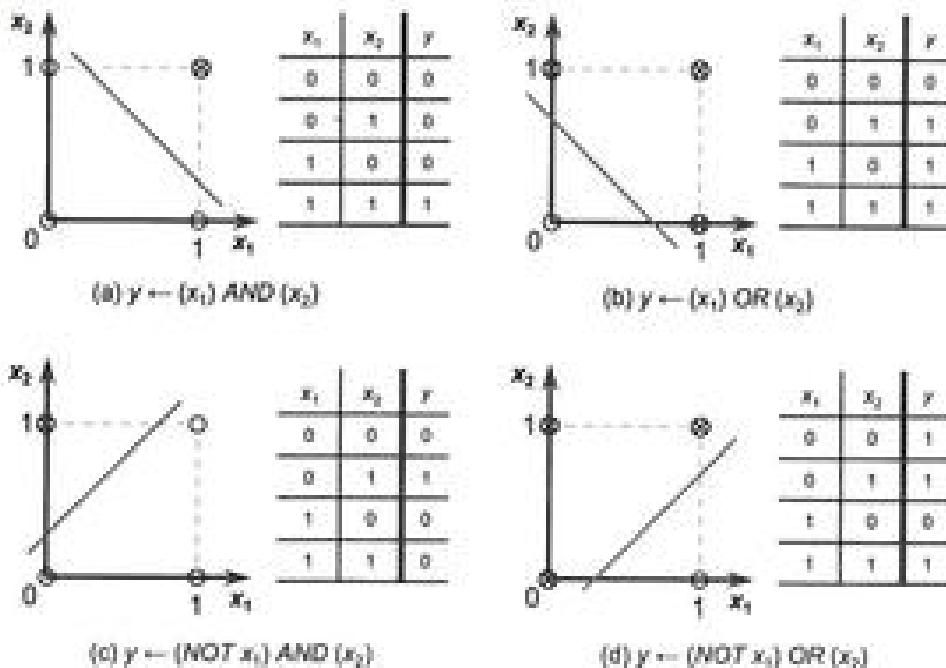


Figura 5.22 – Exemplos de operações lógicas linearmente separáveis

Complementarmente, em se tratando de problemas de classificação de padrões com mais de duas classes, há então a necessidade de se inserir mais

neurônios na camada de saída da rede, pois um PMC com apenas um neurônio em sua camada de saída é capaz de distinguir somente duas classes. Como exemplo, um PMC composto de dois neurônios em sua camada de saída poderia representar, no máximo, quatro classes possíveis (figura 5.23), ao passo que três neurônios poderiam diferenciar oito classes no total. Generalizando, um PMC com  $m$  neurônios em sua camada de saída seria capaz de classificar, teoricamente, até  $2^m$  classes possíveis.

Entretanto, em termos práticos, devido à eventual complexidade do problema a ser tratado, a adoção desta codificação sequencial de conjuntos pode tornar o treinamento do PMC bem mais difícil [Hampshire II & Pearlmuter, 1991], pois as classes estariam sendo representadas por pontos que estão espacialmente bem próximos entre si. Tal situação poderia então demandar um incremento substancial no número de neurônios de suas camadas intermediárias, além da ocasional dificuldade de seu ajuste topológico.

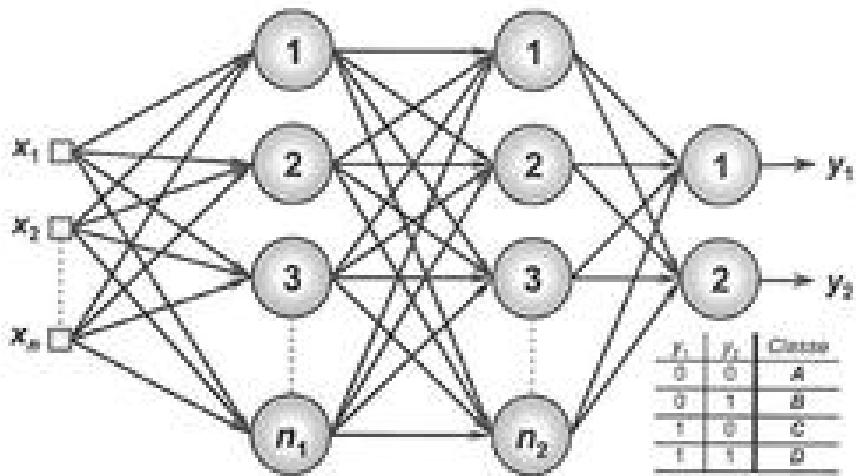


Figura 5.23 – Codificação sequencial para problemas de classificação de padrões envolvendo quatro classes possíveis

Alternativamente, um dos métodos mais utilizados é aquele denominado *one of r-classes*, o qual consiste em associar a saída de cada neurônio diretamente à classe correspondente, sendo que neste caso a quantidade de neurônios na camada de saída é igual ao número de classes do problema. Na condição de se ter quatro classes possíveis, a configuração de saída da rede PMC seria igualmente composta de quatro neurônios, conforme ilustração da figura 5.24.

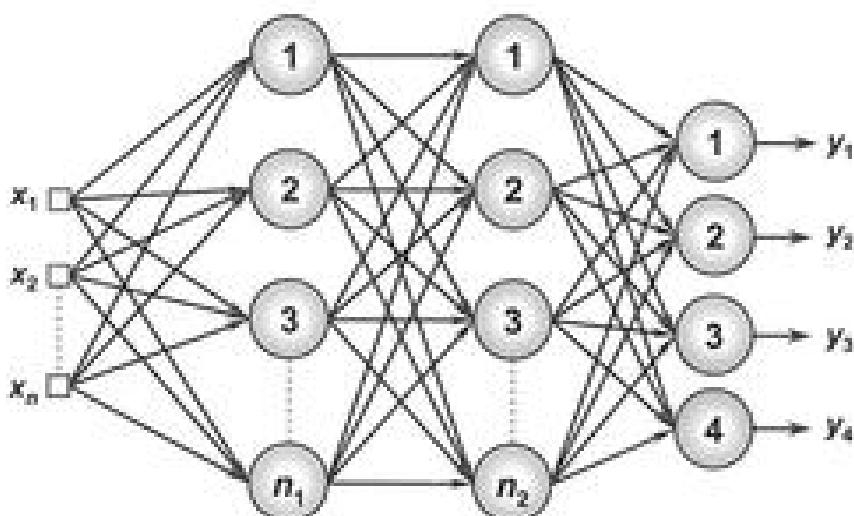


Figura 5.24 – Regra *one of c-classes* para classificação de padrões envolvendo quatro classes possíveis

Como exemplo, a tabela 5.1 descreve também as codificações dessas quatro classes, considerando-se para os neurônios de saída tanto a regra de codificação sequencial binária (figura 5.23) como a regra *one of c-classes* (figura 5.24).

Tabela 5.1 – Codificação de classes em problemas de classificação de padrões

Codificação sequencial			Codificação <i>one of c-classes</i>				
$y_1$	$y_2$	Classe	$y_1$	$y_2$	$y_3$	$y_4$	Classe
0	0	A	1	0	0	0	A
0	1	B	0	1	0	0	B
1	0	C	0	0	1	0	C
1	1	D	0	0	0	1	D

Em referência ainda à regra *one of c-classes*, inspecionando-se a tabela 5.1, verificou-se que uma determinada amostra pertenceria à classe B somente se a saída do neurônio  $y_3$  fosse igual a 1, sendo que todas as saídas dos demais devem ser iguais a 0. Conforme relatado anteriormente, tal estratégia de codificação ortogonal ficou conhecida como representação *one of c-classes* [Duda *et al.*, 2001], em que se assume que cada uma das classes do problema será representada por um neurônio de saída.

Finalmente, menciona-se que os valores  $y_i^{saída}$  produzidos pelos neurônios da camada de saída da rede PMC, considerando os problemas de classificação de padrões, devem ser pós-processados, pois as funções de ativação produzem números reais que, no caso da função logística, podem estar próximos de um ou próximos de zero. Face a esta circunstância, dependendo da precisão requerida, os valores  $y_i^{saída}$  advindos desta operação de pós-processamento podem ser obtidos pela aplicação da seguinte sistemática:

$$y_i^{pos} = \begin{cases} 1, & \text{se } y_i^{saída} \geq \delta m^{sup} \\ 0, & \text{se } y_i^{saída} \leq \delta m^{inf} \end{cases} \quad (5.58)$$

onde  $\delta m^{sup}$  e  $\delta m^{inf}$  definem, respectivamente, os valores de limiares superiores e inferiores para os neurônios da camada de saída da rede visando propósitos de atribuição das classes, isto é, se a saída do neurônio for maior ou igual ao  $\delta m^{sup}$ , atribui-se então o valor unitário; caso contrário, se for menor ou igual ao  $\delta m^{inf}$ , assume-se o valor zero. As especificações de tais limites dependem essencialmente da precisão requerida, sendo que para a função de ativação logística são tipicamente adotados valores de  $\delta m^{sup} \in [0,5 ; 0,9]$  e  $\delta m^{inf} \in [0,1 ; 0,5]$ . Outros detalhes desta implementação são apresentados na seção 5.9.

#### 5.4.2 – Problemas envolvendo aproximação funcional

A outra classe de problemas em que as redes PMC podem usufruir de maior destaque é aquela envolvida com a aproximação funcional, a qual consiste em mapear o comportamento de um processo buscando-se em diversas medições efetivadas em suas entradas e saídas. De fato, considerando tal aspecto, observa-se aqui uma das principais características intrínsecas das redes neurais artificiais, ou seja, o aprendizado a partir de exemplos, sendo que no caso de aproximação de funções, traduz-se na disponibilização de um conjunto de entradas e saídas que reproduzem o comportamento do sistema a ser tratado.

Em virtude desta capacidade de mapear processos por intermédio de exemplos, as redes PMC se tornam candidatas a muitas aplicações em que as únicas informações disponíveis se resumem a uma coleção de dados de entradas e saídas. Nesta direção, constata-se que as redes neurais artificiais têm sido extensivamente aplicadas em situações em que o processo a ser modelado é de certa forma complexo, nas quais as utilizações de métodos convencionais

produzem resultados insatisfatórios; ou então, naquelas situações em que o sistema já modelado se torna demasiadamente particularizado em torno de alguns pontos de operações que produzem soluções satisfatórias.

O teorema da aproximação universal aplicado ao PMC, o qual é baseado nas demonstrações de Kolmogorov (1957), fornece as bases necessárias para definir as configurações estruturais destas redes com a finalidade de mapear funções algébricas [Cybenko, 1989].

Assumindo que a função de ativação  $g(\cdot)$  a ser adotada nas redes PMC sejam contínuas e limitadas em suas imagens, tais como são a função logística e a tangente hiperbólica, demonstra-se então que uma topologia de PMC, constituída de apenas uma camada neural escondida, é capaz de mapear qualquer função contínua no espaço das funções reais. Em termos matemáticos, tem-se:

$$y(x_1, x_2, \dots, x_n) = \sum_{i=1}^n \underbrace{\lambda_i}_{\text{fator } (\ddagger)} \cdot \underbrace{g_i^{(1)}(u_i^{(1)})}_{\text{parcela } (\ddagger)}$$
(5.59)

$$u_i^{(1)} = \sum_{j=1}^n W_{ij}^{(1)} \cdot x_j - \theta_i$$
(5.60)

onde  $\lambda_i$  são constantes que ponderam as funções  $g_i^{(1)}(\cdot)$ .

As expressões (5.59) e (5.60) podem ser traduzidas para uma representação de rede PMC, como aquela ilustrada na figura 5.25, em que é composta de fato por apenas uma camada neural escondida, tendo-se a função logística (1.8) como ativação para os respectivos neurônios desta camada. Em outras palavras, conclui-se que a função  $y$  a ser mapeada pelo PMC será então constituída por uma superposição de funções de ativação do tipo logística (parcela  $(\ddagger)$ ), representadas pelos termos  $g_i^{(1)}(u_i^{(1)})$ , as quais são ponderadas pelos fatores  $\lambda_i$  (parcela  $(\ddagger)$ ). Similarmente, a demonstração é igualmente válida quando se assumem a tangente hiperbólica como funções de ativação para os neurônios da camada escondida.

Por conseguinte, utilizando como ativação da camada de saída a função linear (1.14), tem-se então que o único neurônio de saída realizará tão somente uma combinação linear das funções de ativação logística implementadas pelos neurônios da camada anterior.

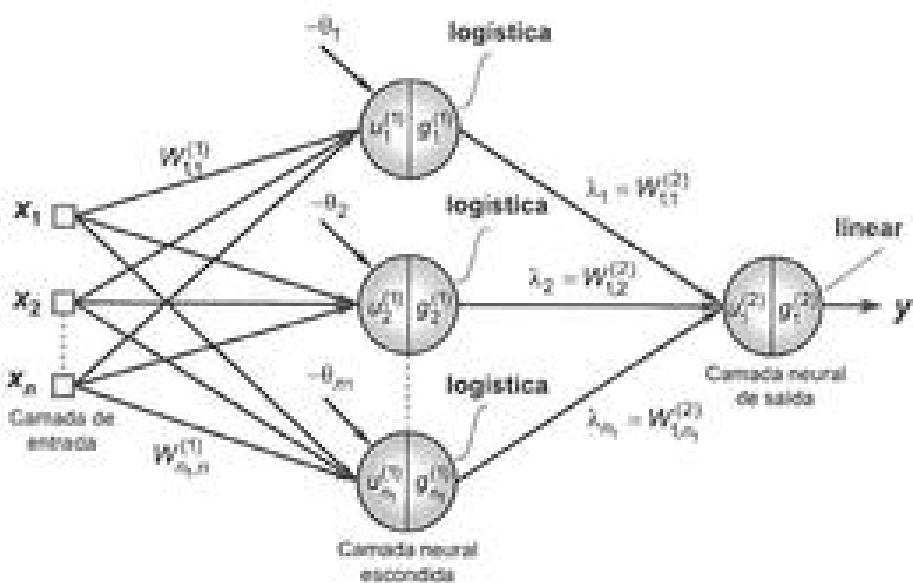


Figura 5.25 – Ilustração de rede PMC aplicada em aproximação funcional

Portanto, após o processo de treinamento da rede PMC, a matriz de pesos referentes ao neurônio de saída corresponderá aos próprios parâmetros  $\lambda_i$  da expressão (5.59), isto é,  $\lambda_i = W_{ij}^{(2)}$ .

Para fins de mera elucidação, considera-se que o PMC seja utilizado para implementar o problema de aproximação funcional, cujas amostras de treinamento estão dispostas na figura 5.26.

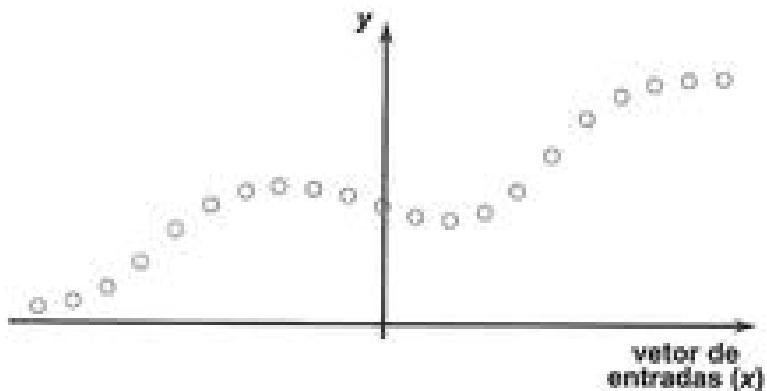


Figura 5.26 – Amostras de treinamento ao problema de aproximação funcional

Neste caso, deseja-se implementar um PMC para mapear (aproximar) o comportamento funcional do processo. Como exemplo de topologia, utilizar-se-á um PMC composto de três neurônios {A, B, C} em sua camada escondida, conforme ilustrado na figura 5.27.

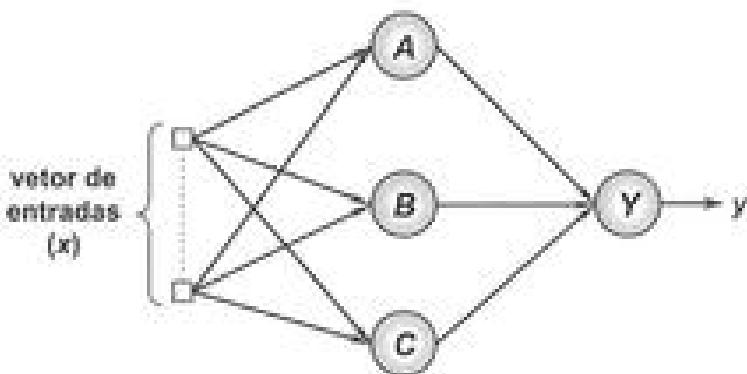


Figura 5.27 – Topologia de PMC usada no problema de aproximação funcional

Após o ajuste dos parâmetros internos do PMC, mediante a aplicação do algoritmo *backpropagation*, apresenta-se na figura 5.28 uma possível representação advinda da combinação linear efetuada sobre as funções de ativação logística (não-lineares) que compõem as saídas dos neurônios da camada escondida. Com efeito, verifica-se que o neurônio de saída acaba implementando uma operação que consegue mapear o comportamento do processo ilustrado na figura 5.26.

A partir ainda da figura 5.28, observa-se também que os limiares  $\{0\}$  dos neurônios  $A$ ,  $B$  e  $C$ , pertencentes à camada escondida, são responsáveis pela translação das funções de ativação em seus domínios de definição, ao passo que os pesos  $\{\lambda_j\}$  do neurônio de saída  $Y$  são responsáveis pelo escala-mento das mesmas.

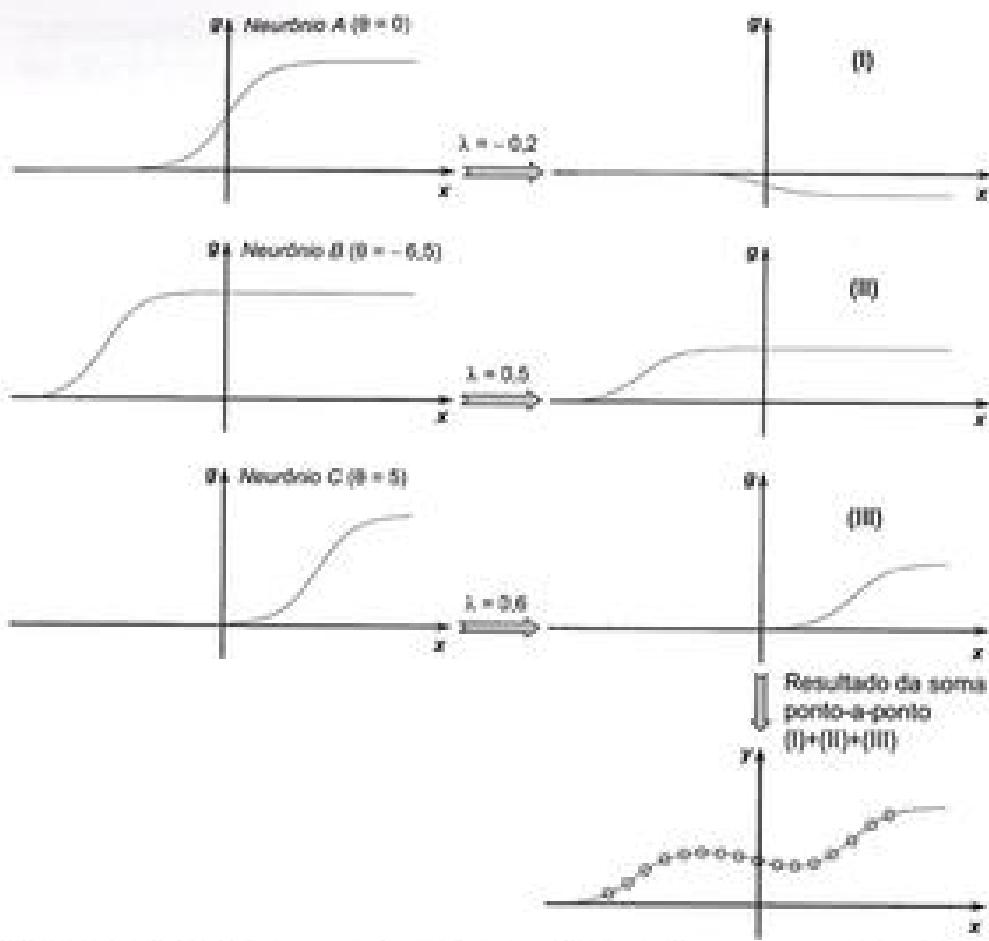


Figura 5.28 – Ilustração de superposição de funções do tipo logística para um problema de aproximação funcional

De forma similar aos problemas de classificação de padrões, embora um PMC com apenas uma camada escondida seja suficiente para mapar qualquer função não-linear contínua definida num domínio compacto (fechado), há situações em que se utiliza mais de uma camada escondida com o objetivo tanto de incrementar o desempenho do processo de treinamento como de reduzir a sua estrutura topológica.

Além disso, deve-se ressaltar que o teorema da aproximação universal enuncia tão somente a necessidade de uma única camada escondida; entretanto, o número de neurônios para realizar tal tarefa é ainda desconhecido, e dependendo da complexidade do problema a ser tratado, um número expressivo de neurônios poderá ser necessário.

Complementarmente, para alguns problemas particulares que se enquadram na classe de problemas inversos, tais como aqueles que envolvem cinemática inversa em robótica, prova-se que o mapeamento destes por uma rede PMC só se torna possível por intermédio de topologia com duas camadas escondidas [Sontag, 1992]. A figura 5.29 ilustra uma situação envolvendo a representação de uma função advinda do mapeamento de um problema inverso.

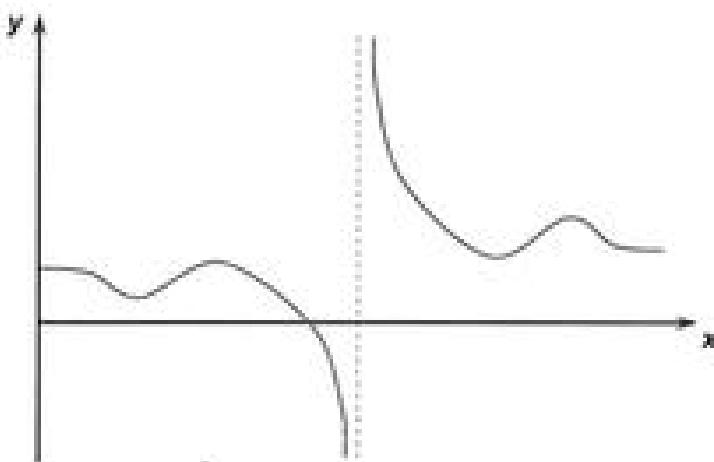


Figura 5.29 – Representação de uma função mapeando o comportamento de problema inverso

Conforme observado na figura 5.29, visualiza-se a presença de descontinuidade no domínio da função, sendo que tal condição é uma das responsáveis pela necessidade de se atribuir duas camadas escondidas ao PMC, quando de sua aplicação em problemas inversos.

#### 5.4.3 – Problemas envolvendo sistemas variantes no tempo

A última classe de problemas que será tratada neste livro, quanto à aplicabilidade de redes PMC, é denominada de sistemas dinâmicos, em que cujos comportamentos são considerados variantes no tempo ou dependentes dele.

Como exemplos de aplicação, tem-se a previsão de valores futuros para ações do mercado financeiro frente a um horizonte semanal, ou então, a previsão de consumo de energia elétrica para os próximos meses.

Em contraste aos problemas de aproximação de funções e reconhecimento de padrões (considerados estáticos), as saídas dos sistemas denomi-

nados dinâmicos, assumindo um instante de tempo qualquer, dependem de seus valores anteriores de saída e de entrada [Aguirre, 2000; Ljung, 1998].

Visando ressaltar ainda mais as diferenças entre problemas envolvendo com aproximação de funções e aqueles relacionados com sistemas dinâmicos, a figura 5.30 ilustra o domínio de definição referente aos dados de treinamento e teste de um PMC aplicado num problema de aproximação de funções, os quais são delimitados pelos valores mínimos  $\{x_i^{\min}\}$  e máximos  $\{x_i^{\max}\}$  associados a cada uma de suas variáveis de entrada. Nesta situação, o domínio de operação em que a efetiva aplicação da rede PMC estará sujeita, após ter sido treinada, coincide com o seu domínio de definição, pois as respostas a serem produzidas sempre estarão em função de valores de entrada compreendidos entre  $x_i^{\min}$  e  $x_i^{\max}$ .

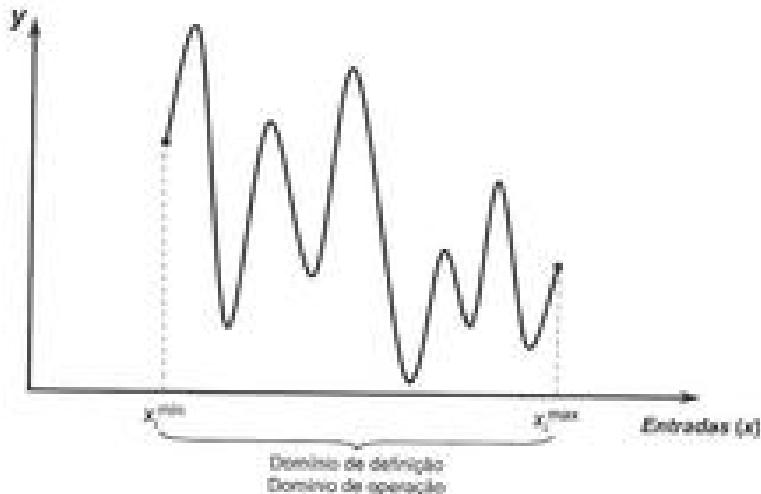


Figura 5.30 – Domínios de definição e de operação de PMC aplicado em problemas de aproximação de funções

Já a figura 5.31 mostra os domínios de definição e de operação associados a um PMC que estará mapeando um problema de sistema dinâmico. Observa-se nesta circunstância que ambos os domínios são regidos pelo tempo, sendo que o domínio de operação se inicia após o seu domínio de definição. Decerto, como nos sistemas dinâmicos a saída atual depende das saídas e entradas anteriores, utiliza-se então os dados de treinamento e teste para ajustar os parâmetros internos da rede. Em seguida, esta estará apta para estimar valores futuros que estarão pertencendo ao seu domínio de operação.

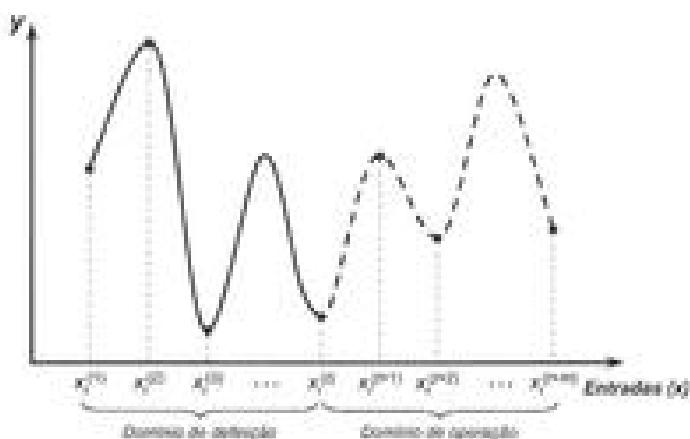


Figura 5.31 – Dominios de definição e de operação de PMC aplicado em problemas envolvendo sistemas dinâmicos

Assim, em se tratando de aplicação de redes PMC no mapeamento de problemas envolvendo sistemas dinâmicos, duas configurações principais podem ser utilizadas, ou seja, a rede PMC com entradas atrasadas no tempo (*TDNN – time delay neural network*) e a PMC com saídas recorrentes às entradas.

Além disso, deve-se também considerar, em contrapartida às redes PMC de duas ou mais camadas escondidas, que aquelas compostas de uma única camada escondida são normalmente menos propensas a estacionar em mínimos locais, pois sua estrutura mais compacta reduz a complexidade geométrica da função que mapeia o erro quadrático médio [Curry & Morgan, 2006; Xiang *et alii*, 2005].

#### (A) Rede PMC de entradas atrasadas no tempo

As redes PMC de entradas atrasadas no tempo, idealizadas pioneiramente por Lang & Hinton (1988), são enquadradas dentro da arquitetura *feedforward* de camadas múltiplas, inexistindo qualquer realimentação das saídas de neurônios de camadas posteriores em direção aos neurônios da primeira camada.

A previsão ou predição de valores posteriores a partir do instante  $t$ , associados ao comportamento do processo, é computada em função do conhecimento de seus valores temporariamente anteriores, isto é:

$$x(t) = f(x(t-1), x(t-2), \dots, x(t-n_p)) \quad (5.61)$$

onde  $n_p$  é a ordem do *preditor*, ou seja, a quantidade de medidas (amostras) passadas que serão necessárias para a estimativa do valor  $x(t)$ . Em terminologia da área de identificação de sistemas, o modelo apresentado em (5.61) é também conhecido como Auto-Regressivo (AR), cuja função  $f_r$  estará sendo implementada pela rede PMC.

Assim, considerando-se a expressão (5.61), uma rede PMC a ser aplicada em processos variantes no tempo teria configuração topológica similar àquela ilustrada na figura 5.32. Diferentemente da TDNN originalmente concebida [Waibel *et alii*, 1989], em que atrasos temporais estão inseridos em todas as camadas da rede, a configuração ilustrada na figura 5.32 estará introduzindo linha de atrasos de tempo somente na camada de entrada, sendo que tal arranjo topológico é também denominado de TDNN com configuração focada ou concentrada (*focused time-delayed feedforward network*) [Haykin, 1999]. Neste caso, a linha contendo os atrasos de tempo funciona como uma memória, garantindo que amostras anteriores que refletem o comportamento temporal do processo sejam sempre inseridas dentro da rede, sem, contudo, haver a necessidade de realimentação de suas saídas.

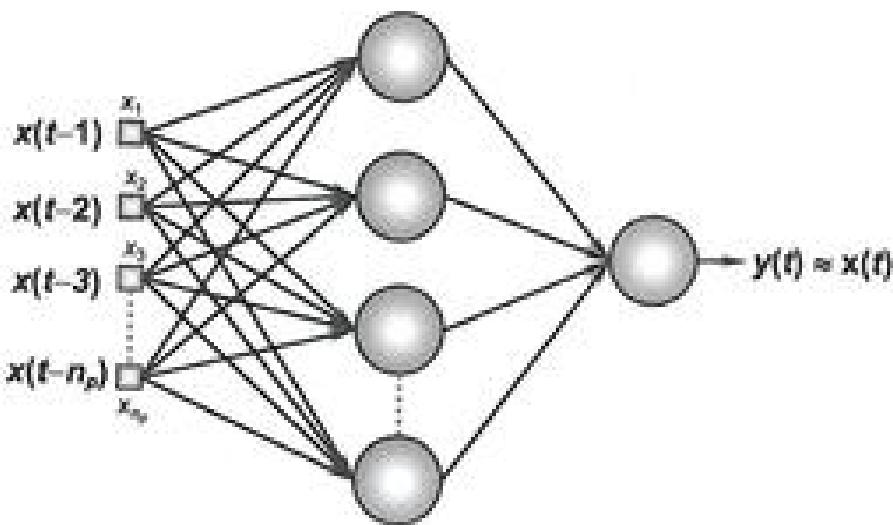


Figura 5.32 – Topologia de PMC com entradas atrasadas no tempo

A partir ainda da ilustração, verifica-se que a rede recebe as  $n_p$  entradas  $\{x(t-1), x(t-2), \dots, x(t-n_p)\}$ , as quais estão representando o comportamento do processo, e prediz então como resposta o respectivo valor esperado para

$x(t)$ , cujo resultado é explicitado pelo valor  $y(t)$  fornecido pelo seu neurônio de saída. Assim, durante o processo de treinamento, a rede tentará ajustar as suas matrizes de pesos visando minimizar o erro  $E(t)$  produzido pela diferença entre  $x(t)$  e  $y(t)$ . Em termos matemáticos, tem-se:

$$E(t) = x(t) - y(t), \text{ onde } (n_p + 1) \leq t \leq N \quad (5.62)$$

onde  $N$  é a quantidade total de medidas (amostras) disponíveis e que foram sequencialmente coletadas ao longo do tempo.

O treinamento da rede PMC com entradas atrasadas no tempo é similar ao PMC convencional e o processo de aprendizado, por sua vez, é também efetuado de maneira idêntica. Os cuidados que ora devem ser levados em conta estão associados com a montagem do conjunto de treinamento da rede. Para elucidar tal mecanismo, considera-se que para um determinado sistema dinâmico foram colhidas as seguintes oito medidas ( $N = 8$ ) ao longo do tempo:

$$x(t) = \begin{bmatrix} t=1 & t=2 & t=3 & t=4 & t=5 & t=6 & t=7 & t=8 \\ x(1) & x(2) & x(3) & x(4) & x(5) & x(6) & x(7) & x(8) \end{bmatrix}^T \quad (5.63)$$

Por intermédio da figura 5.32, assumindo-se ainda que o processo possa ser mapeado com uma ordem de predição igual a três ( $n_p = 3$ ), ter-se-á então para o PMC com entradas atrasadas no tempo um conjunto de treinamento composto por um total de cinco amostras, pois, conforme (5.62), o parâmetro  $t$  variará de quatro até oito, como ilustrado no quadro seguinte.

	Relação entradas/saídas			Conjunto de treinamento					
	$x_1$	$x_2$	$x_3$	Saída desejada	$x_1$	$x_2$	$x_3$	$d$	
$t = 4$	$x(3)$	$x(2)$	$x(1)$	$x(4)$	$x^{(1)}$	0,53	0,32	0,11	$d^{(1)} = 0,17$
$t = 5$	$x(4)$	$x(3)$	$x(2)$	$x(5)$	$x^{(2)}$	0,17	0,53	0,32	$d^{(2)} = 0,98$
$t = 6$	$x(5)$	$x(4)$	$x(3)$	$x(6)$	$x^{(3)}$	0,98	0,17	0,53	$d^{(3)} = 0,67$
$t = 7$	$x(6)$	$x(5)$	$x(4)$	$x(7)$	$x^{(4)}$	0,67	0,98	0,17	$d^{(4)} = 0,83$
$t = 8$	$x(7)$	$x(6)$	$x(5)$	$x(8)$	$x^{(5)}$	0,83	0,67	0,98	$d^{(5)} = 0,79$

$4 \leq t \leq 8$   
 $\longleftrightarrow$   
 $(\text{ordem } 3)$

$n_p = 3$

O valor  $x_0 = -1$ , associado ao limitat do neurônio, deverá ser considerado em todas essas amostras.

Na realidade, procede-se no vetor  $\mathbf{x}(t)$ , em (5.63), uma operação de janela deslizante de largura  $n_x$ , movimentando-se a mesma de uma unidade para a direita em cada iteração de tempo. O quadro a seguir mostra tal mecanismo.

$$\mathbf{x}(t) = [(-1) \quad 0,11 \quad 0,32 \quad 0,53 \quad 0,17 \quad 0,98 \quad 0,67 \quad 0,83 \quad 0,79]^T$$

janela 1 ( $t = 4$ )

$$\mathbf{x}(t) = [(-1) \quad 0,11 \quad 0,32 \quad 0,53 \quad 0,17 \quad 0,98 \quad 0,67 \quad 0,83 \quad 0,79]^T$$

janela 2 ( $t = 5$ )

$$\mathbf{x}(t) = [(-1) \quad 0,11 \quad 0,32 \quad 0,53 \quad 0,17 \quad 0,98 \quad 0,67 \quad 0,83 \quad 0,79]^T$$

janela 3 ( $t = 6$ )

$$\mathbf{x}(t) = [(-1) \quad 0,11 \quad 0,32 \quad 0,53 \quad 0,17 \quad 0,98 \quad 0,67 \quad 0,83 \quad 0,79]^T$$

janela 4 ( $t = 7$ )

$$\mathbf{x}(t) = [(-1) \quad 0,11 \quad 0,32 \quad 0,53 \quad 0,17 \quad 0,98 \quad 0,67 \quad 0,83 \quad 0,79]^T$$

janela 5 ( $t = 8$ )

Após o treinamento da rede, basta realizar a inserção de amostras anteriores da série a fim de se executar a predição de seus valores futuros (posteiros), considerando-se agora o seu domínio de operação  $\{t \geq 9\}$ . Como exemplo, tem-se:

Predição de valores futuros

	$x_i$	$x_j$	$x_k$	saída estimada
$t = 9$	$x(8)$	$x(7)$	$x(6)$	$x(9) = y(9)$
$t = 10$	$x(9)$	$x(8)$	$x(7)$	$x(10) = y(10)$
$t = 11$	$x(10)$	$x(9)$	$x(8)$	$x(11) = y(11)$
(...)	(...)	(...)	(...)	(...)

A partir da análise deste quadro, verifica-se que, para a predição do comportamento futuro do processo frente ao primeiro instante do domínio de operação ( $t = 9$ ), basta-se inserir na entrada da rede os três últimos valores da série ( $x(8), x(7), x(6)$ ) para se obter uma estimativa de  $x(9)$ , representada pelo valor  $y(9)$  de seu neurônio de saída. Consequentemente, para se estimar o valor de  $x(10)$ , deve-se então utilizar os dois últimos valores originais da série ( $x(8), x(7)$ ) e mais o valor estimado de  $x(9)$ , que foi produzido no instante anterior.

Assim, conclui-se que a rede sempre realiza uma predição de um passo à frente, calculando sequencialmente o seu valor atual ou futuro a partir de seus três últimos valores, quando se assume uma ordem de predição igual a três ( $n_p = 3$ ). Contudo, há situações em que a ordem de predição deve ser incrementada a fim de assegurar uma maior precisão na estimativa do comportamento futuro do processo. Como exemplo, para a ação de se utilizar uma ordem de predição igual a quatro ( $n_p = 4$ ), o próximo valor obtido pela rede estaria em função dos quatro últimos valores, sendo que sua topologia para este caso seria igualmente composta de quatro entradas.

#### (B) Rede PMC de saídas recurrentes às entradas

Diferentemente das redes PMC com linha de entradas atrasadas no tempo, a arquitetura com saídas recurrentes às entradas possibilitam a recuperação de respostas passadas a partir da realimentação de sinais produzidos em instantes anteriores. Desta forma, pode-se dizer que tais topologias possuem memória, sendo capazes de "relembra" saídas passadas a fim de produzir a resposta atual ou futura.

De acordo com as definições sobre arquiteturas neurais apresentadas na seção 2.2, tais redes pertencem à classe de arquiteturas recurrentes ou re-alimentadas. A predição de valores futuros associados ao comportamento do processo, a partir do instante  $t$ , será também baseada em função dos valores anteriores que foram produzidos por suas saídas, isto é:

$$\hat{x}(t) = f(x(t-1), x(t-2), \dots, x(t-n_p), y(t-1), y(t-2), \dots, y(t-n_q)) \quad (5.64)$$

onde  $n_p$  é a ordem do *preditor* e indica o número de medidas (amostras) anteriores que serão necessárias para a estimativa de  $\hat{x}(t)$ . O valor  $n_q$  expressa a ordem de contexto, ou seja, a quantidade de saídas passadas que serão também utilizadas na estimativa de  $\hat{x}(t)$ . Nesta condição, o papel desempenhado pela rede, após o treinamento da mesma, seria representar de forma indireta a função  $f(\cdot)$ , a qual estará incumbida de identificar o relacionamento entre as entradas e saídas do sistema. Por conseguinte, a figura 5.33 ilustra uma representação de PMC recorrente que implementa o processo dinâmico explicitado em (5.64).

Verifica-se ainda, por meio da análise da referida figura, que os sinais rotulados como unidades de contexto executam, simplesmente, a tarefa de realimentar aos neurônios da primeira camada todas aquelas  $n_y$  respostas passadas que foram produzidas pelo neurônio de saída em seus instantes anteriores.

Tal configuração possibilita que a rede PMC recorrente execute, de maneira implícita, o mapeamento entre entradas e saídas de processos que sejam tanto não-lineares como também variantes no tempo, tornando-se uma ferramenta bem flexível para aplicações envolvendo identificação de sistemas.

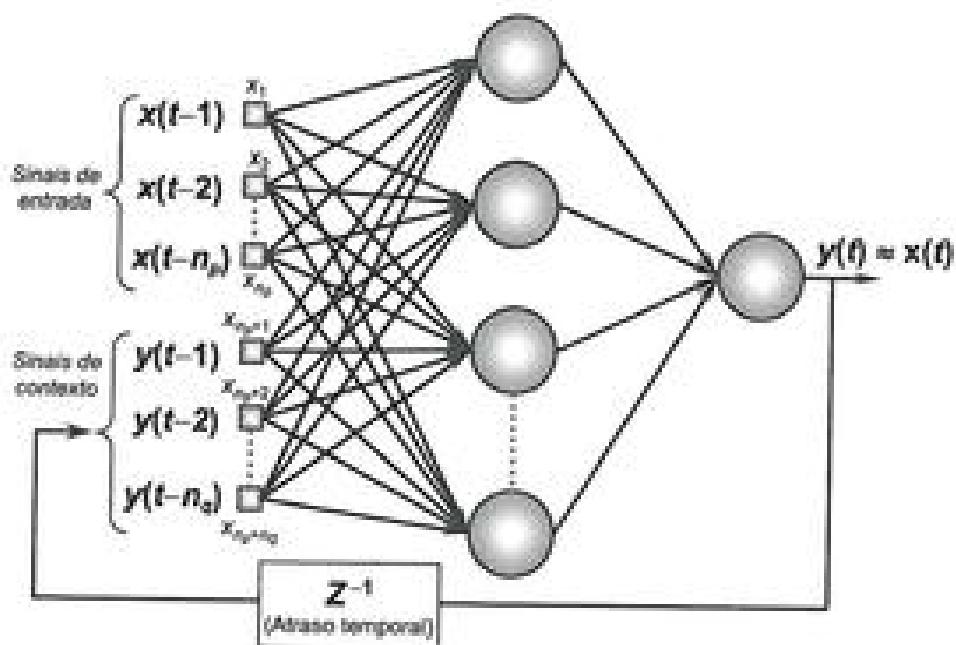


Figura 5.33 – Topologia de PMC com saídas realimentadas às entradas

De fato, em terminologia da área de identificação de sistemas, a rede PMC recorrente que estariá implementando o relacionamento representado em (5.64) funciona como um modelo auto-regressivo não-linear com entradas exógenas (*Nonlinear Auto-Regressive with eXogenous inputs* – NARX), cuja aplicabilidade direciona-se ao mapeamento de sistemas com dinâmicas tipicamente não-lineares [Nelles, 2005].

Assim como no PMC com entradas atrasadas no tempo, o treinamento do PMC recorrente é também efetuado de maneira similar ao PMC convencional, que foi tratado tanto para mapeamento de problemas de classificação

de padrões como para aproximação funcional. Logo, baseando-se ainda na expressão (5.62), o treinamento do PMC recorrente estará então promovendo os ajustes necessários em suas matrizes de pesos visando minimizar o erro  $E(f)$  entre o valor esperado de  $x(t)$  frente à resposta  $y(t)$  estimada pela rede.

Desta forma, assumindo-se a sequência temporal fornecida em (5.63), e considerando ainda a topologia apresentada na figura 5.33, o conjunto de treinamento do PMC recorrente para  $n_x = 3$  (ordem de predição usando três entradas passadas) e  $n_y = 2$  (ordem de contexto utilizando duas saídas passadas) seria constituído por:

Relação entre as entradas/saídas							Conjunto de treinamento					
	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	Saída desejada	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$d$
$t = 4$	$x(2)$	$x(3)$	$x(4)$	0	0	$y(4)$	$x(1)$	$x(2)$	$x(3)$	$x(4)$	$x(5)$	$d^1 = 0,17$
$t = 5$	$x(3)$	$x(4)$	$x(5)$	$y(4)$	0	$y(5)$	$x(2)$	$x(3)$	$x(4)$	$x(5)$	$y(6)$	$d^2 = 0,08$
$t = 6$	$x(4)$	$x(5)$	$x(6)$	$y(5)$	$y(4)$	$x(6)$	$x(3)$	$x(4)$	$x(5)$	$x(6)$	$y(7)$	$d^3 = 0,07$
$t = 7$	$x(5)$	$x(6)$	$x(7)$	$y(6)$	$y(5)$	$x(7)$	$x(4)$	$x(5)$	$x(6)$	$x(7)$	$y(8)$	$d^4 = 0,03$
$t = 8$	$x(6)$	$x(7)$	$x(8)$	$y(7)$	$y(6)$	$x(8)$	$x(5)$	$x(6)$	$x(7)$	$x(8)$	$y(9)$	$d^5 = 0,28$

O valor  $x_0 = -1$ , associado ao limiar neural, deverá também ser inserido em todos os neurônios da rede.

Conforme observado no quadro anterior, considerando-se o primeiro instante  $\{t = 4\}$ , o PMC recorrente utilizará apenas os sinais de entradas atrasadas no tempo, pois a realimentação dos dois últimos valores produzidos pela rede é nula em virtude da inexistência de saídas passadas. Entretanto, no próximo instante  $\{t = 5\}$ , o primeiro valor produzido pela saída da rede ( $y(4)$ ), obtido no instante anterior, estará já disponível. Successivamente, para o instante posterior  $\{t = 6\}$ , as duas saídas passadas ( $y(5)$ ,  $y(4)$ ), que já foram produzidas pela rede nos dois instantes anteriores, são então introduzidas em suas entradas. Tal processo se repete para todos os instantes posteriores visando a compilação do conjunto de treinamento do PMC recorrente.

A predição de valores futuros relacionados ao comportamento do processo, a ser realizada após o treinamento da rede, será similarmente efetuada usando aqueles mesmos valores de  $n_x$  e  $n_y$  que foram assumidos no aprendizado. Para o caso do exemplo adotado, considerando o seu domínio de operação  $\{t \geq 9\}$ , tem-se o seguinte quadro:

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	Saída estimada
$t = 9$	$x(8)$	$x(7)$	$x(6)$	$y(8)$	$y(7)$	$x(9) \approx y(9)$
$t = 10$	$x(9)$	$x(8)$	$x(7)$	$y(9)$	$y(8)$	$x(10) \approx y(10)$
$t = 11$	$x(10)$	$x(9)$	$x(8)$	$y(10)$	$y(9)$	$x(11) \approx y(11)$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

Assim, considerando-se quaisquer instantes futuros do domínio de operação do PMC recorrente aqui usado, a predição de seus valores sempre levará em conta tanto as três últimas entradas atrasadas no tempo como as duas últimas saídas produzidas pela rede.

Esta configuração topológica, em que apenas os resultados produzidos pelos neurônios de saída da rede são realimentados às suas entradas, é também conhecida como rede de Elman ou rede recorrente simples [Elman, 1990]. Por outro lado, a configuração poderá também ser denominada de rede de Jordan, quando somente os resultados dos neurônios das camadas intermediárias são realimentados para as suas unidades de contexto [Jordan, 1986], produzindo, por sua vez, sinais realimentados considerados semi-recorrentes (localmente).

Além da utilização como estimador de comportamento futuro de processos envolvendo sistemas dinâmicos, o PMC recorrente pode ser convertido em diversas configurações que tem visado sua elevada aplicabilidade na temática de sistemas de controle. Um dos trabalhos pioneiros nesta direção foi realizado por Narendra & Parthasarathy (1990), sendo que investigações mais abrangentes e detalhadas encontram-se relatadas em Leondes (2006), Norgaard *et alii* (2006) e Suykens *et alii* (2001).

## 5.5 – Aspectos de especificação topológica de redes PMC

A especificação da topologia de rede PMC mais apropriada para mapear um problema específico é usualmente efetuada de forma empírica, pois tal dimensionamento depende, entre outros fatores, do algoritmo de aprendizado utilizado, da forma como as matrizes de pesos foram iniciadas, da complexidade do problema a ser mapeado, da disposição espacial das amostras e, por sua vez, da qualidade do conjunto de treinamento disponível. Este último fator pode ser sensivelmente afetado em função da quantidade de ruídos presentes nas amostras, da existência de erros estruturais e da presença de amostras espúrias (*outliers*), dentre outros.

Como exemplo ilustrativo, considera-se que para um determinado problema tem-se quatro topologias candidatas de PMC, constituídas de apenas uma camada escondida, e que podem ser capazes de mapear o seu comportamento. São as seguintes:

Topologia candidata 1 → 5 neurônios na camada escondida.

Topologia candidata 2 → 10 neurônios na camada escondida.

Topologia candidata 3 → 15 neurônios na camada escondida.

Topologia candidata 4 → 20 neurônios na camada escondida.

O objetivo agora colocado está em definir qual seria a mais indicada para executar o mapeamento do referido problema.

Uma das técnicas estatísticas mais utilizadas para seleção das melhores topologias candidatas é a validação cruzada (*cross-validation*) [Kohavi, 1995; Ripley, 1996], cujo propósito é avaliar a aptidão de cada uma quando aplicadas a um conjunto de dados que seja diferente daquele usado no ajuste de seus parâmetros internos. Conforme apresentado na subseção seguinte, três métodos de validação cruzada são geralmente empregados no processo de seleção de redes PMC.

### 5.5.1 – Aspectos de métodos de validação cruzada

O primeiro método é denominado de validação cruzada por amostragem aleatória (*random subsampling cross-validation*), em que o conjunto total de dados (amostras) disponíveis é aleatoriamente dividido em duas partes, isto é, subconjunto de treinamento e subconjunto de teste (validação).

Mais especificamente, o subconjunto de treinamento, conforme o próprio nome sinaliza, será utilizado para treinar todas as topologias candidatas, sendo que o subconjunto de teste é somente aplicado para selecionar aquela que estará apresentando os melhores resultados de generalização. Com efeito, como as amostras do subconjunto de teste não participaram do processo de aprendizado, torna-se então possível avaliar o desempenho da generalização proporcionada em cada uma das topologias candidatas. Para tanto, basta se comparar os resultados produzidos em suas saídas frente aos respectivos valores desejados.

Em termos práticos, a partir do conjunto total de dados disponíveis, cerca de 60 a 90% são aleatoriamente escolhidos para o subconjunto de treinamento,

enquanto que o restante ficará alocado ao subconjunto de teste. Esta sistemática de partição deve ser repetida várias vezes durante o processo de aprendizado das topologias candidatas, permitindo-se (em cada ensaio) a possibilidade de contemplação de amostras diferentes tanto no subconjunto de treinamento como no de teste. O desempenho global de cada topologia candidata será então compilado a partir da média dos desempenhos individuais em cada experimento.

A figura 5.34 ilustra a estratégia envolvida com a seleção de amostras de treinamento e de teste que serão alocadas nos respectivos subconjuntos. Para cada um dos cinco ensaios que foram didaticamente assumidos, escolhem-se aleatoriamente, dentre todas as amostras disponíveis (dezoito), aquelas (seis) que pertencerão ao subconjunto de teste, sendo que todas as demais (doze) serão alocadas ao subconjunto de treinamento. Em cada experimento em questão, o sorteio das amostras para o subconjunto de teste será sempre realizado considerando todas as amostras disponíveis, independentemente de algumas já terem sido sorteadas em ensaios anteriores.

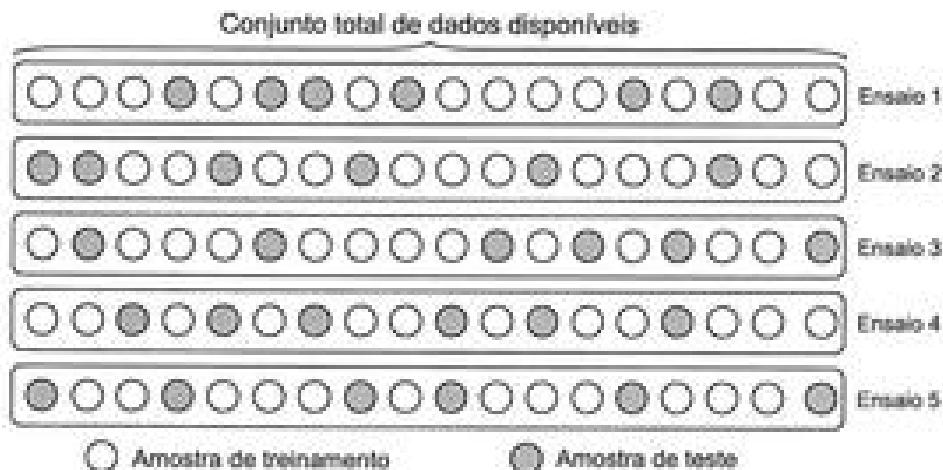


Figura 5.34 – Método de validação cruzada usando amostragem aleatória

O segundo método de validação cruzada utilizada para dimensionamento estrutural de redes PMC é denominado de *k*-partições (*k-fold cross-validation*). Realiza-se aqui a divisão do conjunto total de amostras em *k* partições, sendo que (*k* - 1) delas serão usadas para compor o subconjunto de treinamento, ao passo que a partição restante constituirá o subconjunto de teste.

Por conseguinte, o processo de aprendizado se repete *k* vezes até que todas as partições tenham sido utilizadas como subconjunto de teste. A figura 5.35 sistematiza o mecanismo desta estratégia para um total de 20 amostras,

quando se assume um valor de  $k$  igual a 5, que implica necessariamente na realização de um número igual de ensaios.

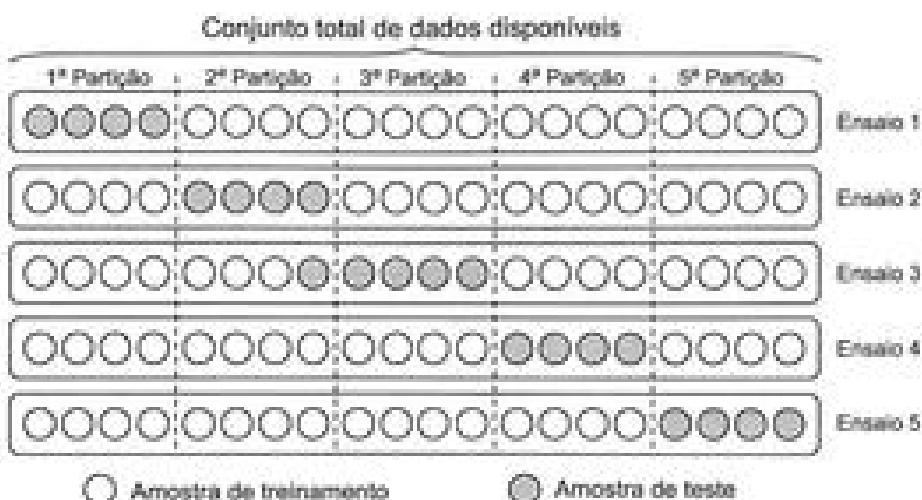


Figura 5.35 – Método de validação cruzada usando  $k$ -partições

O valor do parâmetro  $k$  está atrelado à quantidade total de amostras disponíveis, sendo usualmente atribuído um número compreendido entre 5 e 10. O desempenho global de cada topologia candidata será agora também obtido em função da média entre os desempenhos individuais observados quando da aplicação das  $k$ -partições.

Finalmente, o terceiro método (menos usual) é chamado de validação cruzada por unidade (*one-one-unit cross-validation*), que consiste na utilização de uma única amostra para o subconjunto de teste, sendo as demais alocadas para o subconjunto de treinamento. O processo de aprendizado é então repetido até que todas as amostras sejam individualmente utilizadas como subconjunto de teste.

Na realidade, esta última técnica acaba sendo um caso particular do método de  $k$ -partições, pois se basta apenas atribuir ao parâmetro  $k$  o valor que corresponde ao número total de amostras disponíveis. Contudo, tem-se aqui um elevado esforço computacional, pois o processo de aprendizagem será repetido, considerando cada uma das topologias candidatas, um número de vezes que será igual ao tamanho do conjunto total de amostras. A figura 5.36 sintetiza esta estratégia de validação cruzada para um total de 20 amostras disponíveis.



Figura 5.36 – Método de validação cruzada por unidade

A sequência de passos algorítmicos envolvida com a implementação de quaisquer uns desses três métodos de validação cruzada está como se segue.

#### Início (Algoritmo Validação Cruzada)

- <1> Definir para o problema as topologias candidatas de PMC;
- <2> Disponibilizar os subconjuntos de treinamento e de teste;
- <3> Aplicar o algoritmo de treinamento do PMC para todas as topologias candidatas usando os subconjuntos de treinamento;
- <4> Aplicar os subconjuntos de teste nas topologias candidatas (já treinadas) visando avaliar os potenciais de generalização;
- <5> Obter o desempenho final de cada topologia candidata em função do número de ensaios utilizados;
- <6> Selecionar aquela topologia candidata que obteve o melhor desempenho global;
- <7> Se o desempenho global desta melhor topologia candidata estiver de acordo com a precisão requerida ao problema,
  - <7.1> Então: Terminar o processo de validação cruzada.
  - <7.2> Senão: Especificar novo conjunto de topologias candidatas e voltar ao passo <3>.

#### Fim (Algoritmo Validação Cruzada)

Em contraste aos três métodos apresentados, na situação de se aplicar todas as amostras disponíveis tanto na fase de aprendizado como na de va-

Isolação, torna-se então impossível verificar a capacidade de generalização da rede, sendo tão somente possível de se avaliar a sua habilidade em memorizar satisfatoriamente quais seriam as respostas desejadas frente aos padrões apresentados.

Alternativamente aos métodos de *cruz-validação*, duas das regras muito utilizadas em especificação topológica, considerando um PMC constituído de apenas uma única camada escondida, são dadas por:

$$n_1 = 2 \cdot n + 1 \quad \{ \text{método de Kolmogorov} \} \quad (5.65)$$

$$2 \cdot \sqrt{n} + n_2 \leq n_1 \leq 2 \cdot n + 1 \quad \{ \text{método de Fletcher-Gloss} \} \quad (5.66)$$

onde  $n$  é o número de entradas da rede,  $n_1$  é a quantidade de neurônios na camada escondida e  $n_2$  é a quantidade de neurônios em sua camada de saída. Em particular, para problemas de classificação de padrões com  $n_c$  classes, outra heurística que tem sido comumente utilizada é aquela implementada na plataforma *Weka* (*Waikato environment for knowledge analysis*) [Witten & Flank, 2005], ou seja:

$$n_1 = \frac{n + n_c}{2} \quad (5.67)$$

Tais regras, embora ainda muito utilizadas, são apropriadas tão somente para alguns tipos de problemas bem comportados, pois desconsideram atributos que são de fato muito relevantes para propósitos de especificação topológica de redes PMC, tais como a quantidade de dados, a complexidade do problema, a qualidade dos dados e suas disposições no espaço amostral.

### 5.5.2 – Aspectos de subconjuntos de treinamento e teste

Um dos cuidados que se deve tomar no momento da compilação dos subconjuntos de treinamento é assegurar que todas as amostras, que carregam os valores mínimos e máximos de cada variável de entrada, estejam também dentro desses subconjuntos. Caso contrário, se tais valores forem

inadvertidamente alocados aos subconjuntos de teste, o PMC poderia gerar erros significativos, pois estaria tentando generalizar valores que estão fora dos domínios de definição de suas variáveis de entrada.

De fato, deve-se sempre ter em mente que o PMC, quando projetado para problemas de aproximação funcional ou de reconhecimento de padrões, vai continuamente desconhecer qual é o comportamento do processo fora de seus domínios de definição, pois, no decorrer do aprendizado, utilizaram-se somente amostras advindas destes.

Consequentemente, durante também toda a fase de operação, deve-se ainda garantir que os atuais sinais, referentes a cada uma das variáveis de entrada, estejam novamente compreendidos dentro daqueles domínios de definição que foram obtidos a partir dos valores mínimos e máximos dos subconjuntos de treinamento.

Como ilustração desses aspectos de projeto, a figura 5.37 mostra uma situação em que o PMC foi treinado para mapear a função seno, cujas amostras de treinamento pertenciam ao domínio especificado entre 0 e 10.

Inspeccionando a referida figura, constata-se que o PMC estimou de forma precisa somente os valores da função seno que estavam compreendidos dentro do respectivo domínio de definição.



Figura 5.37 – Aplicação do PMC considerando pontos situados dentro e fora de seu domínio de definição

Entretanto, os pontos que formam os segmentos de curva fora desse domínio resultam em valores de estimação bem imprecisos. Certamente, con-

forme comentado anteriormente, tal fato ocorre em virtude de o PMC ignorar o comportamento da função tanto à esquerda como à direita do domínio de definição, pois nenhuma amostra pertencente a esses segmentos estava presente nos subconjuntos de treinamento.

### 5.5.3 – Aspectos de situações de *overfitting* e *underfitting*

Cabe ressaltar que o aumento indiscriminado de neurônios assim como o incremento de camadas intermediárias não asseguram a generalização apropriada do PMC em relação às amostras pertencentes aos subconjuntos de teste.

Invariavelmente, tais ações prematuras tendem a levar a saída do PMC para a circunstância de memorização excessiva (*overfitting*), em que este acaba decorrente as suas respostas frente aos estímulos introduzidos em suas entradas. Nessas ocorrências, o erro quadrático durante a fase de aprendizado tende a ser bem baixo; contudo, durante a fase de generalização frente aos subconjuntos de teste, o erro quadrático tende a assumir valores bem elevados, fato que denota a condição de *overfitting*.

A figura 5.38 mostra um comportamento em que o PMC está operando numa situação de *overfitting* (topologia 1), tendo-se, em contraste, outra situação em que o PMC está generalizando de forma bem satisfatória (topologia 2), ou seja, sem *overfitting*.



Figura 5.38 – Comportamento do PMC operando em situação de *overfitting*.

Percobe-se na ilustração que a topologia 1 (com *overfitting*), contendo,

por exemplo, um total de 20 neurônios, produzirá um erro quadrático durante a fase de aprendizado que será certamente bem pequeno, pois as amostras de treinamento estão quase que coincidentes com a curva mapeada pela saída da rede. Entretanto, em relação às amostras de teste, a topologia 1 gerará um erro significativo, pois estas estão agora mais distantes. Sinteticamente, apropriando da terminologia usada em identificação de sistemas, pode-se dizer que a topologia 1 está se comportando de maneira parecida a um estimador viciado (tendencioso), cujos valores produzidos são forçosamente muito divergentes daqueles que seriam esperados [Ljung, 1998].

Diferentemente, a topologia 2 (sem overfitting), constituída, por exemplo, de 10 neurônios, fornecerá um erro bem menor frente às amostras de teste, pois a sua curva de saída está representando mais fidedignamente o comportamento do processo que foi mapeado. Assim sendo, esta procede então como um estimador não-viciado (não tendencioso), pois as suas respostas estão dentro de uma margem de erro considerada aceitável.

Outro gráfico bem ilustrativo de overfitting é mostrado na figura 5.39, em que o PMC está operando com memorização excessiva com o objetivo de mapear a função seno (afetada por ruidos).

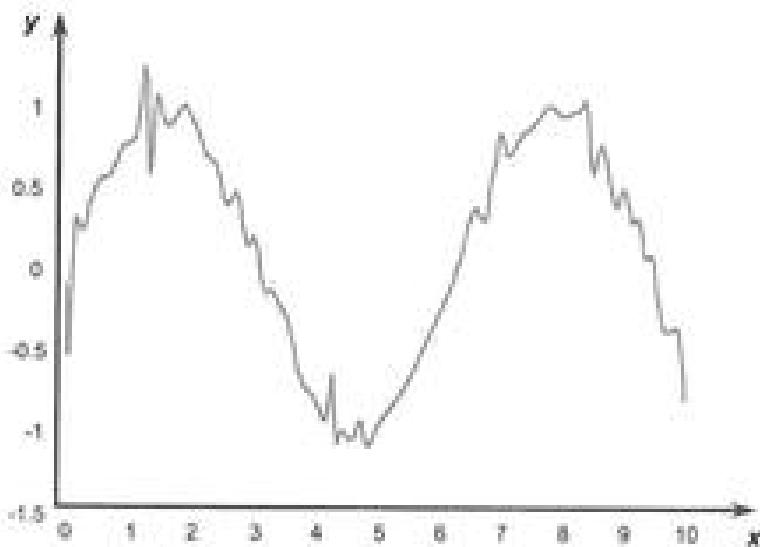


Figura 5.39 – PMC usado para mapear a função seno (com overfitting).

Em contraste, a curva gerada pelo PMC da figura 5.40 (sem overfitting), quando treinado com as mesmas amostras utilizadas na figura 5.39, produzirá

provavelmente respostas de generalização que seriam bem mais satisfatórios, visto que seu formato geométrico ficou mais próximo daquele esperado para a função seno. Além disso, o PMC em questão acabou naturalmente desconsiderando os ruidos, pois decidiu de realizar uma memorização extremamente excessiva ao ponto de inclui-los.

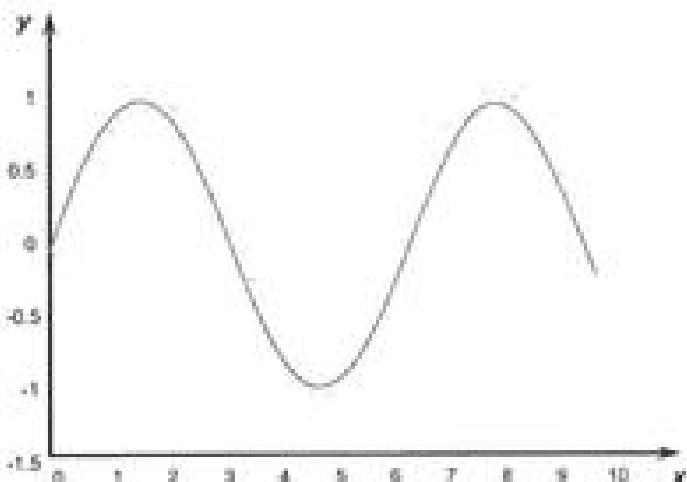


Figura 5.40 – PMC usado para mapear a função seno (*sem overfitting*)

Em contrapartida, frente à precisão requerida, uma topologia de PMC com número muito reduzido de neurônios pode ser insuficiente para a extração e armazenamento de características que permitam à rede implementar as hipóteses a respeito do comportamento do processo, configurando-se aqui uma situação de debilidade neural denominada de *underfitting*. Nesses casos, por sua vez, o erro quadrático tanto na fase de aprendizado como na fase de teste serão bem significativos.

#### 5.5.4 – Aspectos de inclusão de parada antecipada

Um procedimento muito simples que tem sido também inserido na implementação da técnica *cross-validation* é a parada antecipada ou prematura (*early stopping*), em que o processo de aprendizagem para uma topologia candidata é constantemente checado pela aplicação dos subconjuntos de teste, sendo finalizado quando começar a haver elevação do erro quadrático (frente aos subconjuntos de teste) entre épocas sucessivas.

De fato, esta variação repentina sinaliza a tentativa de extração excessi-

va de características dos subconjuntos de treinamento, por exemplo, os próprios ruídos de medição [Finnoff *et al.*, 1993]. No exemplo na figura 5.41, o processo de aprendizado terminaria na décima época de treinamento.

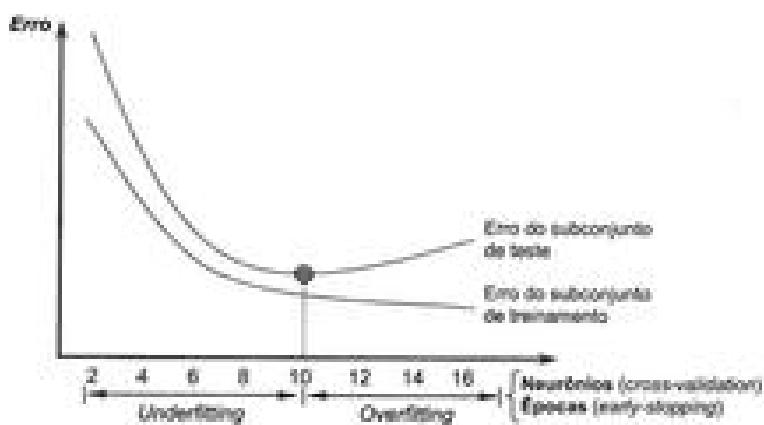


Figura 5.41 – Situações envolvendo *underfitting* e *overfitting*.

Em suma, o processo de especificação de uma topologia de redes neurais artificiais deve levar em consideração o compromisso de superar o *underfitting* e de evitar a condição de *overfitting*. Com o intuito de mostrar tal procedimento, considerando um PMC com apenas uma camada escondida, ilustra-se também na figura 5.41 as situações de *underfitting* e *overfitting* em função do número de neurônios de sua camada escondida.

Ainda mediante a análise da ilustração, tem-se que uma topologia bem apropriada para aquele caso teria um total de 10 neurônios em sua camada escondida, pois um número menor que este acarretaria num erro grande em relação às amostras dos subconjuntos de teste em virtude da insuficiência de neurônios (*underfitting*). Em contraposição, para uma quantidade maior que 10, o PMC começaria também a produzir um erro considerável para as amostras dos subconjuntos de teste, independentemente da constatação de que o erro quadrático frente às amostras dos subconjuntos de treinamento tende a ser cada vez menor, quando da adição de mais neurônios (*overfitting*).

Como última nota, ressalta-se ainda que o método *early stopping*, quando utilizado, deverá ser individualmente aplicado para cada topologia candidata. Entretanto, a seleção da melhor continua atribuída à aplicação da técnica de *cruz-validation*.

### 5.5.5 – Aspectos de convergência para mínimos locais

Em virtude de a superfície de erro produzida pelo PMC ser não-linear, há então a possibilidade de que o processo de aprendizado direcione a matriz de pesos da rede para um ponto de mínimo local, que pode não corresponder aos valores mais apropriados aos propósitos de generalização de resultados.

A figura 5.42 ilustra algumas situações envolvendo pontos de mínimos locais. Na condição de a rede convergir para o ponto indicado em  $p^0$ , certamente, a mesma produziria resultados mais favoráveis que aqueles proporcionados pela sua eventual convergência para  $p^1$  ou  $p^2$ , pois o valor do *Erro* em  $p^0$  seria o menor deles.

A tendência de convergência para um determinado ponto de mínimo fica então condicionada, principalmente, à posição espacial em que a matriz de pesos  $W$  foi iniciada, pois a grande maioria dos algoritmos de aprendizagem é baseada em métodos de gradiente descendente. Para o exemplo da figura 5.42, se a rede tivesse sua matriz de pesos iniciada em  $W^0$ , a tendência de convergência seria para o ponto de mínimo  $p^0$ ; ao passo que se tivesse iniciada em  $W^1$ , a propensão seria em direção a  $p^1$ .

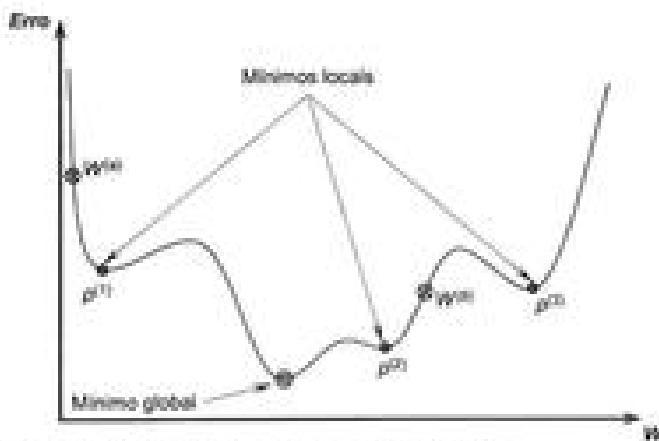


Figura 5.42 – Pontos de mínimos locais associados à função-erro

Visando evitar a convergência da rede para pontos de mínimos locais inapropriados, um dos procedimentos práticos a ser adotado consiste em executar o treinamento de cada topologia candidata mais de uma vez, com diferentes matrizes de pesos iniciais (geradas aleatoriamente). Assim, dependendo de suas posições espaciais, a rede poderia convergir para pontos de

mínimos locais, ou até mesmo globais, que possibilitariam uma melhor representação do comportamento do processo.

Outras possibilidades mais elaboradas, tais como aquelas baseadas em reconhecimento simulado (*simulated annealing*) [Kirkpatrick *et alii*, 1983], em busca tabu [Glover e Laguna, 1998], em algoritmos genéticos [Goldberg, 2002] e em Monte Carlo [Pineus, 1970], podem ser também incorporadas aos algoritmos de aprendizagem como estratégias de escape dos mínimos locais.

### 5.6 – Aspectos de implementação de redes Perceptron multicamadas

Apresenta-se a seguir alguns aspectos práticos referentes à implementação das redes *Perceptron* multicamadas que visam nortear o seu projeto.

Em princípio, a figura 5.43 mostra o diagrama de blocos que ilustram as principais etapas de implementação envolvidas com o projeto dessas redes, considerando-se tanto a fase de treinamento como de operação, que é aplicada somente após a primeira.

Em termos de estruturas de dados computacionais, cada uma pode ser implementada por meio de sub-rotinas (funções e procedimentos), cujas instruções internas podem também ser refinadas ou subgrupadas para propósitos de elevação da eficiência computacional.

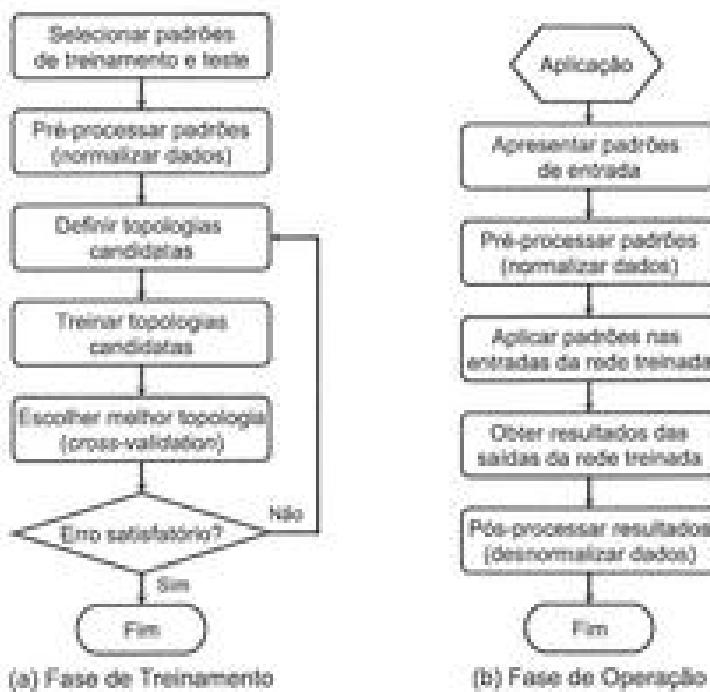


Figura 5.43 – Diagrama de blocos das fases de treinamento e operação

Conforme observado na figura 5.43(a), há a necessidade de pré-processamento dos padrões (amostras) de treinamento e teste visando aspectos de melhoria do desempenho de treinamento.

Tal aspecto implica escalonar as respectivas amostras para a faixa de variação dinâmica das funções de ativação das camadas escondidas, tipicamente representadas pela função logística ou tangente hiperbólica, com o intuito de evitar a saturação dos neurônios (figura 5.9).

Para isto, recomenda-se então escalonar os valores dos sinais de entrada e saída, tanto dos padrões de treinamento como dos padrões de teste, levando-se em consideração aquelas faixas de variações dinâmicas das funções de ativação. Uma das técnicas de escalonamento mais utilizada é aquela baseada no princípio dos segmentos proporcionais (teorema de Tales) ilustrado na figura 5.44, em que um conjunto de valores definidos inicialmente num intervalo fechado entre  $x^{\text{ini}}$  e  $x^{\text{fim}}$ , ou seja,  $x \in [x^{\text{ini}}, x^{\text{fim}}]$ , será convertido para um domínio proporcional entre -1 e 1, o qual pode estar representando as próprias faixas de variações dinâmicas das entradas das funções de ativação.

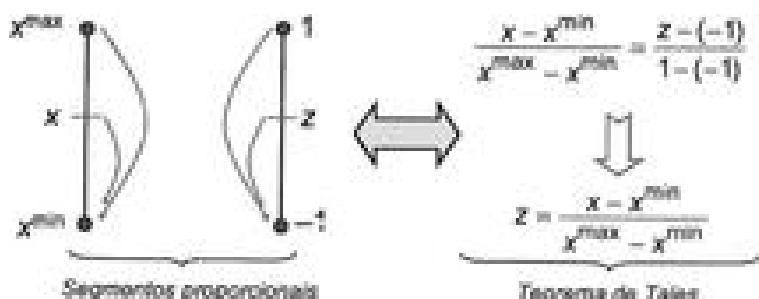


Figura 5.44 – Princípio de normalização dos padrões de treinamento e teste.

Assim, considerando que a função de ativação logística será utilizada nos neurônios das camadas intermediárias, a figura 5.45 ilustra as normalizações a serem realizadas nas entradas e saídas dos padrões de treinamento e teste.

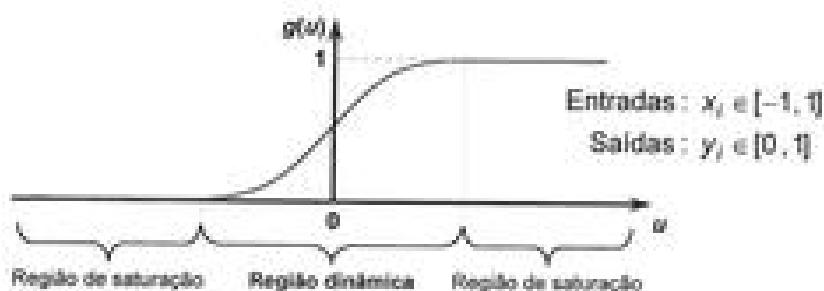


Figura 5.45 – Domínios de normalização para a função de ativação logística

De forma similar, a figura 5.46 descreve as normalizações a serem realizadas, assumindo-se agora que a função de ativação tangente hiperbólica será usada nos neurônios das camadas intermediárias.

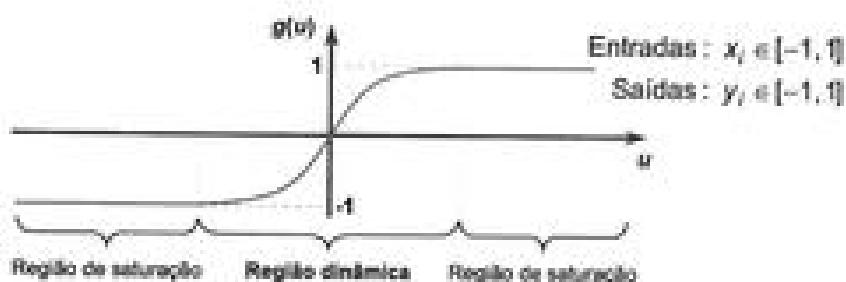


Figura 5.46 – Domínios de normalização para a função de ativação tangente hiperbólica

É importante ressaltar que todas as variáveis de entrada e saída da rede precisam ser individualmente normalizadas, em relação aos seus respectivos valores mínimos e máximos, considerando também todos os dados disponíveis, e assegurando-se ainda que tais valores estejam contidos dentro do conjunto de treinamento. Caso contrário, os valores mínimos e máximos estarão no conjunto de teste, implicando-se então na redução dos domínios referentes às variáveis do conjunto de treinamento.

De forma similar às normalizações efetuadas nos pré-processamentos das amostras, há também a necessidade de se realizar instruções de pós-processamentos quando o PMC já estiver sendo usado na fase de operação, conforme mostra a figura 5.43(b). Nesta situação, executam-se operações de desnormalização a fim de converter as respostas dos neurônios de saída da rede para aqueles valores que representam os domínios reais da aplicação.

Conforme se pôde observar ao longo deste capítulo, inúmeros são os parâmetros passíveis de serem ajustados com o intuito de se melhorar o desempenho do processo de treinamento do PMC. Investigações detalhadas a respeito dessas diversas estratégias são apresentadas em Hagan *et alii* (1996) e Reed & Marks II (1999).

Finalmente, considerando também outros aspectos importantes envolvidos com as fases de treinamento e operação do PMC, apresentam-se as seguintes notas práticas:

- Atentar ao fato de que o aumento de neurônios e de camadas do PMC não implica diretamente em melhoria de seu potencial de generalização;
- Selecionar, considerando-se duas topologias de PMC que estejam generalizando com o mesmo grau de preciso, sempre aquela composta pela menor quantidade de neurônios, pois demonstra que ela foi capaz de extrair mais conhecimento;
- Alocar ao conjunto de treinamento as amostras que contiverem os valores mínimos e máximos relativos a quaisquer variáveis de entrada e saída do PMC;
- Realizar por diversas vezes o treinamento de cada topologia candidata, assumindo-se aleatórios valores iniciais das matrizes de pesos, com o objetivo de escapar de eventuais regiões de mínimos locais;
- Iniciar todas as matrizes de pesos  $W^{(j)}$  com valores aleatórios pequenos.

Uma estratégia bem utilizada é aquela advinda das análises efetuadas em LeCun (1989), cujos valores devem estar compreendidos entre  $(-2,4/N \leq w_j^{(l)} \leq 2,4/N)$ , sendo que  $N$  é o número de entradas do PMC. A estratégia tenta evitar que a soma ponderada dos valores das entradas pelos respectivos pesos sinápticos implique na saturação das funções de ativação;

- Associar valores apropriados aos termos de aprendizagem  $\{\eta\}$  e termos de *momentum*  $\{\alpha\}$ , os quais devem estar compreendidos entre  $(0,05 \leq \eta \leq 0,75)$  e  $(0 \leq \alpha \leq 0,9)$ . Nos exemplos práticos da seção 5.8 são sugeridos alguns valores típicos para esses parâmetros;
- Impor uma quantidade máxima de épocas como critério adicional de parada do algoritmo de treinamento do PMC, pois é uma estratégia simples e eficiente para cessar o treinamento quando a precisão especificada  $\{\varepsilon\}$  se torna inalcançável;
- Adotar para os propósitos de incrementar a rapidez do treinamento do PMC, considerando-se problemas envolvendo aproximação de funções e sistemas variantes no tempo, o uso do método de Levenberg-Marquardt, pois em diversos testes realizados sempre se mostrou o mais rápido. Entretanto, considerando-se problemas envolvendo classificação de padrões, a utilização do método *Back-Propagation* se mostra normalmente a mais rápida, pois há o tratamento direto dos sinais das derivadas do gradiente, sendo este um aspecto importante em problemas de classificação de padrões, em virtude das possíveis saídas desejadas mudarem seu valor de forma abrupta;
- Normalizar os valores de entrada e saída das amostras visando evitar as regiões de saturação das funções de ativação;
- Adotar preferencialmente a tangente hiperbólica como função de ativação para os neurônios das camadas escondidas, pois a sua característica de anti-simetria (função ímpar) contribui para melhorar o processo de convergência da rede durante o respectivo treinamento;
- Assumir sempre os dados dos subconjuntos de testes para avaliação do potencial de generalização;
- Aplicar técnicas de pré-processamento e/ou ferramentas de extração de características (transformada de Fourier, transformada *Wavelet*, análise de componentes principais, etc.) com a finalidade de minimizar redundâncias e reduzir a complexidade dimensional dos sinais de entrada da rede.

## 5.7 – Exercícios

- 1) Explique se é possível realizar o treinamento da rede PMC, por meio do algoritmo *backpropagation*, quando se inicializa todas as matrizes de pesos com elementos nulos. Discorra também se há então alguma implicação quando se inicializa todos os elementos das matrizes de pesos com valores iguais (diferentes de zeros).
- 2) Considerando os problemas envolvendo aproximação de funções, discorra então se há alguma vantagem e/ou desvantagem em se utilizar a função de ativação linear para os neurônios da camada de saída da rede ao invés do uso da tangente hiperbólica.
- 3) Explique o que são situações de *underfitting* e *overfitting*, descrevendo-se também os meios para as suas detecções e as técnicas utilizadas para o seu contorno.
- 4) Discorra sobre as eventuais consequências de se utilizar um valor muito elevado para o termo de aprendizagem ( $\eta$ ) em contraposição a um valor muito pequeno para o termo de *momentum* ( $\alpha$ ).
- 5) Seja um problema de classificação de padrões composto apenas por duas classes, sendo que a respectiva fronteira de separação é representada por uma região convexa compacta (fechada e limitada). Estime então qual seria a quantidade mínima possível de neurônios que poderia estar associado à primeira camada neural escondida.
- 6) Considere para o exercício anterior a situação de que a fronteira de separabilidade entre as classes é agora representada por duas regiões compactas disjuntas, sendo uma convexa e a outra não-convexa. Estime então qual seria a quantidade mínima possível de neurônios tanto para a primeira como para a segunda camada neural escondida.
- 7) Considere a ação de se aplicar o PMC em um problema envolvendo aproximação de funções. As entradas da rede são a temperatura e a pressão, sendo que a saída fornece a quantidade de calor a ser inserida numa caldeira. O PMC (após o seu treinamento) deverá ser colocado em operação com o objetivo de auxiliar no controle da caldeira. Explique qual seria o primeiro procedimento a ser verificado frente aos novos valores de pressão e temperatura que estarão chegando às suas entradas.
- 8) Explique se é possível assumir valores nulos para os limiares ( $\theta$ ) de

todos os neurônios da última camada de um PMC, quando de sua aplicação em problemas envolvendo aproximação de funções.

9) Discorra se é possível aplicar o PMC, em problemas envolvendo aproximação de funções, quando se assume a função de ativação linear para todos os neurônios de sua única camada intermediária.

10) Discorra sobre a necessidade do pré-processamento das amostras de treinamento e teste, explicitando-se ainda a sua influência na velocidade de treinamento da rede PMC.

### 5.8 – Projeto prático 1 (aproximação de funções)

Para a confecção de um processador de imagens de ressonância magnética observou-se que a variável  $\{y\}$ , que mede a energia absorvida do sistema, poderia ser estimada a partir da medição de três outras grandezas  $\{x_1, x_2, x_3\}$ . Entretanto, em função da complexidade do processo, sabe-se que este mapeamento é de difícil obtenção por técnicas convencionais, sendo que o modelo matemático disponível para sua representação tem fornecido resultados insatisfatórios.

Assim, a equipe de engenheiros e cientistas pretende utilizar um *Perceptron Multicamadas* como um aproximador universal de funções, tendo-se como objetivo final a estimativa (após o treinamento) da energia absorvida  $\{y\}$  em função dos valores de  $x_1, x_2$  e  $x_3$ . A topologia da rede a ser implementada, constituída de duas camadas neurais, está ilustrada na figura 5.47.

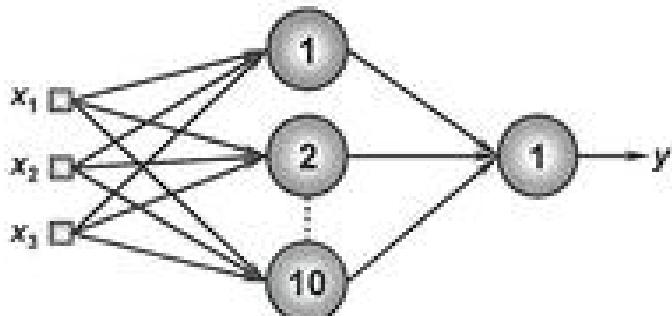


Figura 5.47 – Topologia de PMC (projeto prático 1)

Utilizando o algoritmo de aprendizagem *backpropagation* (regra Delta generalizada), com as amostras de treinamento apresentadas no apêndice III,

e assumindo-se também que todas as saídas já estejam normalizadas, realize as seguintes atividades:

1) Execute cinco treinamentos para a rede PMC, inicializando-se as matrizes de pesos com valores apropriados em cada treinamento. Se for o caso, reinicie o gerador de números aleatórios em cada treinamento a fim de modificar os seus valores iniciais. Utilize a função de ativação logística (sigmóide) para todos os neurônios, com taxa de aprendizado ( $\eta$ ) de 0,1 e precisão ( $\epsilon$ ) de  $10^{-4}$ . O conjunto de treinamento está disponível no apêndice III.

2) Registre os resultados finais dos cinco treinamentos na tabela 5.2.

Tabela 5.2 – Resultados dos treinamentos (projeto prático 1)

Treinamento	Erro quadrático médio	Número total de épocas
1º (T1)		
2º (T2)		
3º (T3)		
4º (T4)		
5º (T5)		

3) Para aqueles dois treinamentos da tabela 5.2, com maiores números de épocas, trace os respectivos gráficos dos valores de erro quadrático médio ( $E_w$ ) em relação a cada época de treinamento. Imprima os dois gráficos numa mesma folha de modo não superpostos.

4) Fundamentado na tabela do item 2, explique de forma detalhada por que tanto o erro quadrático médio como o número total de épocas variam de treinamento para treinamento.

5) Para todos os treinamentos efetuados no item 2, faça a validação da rede aplicando o conjunto de teste fornecido na tabela seguinte. Obtenha para cada treinamento o erro relativo médio (%) entre os valores desejados frente aqueles fornecidos pela rede, em relação a todas as amostras de teste. Forneca também a respectiva variância.

Tabela 5.3 – Conjunto de padrões de teste (projeto prático 1)

Amostra	$x_1$	$x_2$	$x_3$	$d$	$y$ (T1)	$y$ (T2)	$y$ (T3)	$y$ (T4)	$y$ (T5)
1	0,0611	0,2860	0,7464	0,4831					
2	0,5102	0,7464	0,0860	0,5965					
3	0,0004	0,6916	0,5006	0,5318					
4	0,9430	0,4476	0,2648	0,6843					
5	0,1399	0,1610	0,2477	0,2872					
6	0,6423	0,3229	0,8567	0,7683					
7	0,6492	0,0007	0,6422	0,5686					
8	0,1818	0,5078	0,9046	0,6601					
9	0,7382	0,2647	0,1916	0,5427					
10	0,3879	0,1307	0,8656	0,5836					
11	0,1903	0,6523	0,7820	0,6950					
12	0,8401	0,4490	0,2719	0,6790					
13	0,0029	0,3264	0,2476	0,2956					
14	0,7088	0,9342	0,2763	0,7742					
15	0,1283	0,1882	0,7253	0,4662					
16	0,8882	0,3077	0,8931	0,8093					
17	0,2225	0,9182	0,7820	0,7581					
18	0,1957	0,8423	0,3085	0,5826					
19	0,9991	0,5914	0,3933	0,7938					
20	0,2299	0,1524	0,7353	0,5012					
Erro relativo médio (%)									
Variância (%)									

6) Fundamentado nas análises da tabela anterior, indique qual das configurações finais de treinamento {T1, T2, T3, T4 ou T5} seria a mais adequada para o sistema de ressonância magnética, ou seja, qual está oferecendo a melhor generalização.

## 5.9 – Projeto prático 2 (classificação de padrões)

No processamento de bebidas, a aplicação de um determinado conservante é feita em função da combinação de quatro variáveis de tipo real, de-

finidas por  $x_1$  (teor de água),  $x_2$  (grau de acidez),  $x_3$  (temperatura) e  $x_4$  (tensão interfacial). Sabe-se que existem apenas três tipos de conservantes que podem ser aplicados, os quais são definidos por A, B e C. Em seguida, realizam-se ensaios em laboratório a fim de especificar qual tipo deve ser aplicado em uma bebida específica.

A partir de 148 ensaios executados em laboratório, a equipe de engenheiros e cientistas resolveu aplicar uma rede *Perceptron* multicamadas como classificadora de padrões, visando identificar qual tipo de conservante seria introduzido em determinado lote de bebidas. Por questões operacionais da própria linha de produção, utilizar-se-á aqui uma rede *Perceptron* com três saídas, conforme configuração apresentada na figura 5.48.

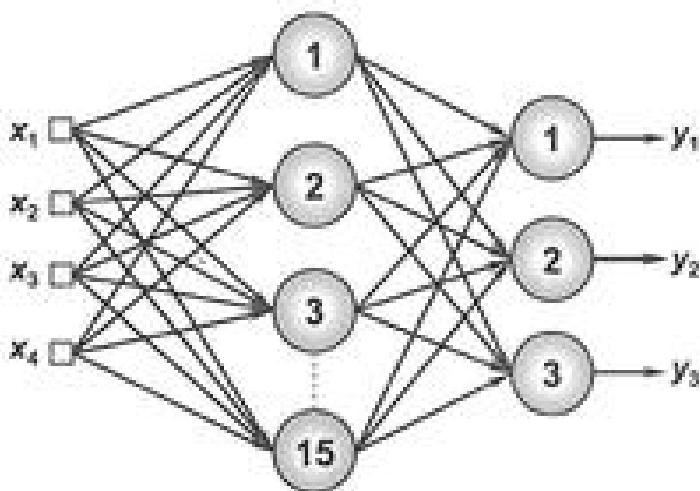


Figura 5.48 – Topologia de PMC (projeto prático 2)

A padronização para a saída, a qual representa o conservante a ser aplicado, ficou definida de acordo com a tabela 5.4.

Tabela 5.4 – Padronização das saídas da rede (projeto prático 2)

Tipo de conservante	$y_1$	$y_2$	$y_3$
Tipo A	1	0	0
Tipo B	0	1	0
Tipo C	0	0	1

Utilizando os dados de treinamento apresentados no apêndice III, execute então o treinamento de uma rede PMC (quatro entradas e três saídas) que possa classificar, em função apenas dos valores medidos de  $x_1$ ,  $x_2$ ,  $x_3$  e  $x_4$ , qual o tipo de conservante que pode ser aplicado em determinada bebida. Para tanto, faça as seguintes atividades:

1) Execute o treinamento da rede *Pemprata*, por meio do algoritmo de aprendizagem *backpropagation* convencional, inicializando-se as matrizes de pesos com valores aleatórios apropriados. Utilize a função de ativação logística (sigmóide) para todos os neurônios, com taxa de aprendizado  $\{\eta\}$  de 0,1 e precisão  $\{\epsilon\}$  de  $10^{-4}$ .

2) Efetue, em seguida, o treinamento da rede *Pemprata* por meio do algoritmo de aprendizagem *backpropagation* com *momentum*, utilizando as mesmas matrizes de pesos iniciais que foram usadas no item anterior. Adote também a função de ativação logística para todos os neurônios, com taxa de aprendizado  $\{\eta\}$  de 0,1, fator de *momentum*  $\{a\}$  de 0,9 e precisão  $\{\epsilon\}$  de  $10^{-4}$ .

3) Para os dois treinamentos realizados nos itens anteriores, trace os respectivos gráficos dos valores de erro quadrático médio  $\{E_m\}$  em função de cada época de treinamento. Imprima os dois gráficos numa mesma página de modo não superpostos. Meça também o tempo de processamento envolvido com cada treinamento.

4) Dado que o problema se configura como um típico processo de classificação de padrões, implemente então a rotina que faz o pós-processamento das saídas fornecidas pela rede (valores reais) para números inteiros. Como sugestão, adote o critério de arredondamento simétrico, isto é:

$$y_i^{\text{pos}} = \begin{cases} 1, & \text{se } y_i \geq 0,5 \\ 0, & \text{se } y_i < 0,5 \end{cases} \quad (5.68)$$

5) Faça a validação da rede aplicando o conjunto de teste fornecido na tabela 5.5. Forneça a taxa de acertos (%) entre os valores desejados frente àsquelas respostas fornecidas pela rede (após o pós-processamento) em relação a todos os padrões de teste.

Tabela 5.5 – Conjunto de padrões de teste (projeto prático 2)

Amostra	$x_1$	$x_2$	$x_3$	$x_4$	$d_1$	$d_2$	$d_3$	$y_t^{real}$	$y_t^{pred}$	$y_t^{erro}$
1	0,6622	0,7101	0,6236	0,7894	0	0	1			
2	0,2741	0,1552	0,1333	0,1516	1	0	0			
3	0,6772	0,8518	0,6543	0,7573	0	0	1			
4	-0,2178	0,5039	0,6415	0,5039	0	1	0			
5	0,7260	0,7600	0,7007	0,4953	0	0	1			
6	0,2473	0,2941	0,4248	0,3087	1	0	0			
7	0,5682	0,5683	0,5054	0,4426	0	1	0			
8	0,6566	0,6715	0,4952	0,3951	0	1	0			
9	0,0766	0,4717	0,2921	0,2954	1	0	0			
10	0,1187	0,2568	0,3149	0,3037	1	0	0			
11	0,5673	0,7011	0,4083	0,5552	0	1	0			
12	0,3164	0,2251	0,3528	0,2560	1	0	0			
13	0,7684	0,9588	0,6825	0,6398	0	0	1			
14	0,9633	0,7850	0,6777	0,6059	0	0	1			
15	0,7739	0,8505	0,7934	0,6626	0	0	1			
16	0,4219	0,4136	0,1408	0,0940	1	0	0			
17	0,6616	0,4365	0,6997	0,8129	0	0	1			
18	0,7325	0,4761	0,3688	0,5683	0	1	0			
Total de acertos (%)										

## 5.10 – Projeto prático 3 (sistemas variantes no tempo)

O preço de determinada mercadoria, disposta para ser comercializada no mercado financeiro de ações, possui um histórico de variação de valor conforme mostrado na tabela do apêndice III.

Um pool de pesquisadores estará tentando aplicar redes neurais artificiais a fim de prever o comportamento futuro deste processo. Assim, pretende-se utilizar uma arquitetura de PMC, com topologia *time delay neural network (TDNN)*, conforme mostrado na figura 5.49.

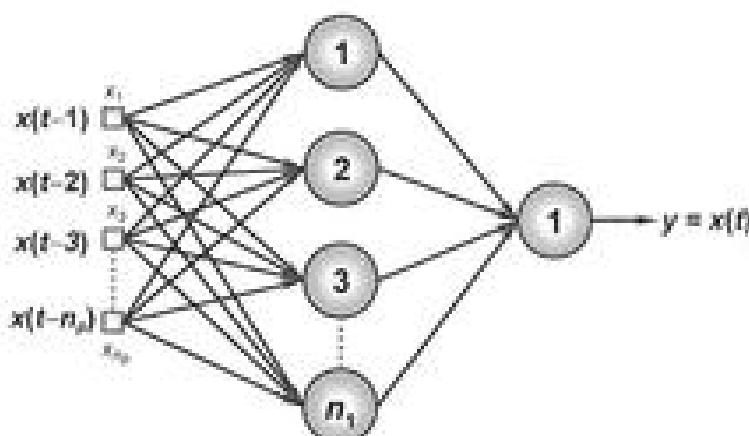


Figura 5.49 – Topologia de PMC (projeto prático 3)

As topologias candidatas passíveis de serem aplicadas no mapamento deste problema são especificadas por:

TDNN 1 → 5 entradas ( $n_s = 5$ ), com  $n_i = 10$

TDNN 2 → 10 entradas ( $n_s = 10$ ), com  $n_i = 15$

TDNN 3 → 15 entradas ( $n_s = 15$ ), com  $n_i = 25$

Utilizando o algoritmo de aprendizagem *backpropagation com momentum*, com os dados de treinamento apresentados no apêndice III, realize as seguintes atividades:

1) Execute três treinamentos para cada rede TDNN candidata, inicializando-se suas matrizes de pesos (em cada treinamento) com valores aleatórios apropriados. Se for o caso, reinicie o gerador de números aleatórios em cada treinamento, de modo que os elementos das matrizes de pesos iniciais não sejam os mesmos. Utilize a função de ativação logística (sigmoid) para todos os neurônios, com taxa de aprendizado ( $\eta$ ) de 0,1, fator de *momentum* ( $\alpha$ ) de 0,8 e precisão ( $\varepsilon$ ) de  $0,5 \times 10^{-6}$ .

2) Considerando-se as respostas dessas três topologias candidatas, registre os resultados finais de seus treinamentos na tabela 5.6.

Tabela 5.6 – Resultados dos treinamentos (projeto prático 3)

Treinamento	TDNN 1		TDNN 2		TDNN 3	
	$E_s$	Épocas	$E_s$	Épocas	$E_s$	Épocas
1º (T1)						
2º (T1)						
3º (T1)						

3) Para todos os treinamentos efetuados no item 2, faça então a validação das três topologias candidatas de TDNN em relação aos valores desejados apresentados na tabela 5.7. Forneca, para cada treinamento, o erro relativo médio entre os valores desejados e as respostas fornecidas pela rede, em relação a todos os padrões de teste. Obtenha também a respectiva variância.

**Tabela 5.7 – Conjunto de padrões de teste (projeto prático 3)**

- 4) Para cada uma das topologias candidatas da tabela 5.7, considerando ainda o melhor treinamento {T1, T2 ou T3} realizado em cada uma delas, trace o gráfico dos valores de erro quadrático médio ( $E_m$ ) em função de cada época de treinamento. Imprima os três gráficos numa mesma página de modo não superpostos.
- 5) Ainda, para cada uma das topologias apresentadas na tabela 5.7, considerando também o melhores treinamento {T1, T2 ou T3} realizado em cada uma, trace o gráfico dos valores desejados frente aos valores estimados pela respectiva rede, em função do domínio de estimação considerado ( $t = 101, \dots, 120$ ). Imprima os três gráficos numa mesma página de modo não superpostos.
- 6) Baseado nas análises dos itens anteriores, indique qual das topologias candidatas {TDNN 1, TDNN 2 ou TDNN 3}, e com qual configuração final de treinamento {T1, T2 ou T3}, seria a mais adequada para realização de previsões neste processo.



Walter Puma

## Redes de funções de base radial (RBF)

### 6.1 – Introdução

As redes denominadas funções de base radial, convencionalmente conhecidas como RBF (*radial basis function*), podem ser também empregadas em quase todos os tipos de problemas tratados pelo PMC, inclusive aqueles que envolvem aproximação de funções e classificação de padrões.

Diferentemente das redes PMC, as quais podem ser compostas de diversas camadas intermediárias, a estrutura típica de uma rede RBF é composta por apenas uma, na qual as funções de ativação são do tipo gaussiana, conforme ilustra a figura 6.1.

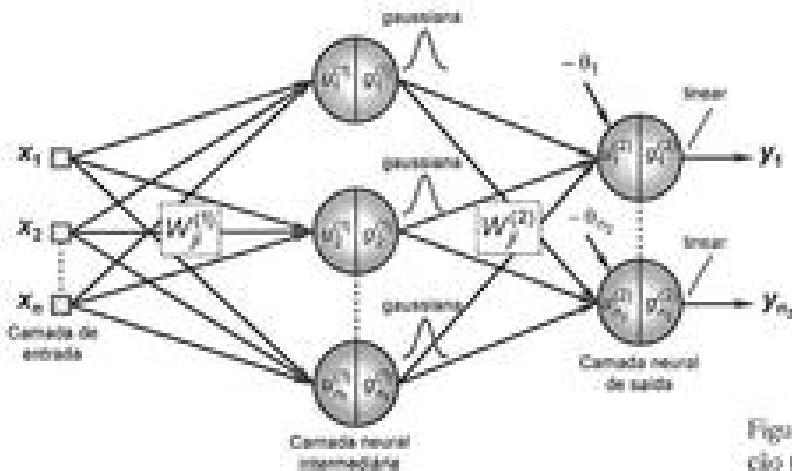


Figura 6.1 – Configuração típica de rede RBF

Uma das principais particularidades da rede RBF está na estratégia de treinamento utilizada para os ajustes dos pesos de suas duas únicas camadas neurais, que será detalhadamente apresentada na próxima seção. Em conformidade ainda ao observado na figura 6.1, outro diferencial desta arquitetura de rede é o tipo de função de ativação assumida nos neurônios da camada intermediária, sendo caracterizada por funções de base radial como as gaussianas.

A rede RBF, de acordo com a classificação exposta no capítulo 2, pertence também à arquitetura *feedforward* de camadas múltiplas, cujo treinamento é efetuado de forma supervisionada. A partir ainda da figura 6.1, verifica-se que o fluxo de informações na sua estrutura se inicia na camada de entrada, percorre então a respectiva camada intermediária (neurônios com funções de ativação gaussiana), finalizando-se seguidamente na camada neural de saída (neurônios com funções de ativação linear).

## 6.2 – Processo de treinamento de redes RBF

O princípio de funcionamento das redes RBF é também similar àquele do PMC, em que cada uma de suas entradas  $\{x\}$ , representando os sinais vindos da aplicação, será então propagada pela referida camada intermediária, em direção à camada de saída.

Entretanto, diferentemente do PMC, a estratégia de treinamento da RBF é constituída de duas fases ou estágios bem distintos entre si. O primeiro estágio, associado com os ajustes dos pesos dos neurônios da camada intermediária, adota um método de aprendizagem auto-organizado (não-supervisionado), que é dependente apenas das características dos dados de entrada. Na realidade, este ajuste está diretamente relacionado com a alocação das funções de base radial. Já o segundo estágio, vinculado aos ajustes dos pesos dos neurônios da camada de saída, utiliza um critério de aprendizagem similar àquele usado na última camada do PMC, ou seja, a regra delta generalizada.

Além disso, ao contrário também das redes PMC, o processo de treinamento se inicia pelos neurônios da camada intermediária, encerrando-se nos neurônios da camada de saída.

### 6.2.1 – Ajuste dos neurônios da camada intermediária (estágio I)

Conforme mencionado anteriormente, os neurônios pertencentes à cam-

da intermediária da RBF são constituídos de funções de ativação do tipo base radiais, sendo que a função gaussiana é uma das mais empregadas. A expressão que define uma função de ativação do tipo gaussiana é representada por:

$$g(u) = e^{-\frac{(u-c)^2}{2\sigma^2}} \quad (6.1)$$

onde  $c$  define o centro da função gaussiana e  $\sigma^2$  denota a sua variância (em que  $\sigma$  equivale ao desvio padrão), a qual indica o quanto disperso está o potencial de ativação ( $u$ ) em relação ao seu centro ( $c$ ). A figura 6.2 ilustra o formato geométrico de uma função gaussiana típica.

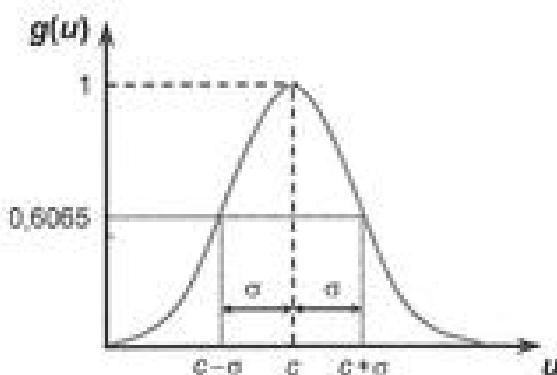


Figura 6.2 – Função de base radial do tipo gaussiana

Como comentário, quanto maior for então o valor da variância, mais alargada será a sua base. A figura 6.3 ilustra este aspecto para uma função gaussiana com três valores distintos para a variância, em que  $\sigma_1^2 < \sigma_2^2 < \sigma_3^2$ .

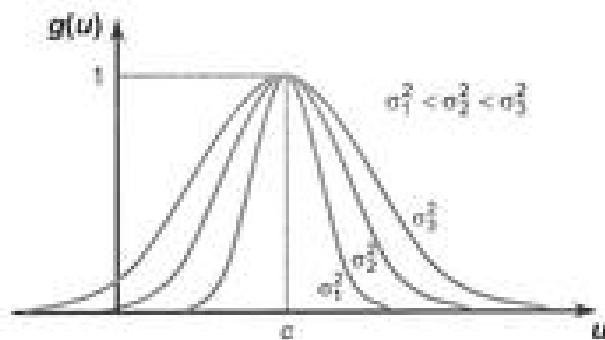


Figura 6.3 – Influência do parâmetro  $\sigma^2$  na função gaussiana

Desta forma, considerando a expressão fornecida em (6.1), os parâmetros livres a serem ajustados seriam então o centro  $c$  e a variância  $\sigma^2$ . De fato, levando-se em conta a terminologia adotada na figura 6.1, com a configuração dos neurônios escondidos da rede RBF, o centro  $c$  está diretamente associado aos seus próprios pesos, sendo que, nesta situação, a entrada  $u_j^{(1)}$  de cada um será o próprio vetor de entrada  $x$ , que representa os  $n$  sinais externos que são aplicados na rede. Consequentemente, a saída de cada neurônio  $j$  da camada intermediária é expressa por:

$$g_j^{(1)}(u_j^{(1)}) = g_j^{(1)}(x) = \frac{e^{-\frac{\sum_{i=1}^n (x_i - w_j^{(1)})^2}{2\sigma^2}}}{\sqrt{2\pi}}, \text{ onde } j = 1, \dots, n_1 \quad (6.2)$$

Para propósitos de ilustração, a figura 6.4 mostra uma função de ativação gaussiana em relação a dois sinais de entrada  $x_1$  e  $x_2$ , os quais compõem uma amostra  $x$ .

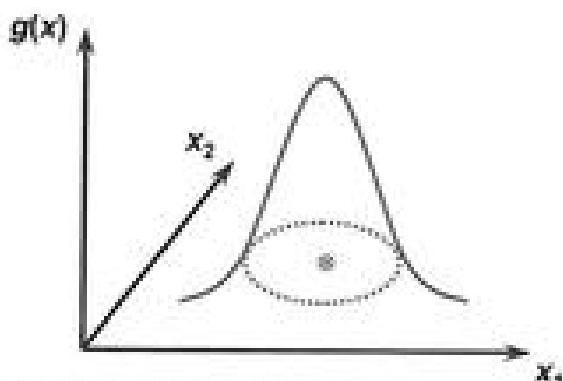


Figura 6.4 – Função de base radial do tipo gaussiana

Observa-se nesta figura que quanto mais próxima esteja uma determinada amostra (padrão) do centro da gaussiana, mais significante será o valor produzido pelo campo receptivo radial da função de ativação [Powell, 1987], a qual tenderá para o valor 1, conforme é possível abstrair da aplicação de (6.2). Em tal condição, o neurônio produzirá respostas similares para todos aqueles padrões que estejam a uma mesma distância radial do centro da gaussiana.

Em se tratando de problemas de classificação de padrões, conforme examinado na subseção 5.4.1, o PMC computa as fronteiras de delimitação de classes por intermédio de uma combinação de hiperplanos. Já na RBF,

com funções de ativação do tipo gaussiana, as fronteiras delimitadoras são definidas por campos receptivos hiperesféricos. Consequentemente, a classificação dos padrões levará em conta a distância radial em que os mesmos se encontram em relação ao centro das hiperesferas. A figura 6.5 ilustra estes princípios para um problema de classificação de padrões constituído de duas entradas  $x_1$  e  $x_2$ .

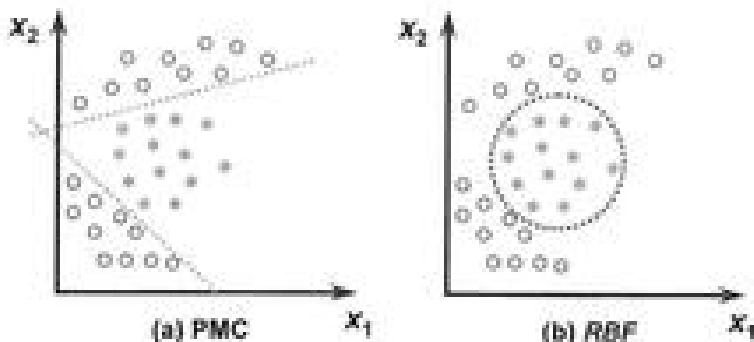


Figura 6.5 – Fronteira de separabilidade do PMC e RBF

Vale ressaltar que o campo receptivo hiperesférico atua em todo domínio real, pois este é também o domínio de uma função gaussiana; entretanto, os valores mais relevantes serão aqueles delimitados pela variação, sendo que esta servirá então para cobrir as amostras de uma das classes, ao passo que valores bem pequenos de ativação seriam produzidos para as amostras da outra classe.

De acordo também com a figura 6.5, considerando um problema bidimensional, a fronteira de separabilidade do PMC será constituída de segmentos de reta, ao passo que a rede RBF será delimitada por um campo receptivo que pode ser representado por uma circunferência, conforme ilustrada na figura 6.5(b). Ainda neste exemplo hipotético, a fronteira do PMC foi formada por dois segmentos de reta, indicando-se então a presença de dois neurônios na camada intermediária. No caso da RBF, um único campo receptivo radial foi capaz de produzir potencial para agrupar todos os padrões de uma mesma classe, assinalando-se a presença de apenas um neurônio na camada escondida.

Assim, o principal objetivo dos neurônios da camada intermediária é posicionar os centros de suas gaussianas de forma mais apropriada possível. Um dos métodos muito bem utilizado para esta finalidade é denominado de *k-médias* (*k*-médias), cujo propósito é posicionar os centros de *k*-gaussianas em regiões onde os padrões de entrada tenderão a se agrupar [Duda *et alii*, 2001].

Vale aqui ressaltar que o valor do parâmetro  $k$  é igual ao número de neurônios da camada intermediária, pois a função de ativação de cada um é uma gaussiana conforme definida por (6.2), sendo que seus centros serão representados pelos seus respectivos vetores de peso.

A sequência de procedimentos computacionais, visando o primeiro estágio de treinamento de redes RBF, é explicitada em termos de pseudocódigo conforme se segue.

#### Início (Algoritmo RBF – Primeiro Estágio de Treinamento)

- {<1> Obter o conjunto de amostras de treinamento { $x^{(k)}$ };
- <2> Iniciar o vetor de pesos de cada neurônio da camada intermediária com os valores das  $n_1$  primeiras amostras de treinamento;
- <3> Repetir as instruções:
  - {<3.1> Para todas as amostras de treinamento { $x^{(k)}$ }, fazer:
    - <3.1.1> Calcular as distâncias euclidianas entre  $x^{(k)}$  e  $w_j^{(l)}$ , considerando-se cada  $j$ -ésimo neurônio por vez;
    - <3.1.2> Selecionar o neurônio  $j$ , que contenha a menor distância, com o intuito de agrupar a referida amostra junto ao centro mais próximo;
    - <3.1.3> Atribuir a amostra  $x^{(k)}$  ao grupo  $\Omega^{(l)}$ ;
  - <3.2> Para todos  $w_j^{(l)}$ , onde  $j = 1, \dots, n_1$ , fazer:
    - <3.2.1> Ajustar  $w_j^{(l)}$  de acordo com as amostras em  $\Omega^{(l)}$ :
$$w_j^{(l)} = \frac{1}{m^{(l)}} \sum_{x^{(k)} \in \Omega^{(l)}} x^{(k)} \quad (m^{(l)} \text{ é o no. de amostras em } \Omega^{(l)})$$

Até que: não haja mudanças nos grupos  $\Omega^{(l)}$  entre as iterações.

- <4> Para todos  $w_j^{(l)}$ , onde  $j = 1, \dots, n_1$ , fazer:
  - <4.1> Calcular a variância de cada uma das funções de ativação gaussianas pelo critério da distância quadrática média:

$$\sigma_j^2 = \frac{1}{m^{(l)}} \sum_{x^{(k)} \in \Omega^{(l)}} \sum_{i=1}^n (x_i^{(k)} - w_j^{(l)})^2$$

#### Fim (Algoritmo RBF – Primeiro Estágio de Treinamento)

Com o propósito de elucidar o primeiro estágio de treinamento da RBF e do método de agrupamento (*clustering k-means*), considera-se um problema aplicativo constituído de duas entradas  $x_1$  e  $x_2$ , cujas amostras de treinamento estão representadas na figura 6.6.

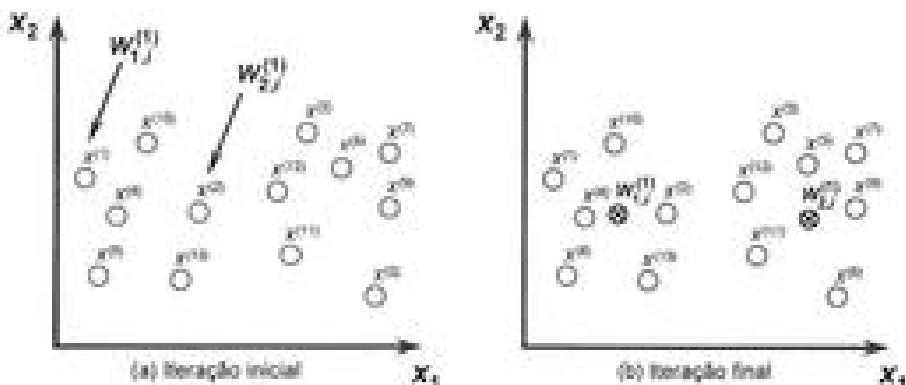


Figura 6.6 – Processo de ajuste utilizando o método *k-means*.

A figura 6.6(a) mostra a disposição espacial das amostras a ser usadas na aplicação do método *k-means*, assumindo-se que, para este exemplo ilustrativo, a RBF será composta de apenas dois neurônios em sua camada intermediária ( $n_i = 2$ ). Assim, considerando o passo <2> do algoritmo referente ao primeiro estágio de treinamento da RBF, observa-se que os vetores de pesos do primeiro e do segundo neurônio receberão inicialmente os valores atribuídos a  $x^{(0)}$  e  $x^{(2)}$ , conforme ilustrado também na figura 6.6(a).

Em seguida, o passo <3.1> consiste de calcular as distâncias de cada uma das amostras em relação aos vetores  $\mathbf{W}_1^{(0)}$  (neurônio 1) e  $\mathbf{W}_2^{(0)}$  (neurônio 2). Como exemplo, considerando a amostra  $x^{(0)}$ , observa-se pela figura 6.6(a) que estará mais próxima à  $\mathbf{W}_1^{(0)}$  do que  $\mathbf{W}_2^{(0)}$ ; ao passo que para a amostra  $x^{(2)}$  se constata que a menor distância está em relação à  $\mathbf{W}_2^{(0)}$ . Nesses casos, a amostra  $x^{(0)}$  será inserida no conjunto  $\Omega^{(0)}$  que contém aquelas cujo neurônio 1 foi vencedor (menor distância), enquanto que a amostra  $x^{(2)}$  será incorporada ao conjunto  $\Omega^{(2)}$  que contém aquelas que o neurônio 2 foi vencedor. Repetindo este procedimento para todas as outras amostras, obter-se-ão os seguintes conjuntos  $\Omega^{(0)}$  e  $\Omega^{(2)}$ :

$$\Omega_{\text{vencedor}}^{(0)} = \{x^{(1)}, x^{(4)}, x^{(8)}, x^{(10)}\}$$

$$\Omega_{\text{vencedor}}^{(2)} = \{x^{(2)}, x^{(3)}, x^{(5)}, x^{(6)}, x^{(7)}, x^{(9)}, x^{(11)}, x^{(12)}, x^{(13)}\}$$

Em seguida, em conformidade com a aplicação do passo <3.2>, atualizam-se os valores de  $\mathbf{W}_1^{(1)}$  e  $\mathbf{W}_2^{(1)}$  levando-se em conta todas as amostras contidas nos respectivos conjuntos  $\Omega^{(1)}$  e  $\Omega^{(2)}$ . Os passos <3.1> e <3.2> são então repetidos de maneira iterativa até que não haja mais mudanças de alocação de amostras nos conjuntos  $\Omega^{(1)}$  e  $\Omega^{(2)}$  entre duas iterações sucessivas. Assim, após o alcance desta convergência, os vetores  $\mathbf{W}_1^{(1)}$  e  $\mathbf{W}_2^{(1)}$  estarão automaticamente posicionados nos centros dos agrupamentos, conforme ilustrado na figura 6.6(b), sendo que os conjuntos  $\Omega^{(1)}$  e  $\Omega^{(2)}$  teriam configurações finais definidas por:

$$\Omega_{final}^{(1)} = \{x^{(1)}, x^{(2)}, x^{(4)}, x^{(8)}, x^{(10)}, x^{(13)}\}$$

$$\Omega_{final}^{(2)} = \{x^{(3)}, x^{(5)}, x^{(6)}, x^{(7)}, x^{(9)}, x^{(11)}, x^{(12)}\}$$

Finalmente, os campos receptivos a ser produzidos pelos vetores de pesos dos neurônios 1 e 2, associados a cada um dos dois agrupamentos, poderão ser então obtidos por intermédio da aplicação do passo <4.1>. A figura 6.6 ilustra os raios de abrangência mais significativos desses campos receptivos formados ao redor dos vetores de pesos ajustados pelo algoritmo *k-means*.

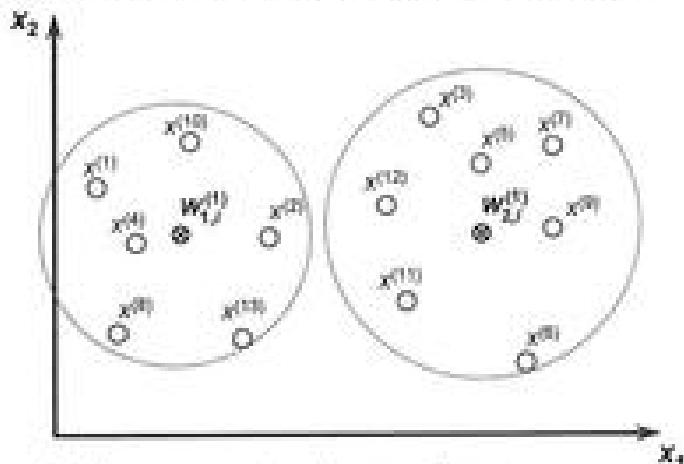


Figura 6.7 – Ilustração de campos receptivos dos vetores de pesos

Vale enfatizar novamente que quanto mais próxima esteja a amostra do centro do agrupamento, mais significante será o valor da resposta do campo receptivo gaussiano.

Diversos outros métodos de clusterização, tais como aqueles baseados em critérios de máxima verossimilhança [Nowlan, 1989] ou no método dos mínimos quadrados recursivo [Chen, 1995], podem ser também utilizados no ajuste dos vetores de pesos dos neurônios da camada intermediária.

### 6.2.2 – Ajuste dos neurônios da camada de saída (estágio II)

A aplicação dos passos de ajuste referentes aos pesos dos neurônios da camada de saída deve ser executada, em sequência, somente após a finalização do primeiro estágio de treinamento.

Desta forma, o segundo estágio de treinamento é efetuado utilizando os mesmos procedimentos usados para a camada de saída do PMC, os quais foram formulados em detalhes na subseção 5.3.1. Assim, diferentemente do primeiro estágio de treinamento da RBF, este segundo estágio utiliza um processo de aprendizado supervisionado.

A partir da terminologia utilizada na figura 6.1, o conjunto de treinamento para os neurônios da camada de saída será então constituído por pares de entrada e saída desejada, em que as entradas serão as respostas produzidas pelas funções de ativação gaussianas dos neurônios da camada intermediária frente às respectivas amostras de treinamento, ou seja:

$$u_j^{(2)} = \sum_{i=1}^m \underbrace{W_j^{(2)} \cdot g_i^{(1)}(u_i^{(1)})}_{\text{peso}_j \quad \text{peso}_i} - \theta_j, \quad \text{onde } j = 1, \dots, n_2 \quad (6.3)$$

onde  $W_j^{(2)}$  e  $\theta_j$  são respectivamente os pesos e limiares referentes aos neurônios da camada de saída, ao passo que os valores correspondentes à parcela (ii) serão aqueles calculados a partir de (6.2), utilizando-se para tanto os valores de  $W_j^{(1)}$  e  $\sigma_j^2$  obtidos no primeiro estágio de treinamento.

Finalmente, assim como ocorre no PMC, adotando-se como ativação para a camada de saída da RBF a função linear definida em (1.14), tem-se que os neurônios de saída realizarão não somente uma combinação linear das funções de ativação gaussianas produzidas pelos neurônios da camada anterior. Portanto, a partir da expressão (6.3), mediante o auxílio da figura 6.1, as respostas produzidas pelos neurônios de saída serão dadas por:

$$y_j = g_j^{(2)}(u_j^{(2)}) = u_j^{(2)}, \quad \text{onde } j = 1, \dots, n_2 \quad (6.4)$$

As instruções em pseudocódigo, visando-se o segundo estágio de treinamento, são explicitadas conforme se segue.

**Inicio {Algoritmo RBF – Segundo Estágio de Treinamento}**

- <1> Obter o conjunto original de amostras de treinamento ( $x^{(k)}$ );
- <2> Obter o vetor de saída desejada ( $d^{(k)}$ ) para cada amostra;
- <3> Iniciar  $W_j^{(2)}$  com valores aleatórios pequenos;
- <4> Especificar taxa de aprendizagem ( $\eta$ ) e precisão requerida ( $\epsilon$ );
- <5> Para todas as amostras ( $x^{(k)}$ ), fazer:
  - <5.1> Obter os valores de  $g_j^{(1)}$  em relação à  $x^{(k)}$ ; (conforme (8.2))
  - <5.2> Assumir  $z^{(k)} = [g_1^{(1)} \ g_2^{(1)} \dots g_{n_p}^{(1)}]^T$ ; (pseudo-amostras)
- <6> Iniciar o contador de número de épocas (época  $\leftarrow 0$ );
- <7> Repetir as instruções:
  - <7.1>  $E_M^{\text{anterior}} \leftarrow E_M$ ; (conforme (5.8))
  - <7.2> Para todos os pares de treinamento ( $z^{(k)}, d^{(k)}$ ), fazer:
    - Ajustar  $W_j^{(2)}$  e  $\theta_j$ , aplicando os mesmos passos usados
    - na camada de saída do PMC (Subseção 5.3.1)
  - <7.3>  $E_M^{\text{atual}} \leftarrow E_M$ ; (conforme (5.8))
  - <7.4>  $\text{época} \leftarrow \text{época} + 1$
- Até que:  $|E_M^{\text{atual}} - E_M^{\text{anterior}}| \leq \epsilon$

**Fim {Algoritmo RBF – Segundo Estágio de Treinamento}**

A variável *época* pode também ser utilizada como critério de parada para o segundo estágio de treinamento, englobando-se aquelas situações em que a precisão especificada para o problema se torna inalcançável.

Assim, a aplicação do primeiro estágio de treinamento (não-supervisionado), seguido pelo segundo (supervisionado), permite o ajustamento de todos os parâmetros livres ( $W_j^{(1)}, \sigma_j^2, W_j^{(2)}, \theta_j$ ) da rede RBF. Em complemento, a determinação da quantidade de neurônios da camada intermediária (funções de ativação gaussianas) pode ainda ser realizada usando as técnicas de validação cruzada apresentadas na subseção 5.5.1.

Portanto, após a finalização dos estágios de treinamento da RBF, a rede já estará apta para estimar as saídas do processo que foi mapeado, frente agora às novas amostras que serão apresentadas em suas entradas. A sequência de passos para esta fase de operação é apresentada conforme se segue.

**Inicio {Algoritmo RBF – Fase de Operação}**

- { <1> Apresentar uma amostra ( $x$ );
- <2> Assumir os parâmetros  $W_j^{(1)}$ ,  $\sigma_j^2$ ,  $W_j^{(2)}$  e  $\theta_j$ , já ajustados durante os estágios de treinamento;
- <3> Executar as seguintes instruções:
  - <3.1> Obter  $g_j^{(1)}$ ; (conforme (6.2))
  - <3.2> Obter  $v_j^{(2)}$ ; (conforme (6.3))
  - <3.3> Obter  $y_j$ ; (conforme (6.4))
- <4> Disponibilizar as saídas da rede mediante os valores contidos em  $y_j$ .

**Fim {Algoritmo RBF – Fase de Operação}**

Durante a aplicação da fase de operação, o passo <3.3> poderá ser também suprimido caso se adote a função de ativação linear (1.14) para os neurônios da camada de saída, pois em tal situação o valor de  $y_j$  é igual ao resultado do passo <3.2>.

### 6.3 - Aplicabilidades das redes RBF

As redes RBF têm sido largamente utilizadas em problemas que envolvem aproximação de funções (identificação de sistemas) e classificação de padrões, embora haja ainda diversos tipos de outras aplicações em que tenham sido implementadas com sucesso.

Em relação aos problemas de aproximação de funções, deduções inspiradas nos estudos de Kolmogorov (1957) são também válidas para as redes RBF. Para tanto, Park & Sandberg (1991) demonstraram que tais redes são consideradas aproximadores universais de funções, assim como acontece também com o PMC. De fato, substituindo a expressão (6.3) em (6.4), obtém-se o seguinte relacionamento:

$$y_j = \sum_{i=1}^{n_1} \underbrace{W_j^{(2)} \cdot g_j^{(1)}(v_j^{(1)})}_{\text{parcial}} - \theta_j, \quad \text{onde } j = 1, \dots, n_2 \quad (6.5)$$

Por intermédio da figura 6.1, pode-se concluir que a função  $y$ , a ser mapeada pela RBF é constituída por uma superposição de funções de ativação gaussianas (parcela (ii)), representadas pelos termos  $g_j^{(n)}(x^{(n)})$ , as quais são linearmente ponderadas pelos pesos  $W_j^{(n)}$  (parcela (i)) associados aos neurônios de sua camada de saída.

Para propósitos de ilustração, conforme também destacado para o PMC na figura 5.26, considera-se que uma RBF será então utilizada para implementar um problema de aproximação funcional, cujas algumas amostras de treinamento são representadas na figura 6.8.

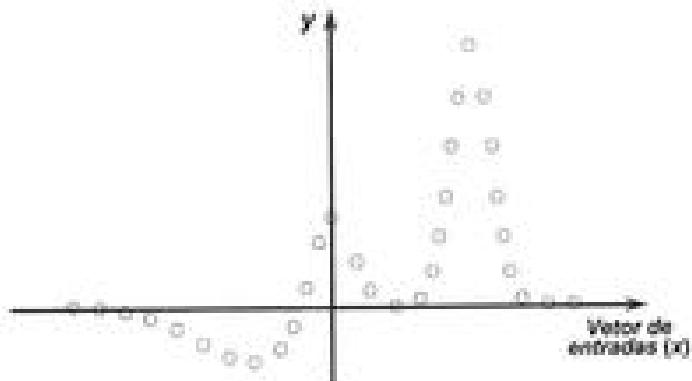


Figura 6.8 – Amostras de treinamento da RBF referente a um problema de aproximação funcional

A figura 6.9 ilustra a RBF a ser utilizada no mapeamento do problema, cuja camada escondida é constituída de três neurônios.

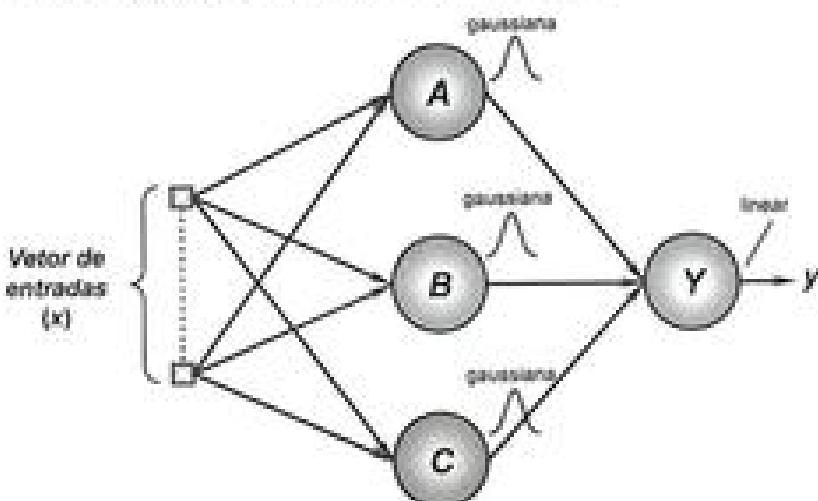


Figura 6.9 – Topologia de RBF aplicada em aproximação de funções

Após a efetivação dos estágios de treinamento da RBF, tem-se na figura 6.10 uma eventual configuração advinda da combinação linear envolvendo as funções de ativação gaussianas, as quais compõem as saídas dos neurônios A, B e C, pertencentes à camada intermediária. Em suma, o neurônio de saída Y da RBF produz uma função que consegue mapear o comportamento do processo apresentado na figura 6.8.

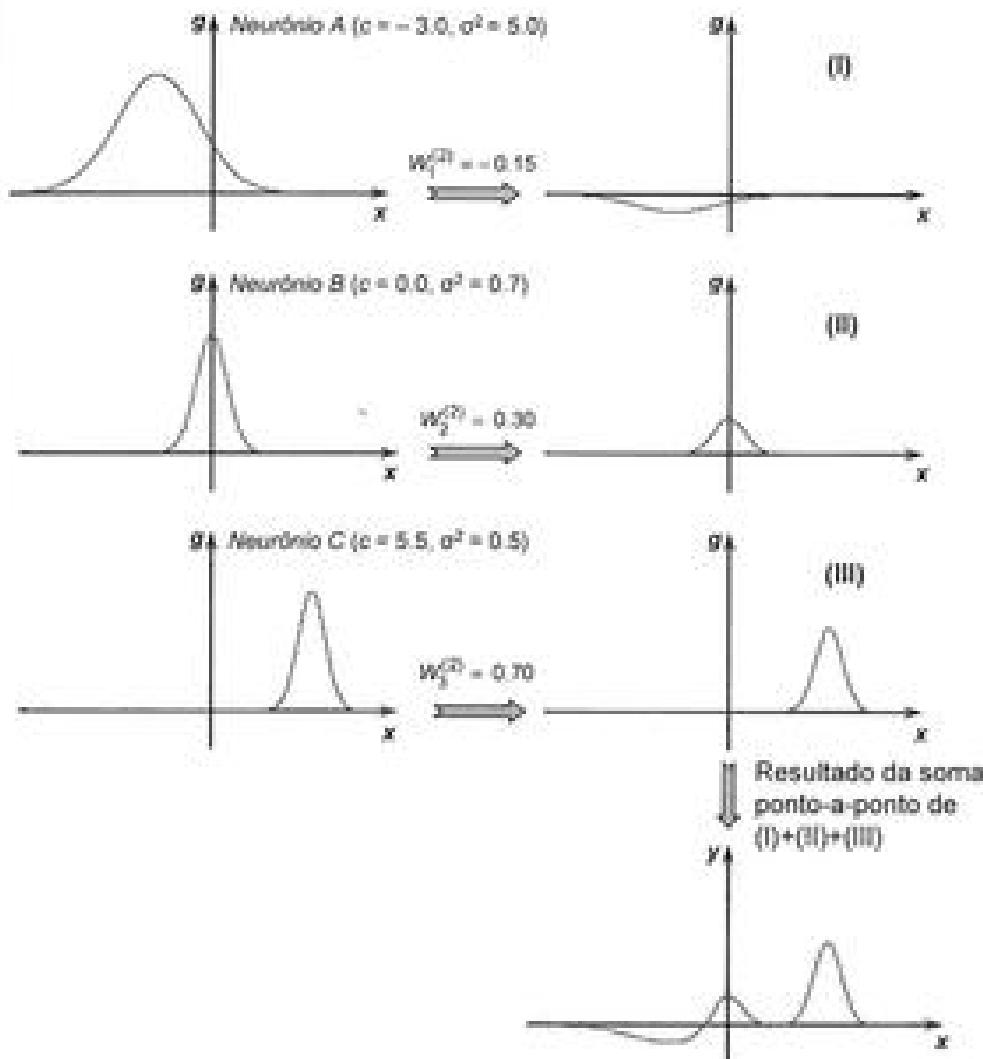


Figura 6.10 – Ilustração de superposição de funções gaussianas para um problema de aproximação funcional

Mediante exame da figura 6.10, observa-se que os vetores de pesos dos neurônios da camada intermediária  $\{W_i^{(1)}\}$  são então responsáveis pela translação das funções de ativação gaussianas em seus domínios de definição. Com efeito, esses pesos estão representando os centros das respectivas funções gaussianas, enquanto que os pesos do neurônio da camada de saída  $\{W_j^{(2)} = W_j^{(3)}\}$  são responsáveis pelo escalamento das mesmas.

Assim como ocorrido com a rede PMC, uma RBF pode ser utilizada para mapear qualquer função não-linear contínua, definida num domínio compacto (fechado). Entretanto, a quantidade exata de neurônios em sua camada escondida, que seja suficiente para realizar tal tarefa, é ainda motivo de pesquisa, pois também depende da complexidade do problema a ser mapeado. Por conseguinte, as técnicas de *cross-validation*, apresentadas na subseção 5.5.1, podem ser também utilizadas na RBF como estratégia para selecionar a topologia mais adequada.

Já em relação aos problemas de classificação de padrões, as fronteiras decisórias proporcionadas pelas funções de ativação gaussianas serão formadas por campos receptivos hiperesféricos, cujos raios de abrangência produzem valores mais significativos para aqueles pontos (amostras) que estejam mais próximos de seus centros. Como exemplo, para um problema de classificação de padrões hipotético, considera-se uma RBF constituída de duas entradas  $\{x_1 \text{ e } x_2\}$ , dois neurônios em sua camada intermediária e apenas um neurônio na camada de saída, conforme ilustração da figura 6.11.

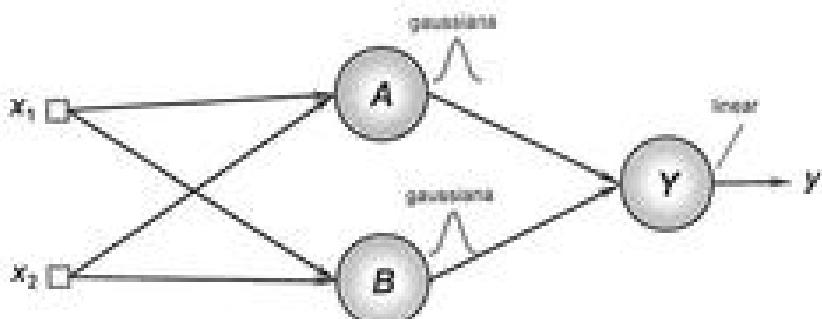


Figura 6.11 – Exemplo de RBF aplicada em classificação de padrões

Levando-se em conta o primeiro estágio de treinamento da RBF, a figura 6.12 ilustra uma possível configuração final para as funções gaussianas associadas aos neurônios A e B, a partir dos resultados produzidos pela aplicação do algoritmo *k-means*.

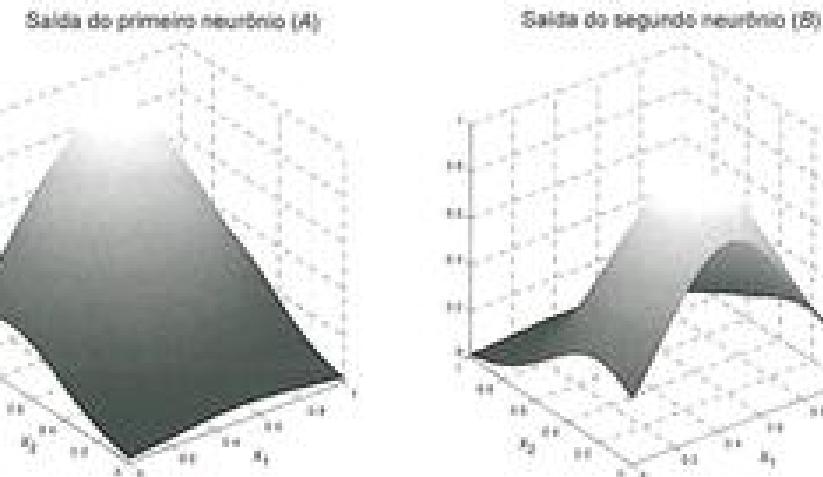


Figura 6.12 – Funções gaussianas associadas aos neurônios A e B.

Verifica-se ainda, por meio da análise da figura 6.12, que a função gaussiana produzida pelo neurônio A é mais larga (maior variância) que aquela computada pelo neurônio B.

Finalmente, para este exemplo ilustrativo, a figura 6.13 mostra a saída do neurônio Y, cujo objetivo é realizar uma combinação linear das duas funções gaussianas implementadas pelos neurônios A e B, frente aos propósitos da correta classificação dos padrões.

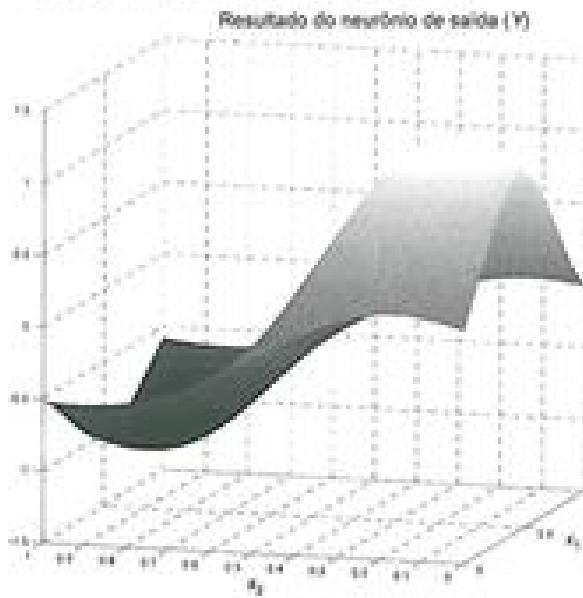


Figura 6.13 – Ilustração da saída do neurônio Y que realiza a combinação linear das duas funções gaussianas produzidas pelos neurônios A e B.

Por intermédio ainda da análise da figura 6.13, constata-se que o neurônio  $Y$  acabou invertendo a função gaussiana produzida pela saída do neurônio  $A$ , tendo-se puramente como objetivo a correta classificação das classes do problema. Para tanto, o valor do peso conectando o neurônio  $A$  ao neurônio  $Y$  será então negativo, ao passo que o valor do peso conectando o neurônio  $B$  ao  $Y$  será positivo.

Finalmente, a figura 6.14 ilustra o campo receptivo, com formato hiperférico (padrão circular em 2D), proporcionado pelo neurônio de saída da RBF.

Diferentemente daquelas fronteiras que seriam produzidas pelo PMC, as quais são delimitadas por hiperplanos (retas em 2D), tem-se aqui uma fronteira radial resultante da combinação linear das saídas produzidas pelos neurônios  $A$  e  $B$ .

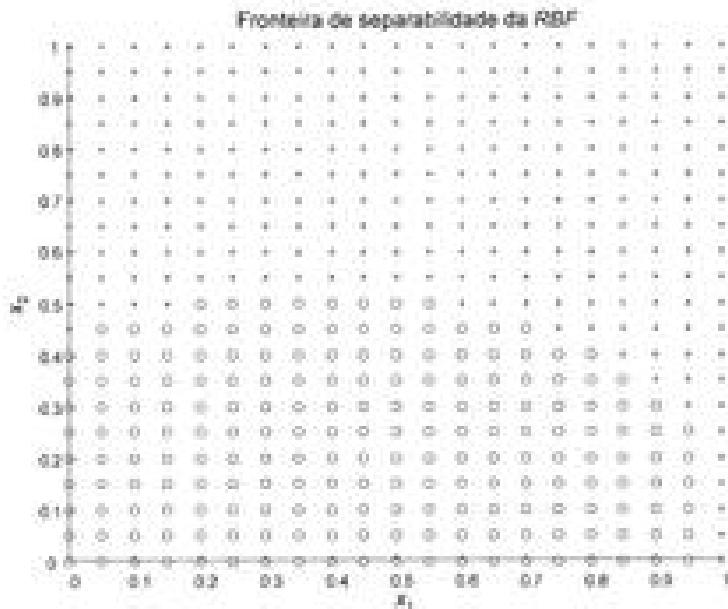


Figura 6.14 – Fronteira de classificação radial para a rede RBF

Caso o problema de classificação de padrões tivesse três entradas, o campo receptivo seria representado por uma esfera, de modo que se teriam hiperesferas para quatro ou mais sinais de entrada.

Para propósitos de entendimento sobre as diferenças entre o PMC e a RBF, quando aplicadas a um mesmo problema de classificação de padrões, consideram-se novamente dois tipos de amostras que devem ser classificadas

em função de seus sinais de entrada, caracterizados por  $x_1$  e  $x_2$ , conforme ilustração da figura 6.15.

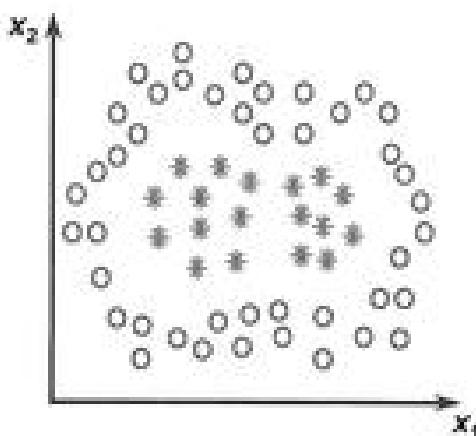


Figura 6.15 – Disposição de amostras visando classificação de padrões utilizando PMC e RBF

A figura 6.16 mostra uma concepção geométrica das eventuais fronteiras de decisão após o treinamento de ambas as abordagens. A partir da análise da figura, depreende-se que a camada intermediária do PMC poderia ser constituída por oito neurônios em virtude de se visualizar esta quantidade de segmentos de retas; ao passo que a camada intermediária da RBF seria composta por somente dois neurônios, os quais são caracterizados pelas circunferências que delimitam as fronteiras.

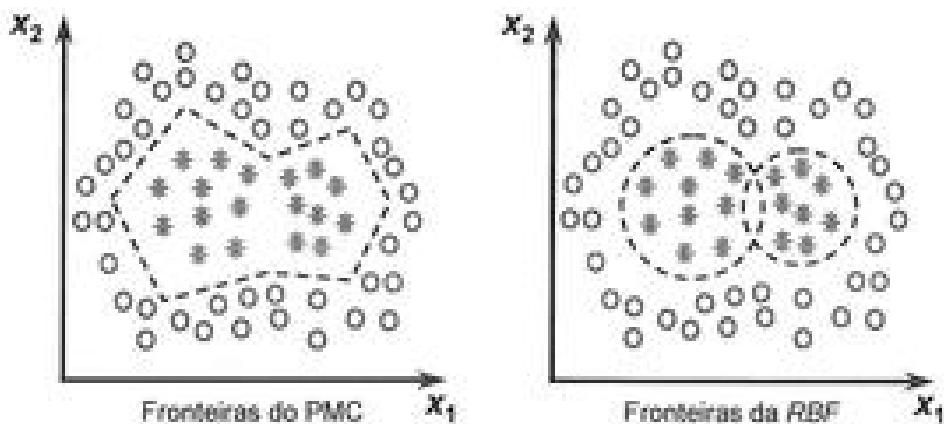


Figura 6.16 – Delimitação de fronteiras de classificação do PMC e RBF

Assim, destaca-se aqui novamente que os limitantes das fronteiras de classificação são sempre definidos por hiperplanos quando da utilização de rede PMC, sendo os mesmos caracterizados por campos receptivos hiperesféricos quando da aplicação de rede RBF. A figura 6.17 ilustra de forma tridimensional as regiões de separabilidade baseadas nos resultados da figura 6.16.

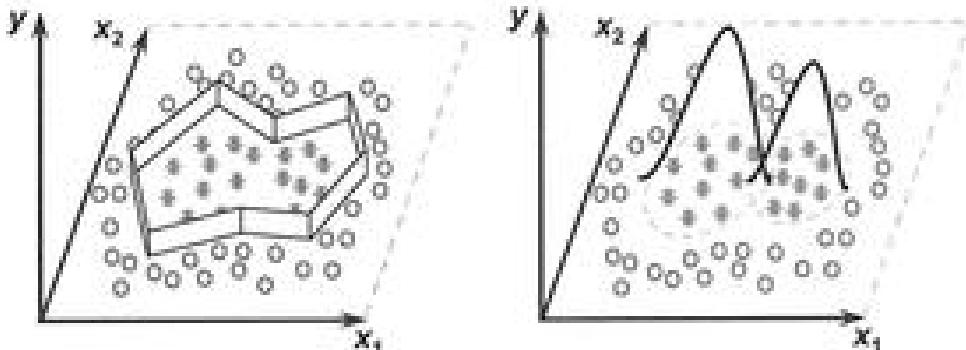


Figura 6.17 – Regiões de separabilidade produzidas pelo PMC e RBF

Em geral, quando se tem disponível uma grande quantidade de amostras de treinamento, independentemente do problema a ser mapeado, as redes RBF podem requerer mais neurônios em sua camada intermediária quando comparada com as redes PMC. Entretanto, o treinamento de redes RBF é normalmente muito mais rápido em relação às redes PMC [Chen *et al.*, 1991].

## 6.4 – Exercícios

- 1) Considerando os aspectos de arquitetura neural, discorra sobre três diferenças entre redes PMC e RBF.
- 2) Explique quais seriam as eventuais implicações em se assumir a mesma variância para todas as funções de ativação dos neurônios que compõem a camada intermediária da rede RBF.
- 3) Discorra sobre a necessidade de se utilizar os limiares de ativação ( $\theta$ ) em todos os neurônios que compõem a camada de saída da rede RBF.
- 4) Descreva qual a finalidade dos campos receptivos hiperesféricos produzidos pelas funções de ativação gaussianas que compõem os neurônios da camada intermediária da rede RBF.

5) Escreva em pseudocódigo os passos necessários para o ajuste de  $W_j^{(2)}$  e  $\theta_j$ , durante a aplicação do segundo estágio de treinamento da rede RBF.

6) Considerando o problema do ou-exclusivo (porta  $X_{OR}$ ), discorra sobre a quantidade mínima de neurônios que seriam necessários na camada intermediária da RBF visando a solução do problema.

7) Mostre que, em termos vetoriais, a expressão dada em (6.2) é equivalente a seguinte equação:

$$g_j^{(2)}(x) = e^{\frac{(x - W_{(j)}^{(1)})^T (x - W_{(j)}^{(1)})}{2\sigma_j^2}}$$

onde  $W_{(j)}^{(1)}$  é o vetor de pesos associados ao  $j$ -ésimo neurônio da camada intermediária.

8) Explique quais as modificações a serem impostas na expressão vetorial do exercício anterior para que esta função de ativação produza campos receptivos hiperclipsoidais.

9) Considerando novamente o problema do ou-exclusivo, explique se o número mínimo de neurônios, a ser utilizado na camada intermediária da RBF, seria também o mesmo quando do uso de funções de ativação que produzam campos receptivos hiperclipsoidais ao invés de radiais.

10) Para problemas de classificação de padrões, quais seriam as eventuais vantagens e limitações da rede RBF frente ao PMC.

## 6.5 – Projeto prático 1 (classificação de padrões)

A verificação da presença de radiação em determinadas substâncias nucleares pode ser efetuada por meio da análise da concentração de duas variáveis,  $\{x_1$  e  $x_2\}$ . A partir de 50 situações conhecidas, resolveu-se então treinar uma RBF para a execução da tarefa de classificação de padrões neste processo, cuja topologia está ilustrada na figura 6.18.

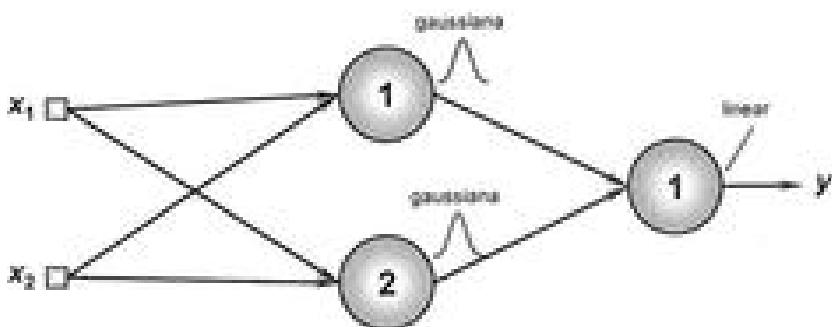


Figura 6.18 – Arquitetura da RBF (projeto prático 1)

A padronização para a saída, representando a presença ou ausência de sinais de radiação, ficou definida conforme a terminologia introduzida na tabela 6.1.

Tabela 6.1 – Padronização de saída da RBF

Status de radiação	Saída ( $y$ )
Presença	1
Ausência	-1

Utilizando os dados de treinamento apresentados no apêndice IV, execute o treinamento de uma RBF (2 entradas e 1 saída) que possa classificar, em função apenas dos valores medidos de  $x_1$  e  $x_2$ , se em determinada substância há a presença ou ausência de radiação. Em sequência, realize as atividades seguintes:

- 1) Execute o treinamento da camada escondida por meio do algoritmo  $k\text{-means}$ . Em se tratando de um problema de classificação de padrões, compute os centros dos dois agrupamentos (*clusters*), levando-se em consideração apenas aqueles padrões com presença de radiação. Após o treinamento (primeiro estágio), forneça os valores das coordenadas do centro de cada agrupamento e sua respectiva variância.

Tabela 6.2 – Resultados do primeiro estágio de treinamento

Agrupamento	Centro	Variância
1		
2		

2) Execute o segundo estágio de treinamento utilizando a regra Delta generalizada, com taxa de aprendizado de 0,01 e precisão de  $10^{-7}$ . Após a convergência ser alcançada, forneça os valores dos pesos e limitar referentes ao neurônio da camada de saída.

Tabela 6.3 – Resultados do segundo estágio de treinamento

Parâmetro	Valor
$W_{11}^{(2)}$	
$W_{21}^{(2)}$	
$\theta_1$	

3) Dado que o problema se configura como um típico processo de classificação de padrões, implemente o procedimento que realize o pós-processamento das respostas computadas pela rede (valores reais), a fim de produzir os números inteiros padronizados na tabela 6.1. Para tanto, utilize a função sinal definida por:

$$y^{\text{pos}} = \begin{cases} 1, \text{ se } y \geq 0 \\ -1, \text{ se } y < 0 \end{cases}$$

onde  $y^{\text{pos}}$  fornece o resultado final para o problema de classificação de padrões, sendo somente utilizada para o pós-processamento do conjunto de teste.

4) Faça a validação da rede aplicando o conjunto de teste fornecido na tabela 6.4. Forneça a taxa de acertos (%) entre os valores desejados e as respostas produzidas pela RBF (após o pós-processamento) em relação a todas as amostras de teste.

Tabela 6.4 – Resultados de validação (projeto prático 1)

Amostra	$x_1$	$x_2$	$d$	$y$	$y^{\text{obs}}$
1	0,8705	0,9329	-1		
2	0,0388	0,2703	1		
3	0,8236	0,4458	-1		
4	0,7075	0,1502	1		
5	0,9587	0,8863	-1		
6	0,6115	0,9365	-1		
7	0,3534	0,3848	1		
8	0,3268	0,2766	1		
9	0,6129	0,4518	-1		
10	0,9948	0,4962	-1		
Taxa de acertos (%):					

5) Se for o caso, explique quais estratégias podem ser implementadas a fim de se tentar aumentar a taxa de acertos desta RBF.

## 6.6 – Projeto prático 2 (aproximação de funções)

A quantidade de gasolina  $\{y\}$  a ser introduzida por um sistema de injeção eletrônica de combustível, para veículos automotores, pode ser computada em tempo-real em função de três grandezas  $\{x_1, x_2 \text{ e } x_3\}$ . Devido à complexidade inherent do processo, configurado como um sistema não-linear, pretende-se utilizar uma rede neural artificial a fim de mapear o relacionamento entre as suas entradas com a respectiva saída.

Sabe-se que, para efetuar o referido mapeamento, que se caracteriza como um problema de aproximação funcional, duas potenciais arquiteturas podem ser aplicadas, a saber, a rede PMC ou RBF. Visto que a equipe de engenheiros e cientistas já realizou o mapeamento do problema por intermédio da arquitetura PMC, o objetivo agora é treinar uma RBF a fim de que os resultados fornecidos por ambas possam ser confrontados.

Para tal propósito, efetue o treinamento de uma RBF com o objetivo de computar a quantidade de gasolina  $\{y\}$ , a ser inserida pelo sistema de injeção eletrônica, em função das variáveis  $x_1, x_2$  e  $x_3$ . A topologia da rede RBF está ilustrada na figura 6.19.

As suas configurações candidatas, passíveis de serem aplicadas no mapeamento deste problema, são especificadas como se segue:

RBF-1 → Topologia com  $n_r = 05$

RBF-2 → Topologia com  $n_r = 10$

RBF-3 → Topologia com  $n_r = 15$

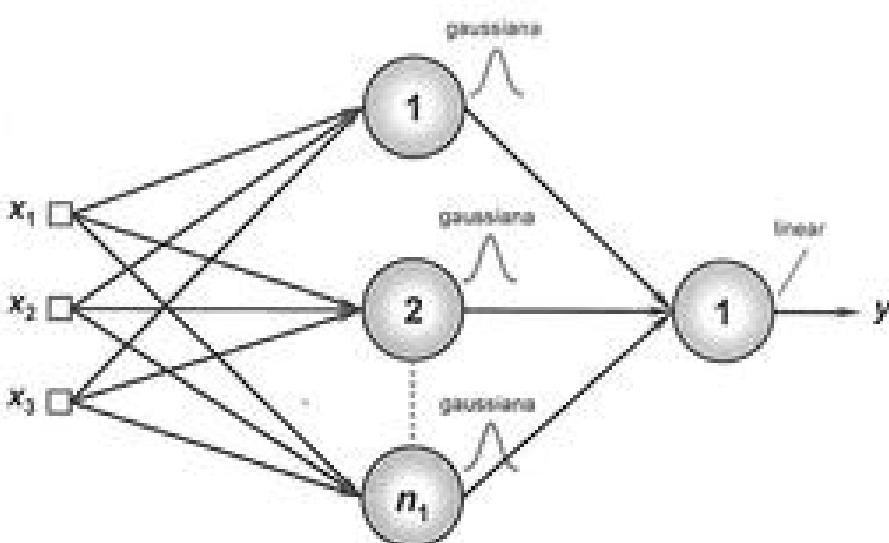


Figura 6.19 – Arquitetura da RBF (projeto prático 2)

Utilizando os dados de treinamento apresentados no apêndice IV, implemente o treinamento destas topologias candidatas, realizando em sequência os seguintes exercícios:

1) Execute três treinamentos para cada topologia de RBF, inicializando-se a matriz de pesos da camada de saída com valores aleatórios entre 0 e 1. Se for o caso, reinicie o gerador de números aleatórios em cada treinamento, de modo que os elementos das matrizes de pesos iniciais não sejam os mesmos. Utilize uma taxa de aprendizado  $\{\eta\}$  de 0,01 e precisão  $\{\epsilon\}$  de  $10^{-3}$ .

2) Registre os resultados finais desses três treinamentos  $\{T1, T2 \text{ e } T3\}$  na tabela 6.5, considerando-se cada uma das três topologias candidatas de RBF.

Tabela 6.5 – Resultados do erro quadrático médio (projeto prático 2)

Treinamento	RBF-1		RBF-2		RBF-3	
	$E_{\mu}$	Épocas	$E_{\mu}$	Épocas	$E_{\mu}$	Épocas
1º (T1)						
2º (T2)						
3º (T3)						

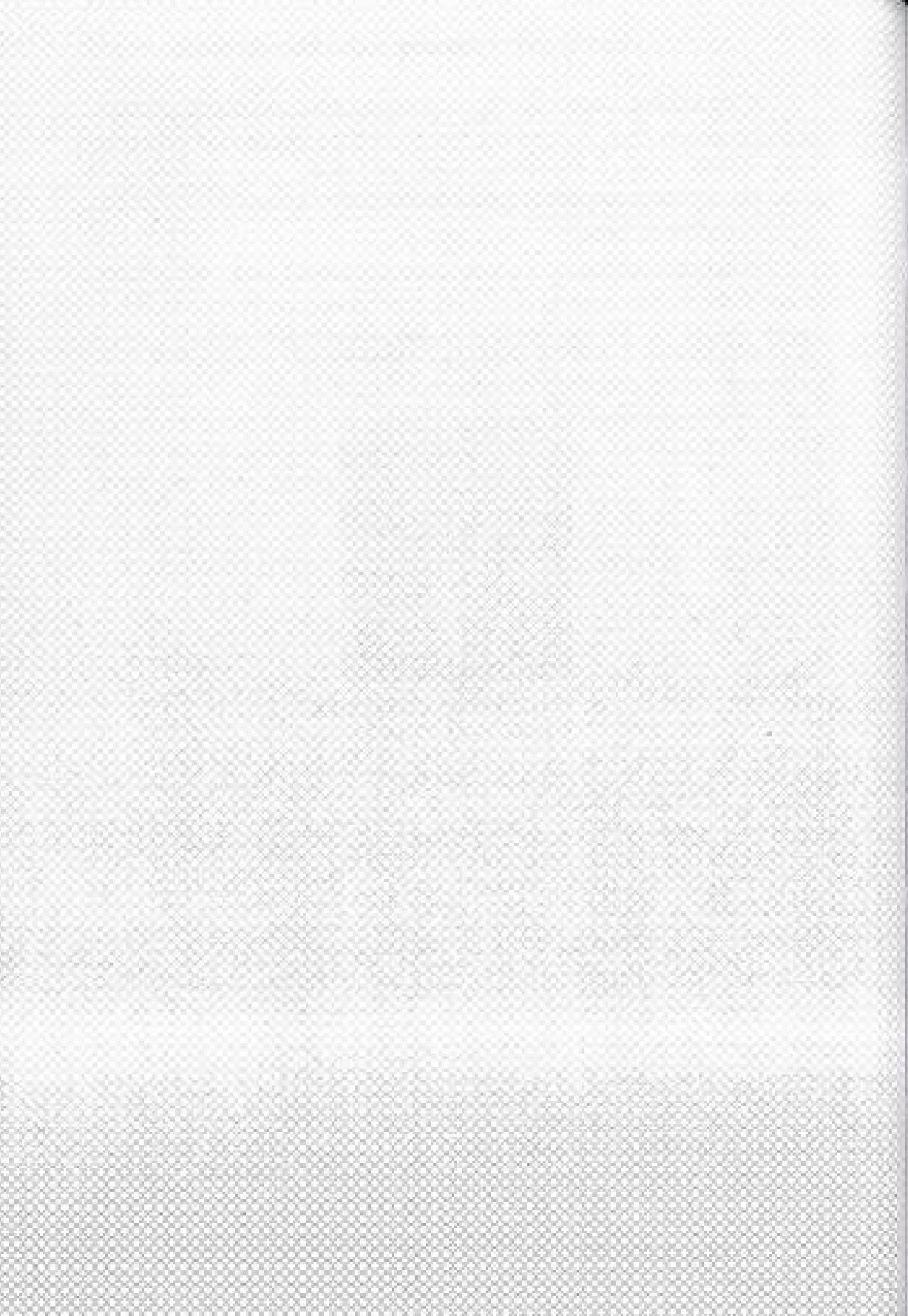
3) Para todos os treinamentos efetuados no item 2, faça a validação da rede frente aos valores desejados apresentados na tabela 6.6. Forneça para cada treinamento o erro relativo médio (%) entre os valores desejados e aqueles fornecidos pela rede em relação a todas as amostras de teste. Obtenha ainda a respectiva variância (%).

Tabela 6.6 – Resultados de validação (projeto prático 2)

- 4) Para cada uma das topologias apresentadas na tabela 6.6, considerando ainda o melhor treinamento {T1, T2 ou T3} realizado em cada uma, trace o gráfico dos valores de erro quadrático médio ( $E_m$ ) em função de cada época de treinamento. Imprima os três gráficos numa mesma página de modo não superpostos.
- 5) Baseado nas análises dos itens anteriores, assinale qual topologia candidata {RBF-1, RBF-2 ou RBF-3}, e com qual configuração final de treinamento {T1, T2 ou T3}, seria a mais adequada para este problema.



Andrey Nikolaevich Kolmogorov  
(Андрей Николаевич Колмогоров)



## Redes recorrentes de Hopfield

### 7.1 – Introdução

Conforme mencionado na subseção 2.2.3, as redes neurais artificiais consideradas recorrentes são aquelas em que as saídas de uma camada neural podem ser realimentadas às suas entradas.

O melhor exemplo de rede recorrente pode ser atribuído, por certo, àquelas que foram idealizadas por Hopfield (1982), as quais são mais comumente conhecidas como redes de Hopfield. Tal arquitetura de rede neural artificial, com realimentação global, possui as seguintes características:

- Comportamento tipicamente dinâmico;
- Capacidade de memorizar relacionamentos;
- Possibilidade de armazenamento de informações;
- Facilidade de implementação em hardware analógico.

Os trabalhos desenvolvidos por Hopfield também contribuíram para desencadear na época um interesse renovado, e ainda bem crescente, por redes neurais artificiais, os quais colaboraram para o renascimento de impor-

tantes pesquisas na área e que estavam, de certo modo, estagnadas desde a publicação do livro *Precipices* por Minsky & Papert (1969).

Com efeito, as propostas de Hopfield discorriam sobre os elos existentes entre as arquiteturas neurais recorrentes frente aos sistemas dinâmicos e à física estatística, impulsionando assim a curiosidade de diversas outras áreas do conhecimento. Seu grande trunfo foi formular diversos aspectos que mostraram que redes neurais recorrentes de uma única camada podiam ser caracterizadas por uma função de energia relacionada aos estados de seu comportamento dinâmico. Tais arquiteturas foram também batizadas como modelos "vídeo de spin" (*ising model*), fazendo-se analogia ao ferromagnetismo [Amit *et al.*, 1985].

Face ao contexto exposto, a minimização da função de energia ( $E(x)$ ) levaria a saída da rede para pontos de equilíbrio estáveis, sendo que estes seriam a solução desejada frente a um problema específico. A figura 7.1 mostra uma ilustração sobre pontos de equilíbrio estáveis e instáveis.

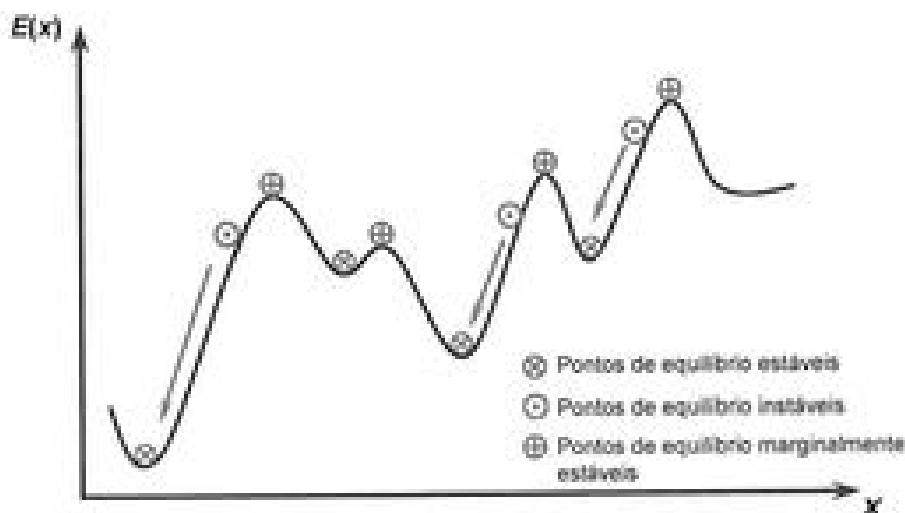


Figura 7.1 – Ilustração mostrando pontos de equilíbrio estáveis

Observa-se aqui que a função de energia pode então conter diversos pontos de equilíbrio considerados estáveis. Durante o processo de convergência da rede, a partir de alguns estados iniciais, a tendência é que estes sempre se movam em direção a um desses pontos de equilíbrio considerados estáveis (pontos fixos).

Além das notáveis memórias associativas (seção 7.4), as principais aplicações relacionadas às redes de Hopfield estão concentradas na área de otimização de sistemas, tais como programação dinâmica [Silva *et alii*, 2001; Wang, 2004], programação linear [Malek & Yari, 2005; Tank & Hopfield, 1986], otimização não-linear restrita [Silva *et alii*, 2007; Xia & Wang, 2004], e otimização combinatorial [Atencia *et alii*, 2005; Hopfield & Tank, 1985].

## 7.2 – Princípio de funcionamento da rede de Hopfield

A rede de Hopfield originalmente proposta, de acordo com a figura 7.2, é constituída de uma única camada, em que todos os neurônios são completamente interligados, isto é, todos os neurônios da rede são conectados a todos os outros e a si próprios (todas as saídas da rede realimentam todas as suas entradas). O operador  $z^{-1}$  indica atraso temporal de uma unidade.

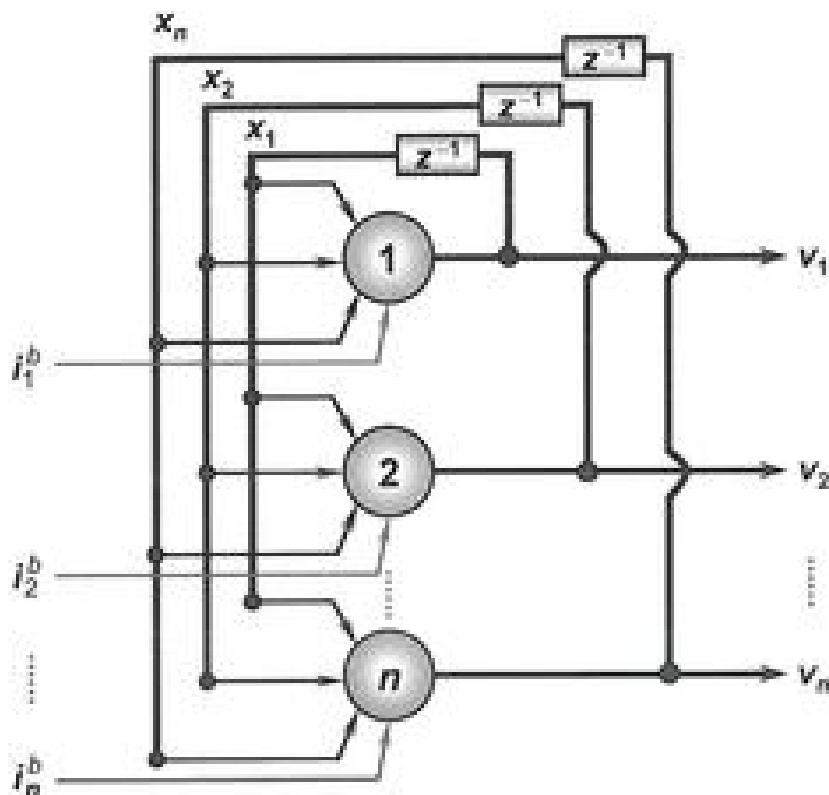


Figura 7.2 – Rede de Hopfield convencional

Em virtude de a rede de Hopfield ser composta por apenas uma camada de neurônios, utilizar-se-á então a mesma terminologia que geralmente é adotada na literatura correlata, tendo-se o intuito de facilitar o entendimento de sua dinâmica. De forma simplificada, a expressão que rege o comportamento, em tempo contínuo, de cada neurônio da rede de Hopfield é dada por:

$$\dot{u}_j(t) = -\eta \cdot u_j(t) + \sum_{i=1}^n W_{ji} \cdot v_i(t) + i_j^0, \text{ onde } j = 1, \dots, n \quad (7.1)$$

$$\left\{ \begin{array}{l} v_j(t) = g(u_j(t)) \\ \end{array} \right. \quad (7.2)$$

onde:

$\dot{u}_j(t)$  é o estado interno do  $j$ -ésimo neurônio, com  $\dot{u}_j(t) = du/dt$ ;

$v_j(t)$  é a saída do  $j$ -ésimo neurônio;

$W_{ji}$  é o valor do peso sináptico conectando o  $j$ -ésimo neurônio ao  $i$ -ésimo neurônio;

$i_j^0$  é o limiar (bias) aplicado ao  $j$ -ésimo neurônio;

$g(\cdot)$  é uma função de ativação, monótona crescente, que limita a saída de cada neurônio em um intervalo predefinido;

$\eta \cdot u_j(t)$  é um termo de decaimento passivo; e

$x_j(t)$  é a entrada do  $j$ -ésimo neurônio (figura 7.2).

Por conseguinte, baseado na interpretação das expressões (7.1) e (7.2), verifica-se que o comportamento da rede de Hopfield é sempre dinâmico, podendo ser então sintetizado pelos seguintes passos:

- I. Aplicar um conjunto de sinais  $\{x\}$  nas entradas.
- II. Obter o vetor de saídas  $\{v\}$  da rede.
- III. Realimentar as entradas com as saídas anteriores  $\{x \leftarrow v\}$ .
- IV. Caso o vetor de saídas  $\{v\}$  permaneça estável (constante) entre iterações sucessivas, então a rede já convergiu para a resposta final (ponto de equilíbrio estável); caso contrário, repita os passos de (I) a (III) até obter convergência.

Eloicando ainda mais os passos envolvidos com a dinâmica da rede de Hopfield, tem-se também a seguinte interpretação para as expressões (7.1) e (7.2):

- Em  $t = t_0$        $\rightarrow$       entrada  $x(t_0)$  gera saída  $v(t_0)$ .
- Em  $t = t_1$        $\rightarrow$       entrada  $x(t_1) = v(t_0)$  gera saída  $v(t_1)$ .
- Em  $t = t_2$        $\rightarrow$       entrada  $x(t_2) = v(t_1)$  gera saída  $v(t_2)$ .
- (...)
- Em  $t = t_m$        $\rightarrow$        $v(t_m) = v(t_{m-1})$ , condição de rede estabilizada.

Logo, essas sequências de iterações sucessivas produzem mudanças (cada vez menores) nas saídas da rede, até que os seus valores se tornem constantes (estáveis). Dependendo da configuração dos parâmetros livres da rede, este processo pode se repetir indefinidamente, gerando-se então evoluções instáveis. A maioria das redes com dinâmica instável apresentam comportamento caótico [Alligood *et al.*, 2000], e tem sido motivo de inspiração de diversos trabalhos relacionando redes de Hopfield e sistemas caóticos, tais como aqueles investigados em Huang & Yang (2006) e Zhang & Xu (2007).

Na maioria das aplicações práticas, em que se implementaram as redes de Hopfield por meio de algoritmos computacionais iterativos, utilizou-se a sua versão em tempo discreto, cujas expressões (7.1) e (7.2) são então convertidas para:

$$\left\{ \begin{array}{l} u_j(k) = \sum_{i=1}^n W_{ji} \cdot v_i(k-1) + I_j^0, \text{ onde } j = 1, \dots, n \\ v_j(k) = g(u_j(k)) \end{array} \right. \quad (7.3)$$

$$\left\{ \begin{array}{l} u_j(k) = \sum_{i=1}^n W_{ji} \cdot v_i(k-1) + I_j^0, \text{ onde } j = 1, \dots, n \\ v_j(k) = g(u_j(k)) \end{array} \right. \quad (7.4)$$

onde  $k$  é um passo de iteração.

Assim como no caso contínuo, dado qualquer conjunto de condições iniciais  $x^{(0)}$ , e impondo-se restrições apropriadas sobre a matriz de pesos  $W$ , a fim de se garantir estabilidade (convergência para pontos de equilíbrio estáveis), a sua versão em tempo discreto também sempre convergirá para aqueles pontos que correspondem à solução do problema.

### 7.3 – Condições de estabilidade da rede de Hopfield

Devido ao comportamento dinâmico das redes de Hopfield, um amplo domínio de sistemas não-lineares complexos pode ser representado pelas mesmas. Dependendo de como os parâmetros da rede são escolhidos, esta deve funcionar como um sistema estável, ou então como um oscilador, ou ainda como um sistema totalmente caótico [Aihara *et al.*, 1990].

A maioria das aplicações que envolvem o uso da rede de Hopfield requer que esta se comporte como um sistema estável, com múltiplos pontos de equilíbrio também estáveis. As condições que garantem este tipo de comportamento são expostas nesta seção.

Para analisar a estabilidade e evolução da rede de Hopfield, há a necessidade de definir uma função de energia ou função de Lyapunov que está associada com sua dinâmica [Vidyasagar, 2002]. Em outras palavras, tem-se que provar que o sistema dissipava energia com o passar do tempo, em consideração às certas condições que lhe são impostas. Para isso, deve-se mostrar que suas derivadas temporais são sempre menores ou iguais que zero, conforme o segundo método de Lyapunov. Visando tal propósito, uma função de Lyapunov para a rede de Hopfield, cujos neurônios são alterados de forma assíncrona (um por vez), é definida por [Hopfield, 1984]:

$$E(t) = -\frac{1}{2} \mathbf{v}(t)^T \cdot \mathbf{W} \cdot \mathbf{v}(t) - \mathbf{v}(t)^T \cdot \mathbf{f}^0 \quad (7.5)$$

A partir de (7.5), obtém-se então a expressão para as suas derivadas temporais, isto é:

$$\dot{E}(t) = \frac{dE(t)}{dt} = (\nabla_{\mathbf{v}} E(t))^T \cdot \dot{\mathbf{v}}(t) \quad (7.6)$$

onde  $\nabla_{\mathbf{v}}$  é o operador gradiente em relação ao vetor  $\mathbf{v}$ .

Impondo a primeira condição, de que a matriz de pesos seja simétrica ( $\mathbf{W} = \mathbf{W}^T$ ), obtém-se a seguinte relação de (7.6) com (7.5):

$$\nabla_{\mathbf{v}} E(t) = -\mathbf{W} \cdot \mathbf{v}(t) - \mathbf{f}^0 \quad (7.7)$$

A partir da análise da expressão (7.1), assumindo que o termo de decaimento passivo seja nulo, conclui-se o seguinte relacionamento com (7.7):

$$\nabla_{\mathbf{v}} E(t) = -\mathbf{u}(t) \quad (7.8)$$

Logo, substituindo (7.8) em (7.6), tem-se:

$$\begin{aligned} \dot{E}(t) &= -\dot{\mathbf{u}}(t)^T \cdot \dot{\mathbf{v}}(t) \\ &= -\sum_{j=1}^n \dot{u}_j(t) \cdot \dot{v}_j(t) = -\sum_{j=1}^n \dot{u}_j(t) \cdot \frac{\partial v_j}{\partial u_j} \cdot \frac{\partial u_j}{\partial t} \\ &= -\sum_{j=1}^n \underbrace{(\dot{u}_j(t))^2}_{\text{parcela (i)}} \cdot \underbrace{\frac{\partial v_j}{\partial u_j}(t)}_{\text{parcela (ii)}} \end{aligned} \quad (7.9)$$

Para concluir a demonstração, visando-se extrair as condições essenciais para que a rede de Hopfield seja estável, basta então mostrar que as derivadas temporais da expressão (7.9) são sempre menores ou iguais que zero. Como o sinal da referida expressão já é negativo, resta mostrar que tanto a parcela (i) como a parcela (ii) produzirá sinais sempre positivos.

De fato, a parcela (i) sempre fornece resultado positivo, independentemente do valor de seu argumento, pois está elevada ao quadrado. Finalmente, é importante mostrar que o sinal da parcela (ii) será também sempre positivo. Para tanto, isto será invariavelmente verdadeiro na situação de se utilizar funções de ativação monótonas crescentes, tais como a função logística e a tangente hiperbólica. A figura 7.3 ilustra tal afirmação.

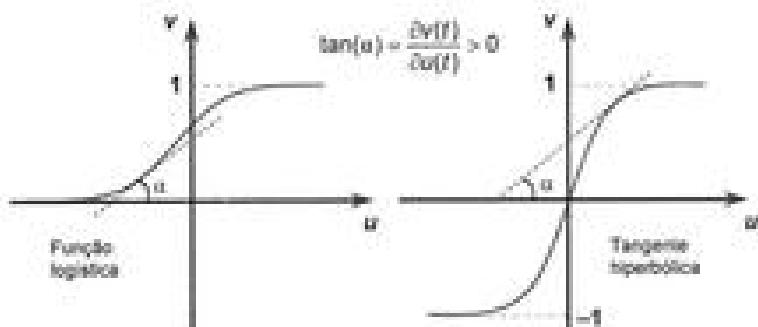


Figura 7.3 – Ilustração de derivadas parciais de primeira ordem

Claramente, pode-se observar que todas as derivadas de primeira ordem (coeficientes angulares das retas tangentes aos pontos da curva) serão positivas em todos os pontos de seus respectivos domínios de definição.

Portanto, tem-se assim as duas condições essenciais para que o comportamento dinâmico da rede de Hopfield seja estável, isto é:

- i. A matriz de pesos ( $W$ ) deve ser simétrica.
- ii. A função de ativação ( $g(\cdot)$ ) deve ser monótona crescente.

Em suma, considerando a condição de que a matriz  $W$  seja simétrica, e que sua função de ativação seja monótona crescente, os resultados da demonstração estabelecem que, dado qualquer conjunto de condições iniciais  $x^{(0)}$ , a rede convergirá para um ponto de equilíbrio estável.

A figura 7.4 mostra de forma ilustrativa um conjunto de pontos de equilíbrio e seus respectivos campos de atração, que foi produzido por uma rede de Hopfield constituída de dois neurônios, cujas saídas são dadas por  $v_1$  e  $v_2$ .

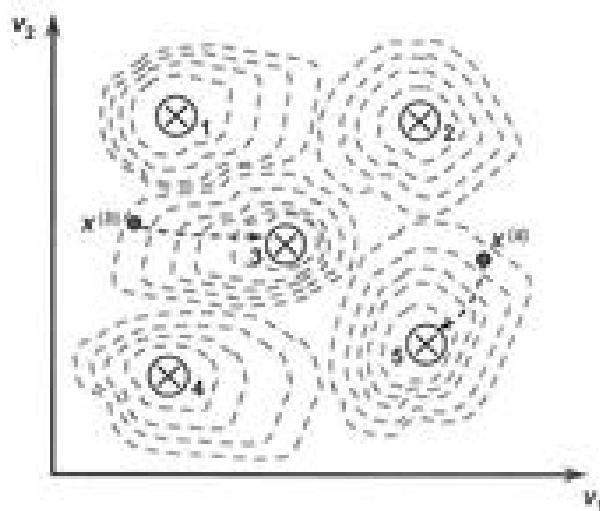


Figura 7.4 – Puntos de equilíbrio e seus campos de atração

Os cinco pontos de equilíbrio (atratores), representados sequencialmente na figura 7.4, seriam então aqueles que produziriam os menores va-

lores para a função de energia dada em (7.5), ou seja, são os estados que minimizam a função de energia da rede. Como exemplo, conforme também mostrado na figura 7.4, a saída da rede convergiria para o quinto ponto de equilíbrio caso fosse inicializada no ponto  $x^4$ , pois este estaria posicionado em seu respectivo campo de atração; de modo que a convergência se daria para o terceiro ponto de equilíbrio se a rede fosse inicializada no ponto  $x^3$ .

Generalizando, como a rede de Hopfield é determinística, tem-se que, para quaisquer posições iniciais que estejam dentro da região de atração de um ponto de equilíbrio, a rede então sempre tenderá a convergir (de forma assintótica) para aquele referido ponto. A quantidade exata de pontos de equilíbrio e suas localizações são determinadas pelos parâmetros livres ( $W$  e  $P$ ) da rede de Hopfield (equação 7.5), sendo que os formatos geométricos de seus respectivos campos de atração estão também relacionados com as características da função de energia que foi definida para o problema.

## 7.4 – Memórias associativas

Uma das aplicações mais difundidas das redes de Hopfield diz respeito às memórias associativas binárias, também denominadas de memórias endereçáveis pelo conteúdo. A finalidade embutida por trás de uma memória associativa está em recuperar (restaurar) corretamente um padrão que foi previamente armazenado em sua estrutura, a partir de uma amostra parcial (incompleta) ou ruídosca (distorcida) do mesmo [Hopfield, 1982]. Para propósitos de clareza, a figura 7.5 ilustra a versão original de um padrão específico e duas versões incorretas de sua amostra.

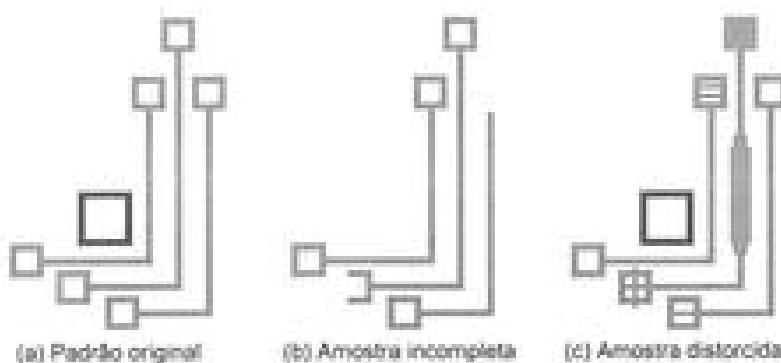


Figura 7.5 – Versões de padrões incorretos em memórias associativas

Para a situação deste exemplo, dado então uma amostra incompleta (figura 7.5b) ou ruim (figura 7.5c), o objetivo da memória associativa seria recuperar o padrão originalmente correto (figura 7.5a) que já tinha sido anteriormente armazenado em sua estrutura.

Face ao problema especificado aqui, a rede de Hopfield pode também ser utilizada com a finalidade de funcionar como uma memória associativa. Assim como em todas as aplicações relacionadas com o uso de redes de Hopfield, o desafio está na especificação apropriada dos seus parâmetros livres que levem à minimização da função de energia correspondente. Para tanto, dois métodos clássicos têm sido utilizados com frequência, ou seja, o método do produto externo e o método da matriz pseudo-inversa.

#### 7.4.1 - Método do produto externo

A forma mais simples para a determinação da matriz de pesos  $W$  e do vetor de limiares  $\theta$ , visando-se parametrização de memórias associativas, é aquela proposta pelo próprio Hopfield, cuja inspiração advém da aplicação do método de aprendizagem de Hebb.

De acordo com Hopfield (1982), dada uma quantidade  $p$  de padrões  $\{z\}$  a serem armazenados na memória, constituídos por  $n$  elementos cada um, os parâmetros livres da rede de Hopfield são definidos por:

$$W = \frac{1}{n} \sum_{k=1}^p z^{(k)} \cdot (z^{(k)})^T \quad (7.10)$$

$$\theta^0 = 0 \quad (7.11)$$

em que utiliza como ativação a função sinal dada na expressão (1.6). De fato, neste caso, tal função seria similar à função tangente hiperbólica com  $\beta$ -elevado.

No caso das memórias associativas, a diagonal da matriz de pesos deve ter valores nulos (indicação de ausência de auto-realimentação). Para tanto, reescrevendo a expressão (7.10), tem-se:

$$W = \frac{1}{n} \sum_{k=1}^p \underbrace{z^{(k)} \cdot (z^{(k)})^T}_{\text{parte } \frac{n}{ij}} - \underbrace{\frac{p}{n} \cdot I}_{\text{parte } \frac{n}{ij}} \quad (7.12)$$

onde  $I \in \mathbb{R}^{n \times n}$  é a matriz identidade.

Certamente, a parcela  $(i)$  da expressão (7.12) realiza o produto externo entre os elementos de cada um dos padrões a ser armazenados, sendo que a parcela  $(ii)$  simplesmente neutraliza os elementos da diagonal principal ( $W_{ii} = 0$ ).

Como exemplo do processo de montagem da matriz  $W$ , considera-se dois padrões a serem introduzidos numa memória associativa, conforme diagramados na figura 7.6.

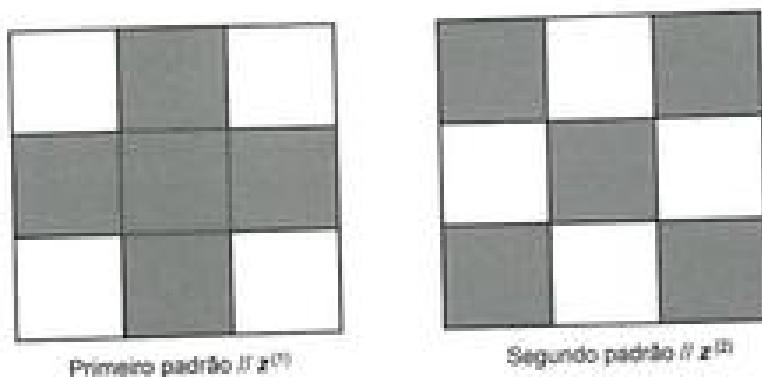


Figura 7.6 – Exemplos de padrões a serem armazenados na memória

Nas imagens a ser armazenadas, definidas em grids de dimensão  $3 \times 3$ , as quadriculas (pixels) escuras são representadas pelo valor 1, ao passo que as quadriculas brancas pelo valor -1. Tais atribuições são diretamente assumidas por corresponder aos valores de saída da função de ativação sinal.

Nesta condição, os vetores  $x^{(1)}$  e  $x^{(2)}$  são constituídos pela concatenação das linhas que compõem cada um dos seus grids, isto é:

$$x^{(1)} = [\underbrace{-1 \quad +1 \quad -1}_{1^{\text{a}} \text{ linha}} \quad \underbrace{+1 \quad +1 \quad +1}_{2^{\text{a}} \text{ linha}} \quad \underbrace{-1 \quad +1 \quad -1}_{3^{\text{a}} \text{ linha}}]^T$$

$$x^{(2)} = [\underbrace{+1 \quad -1 \quad +1}_{1^{\text{a}} \text{ linha}} \quad \underbrace{-1 \quad +1 \quad -1}_{2^{\text{a}} \text{ linha}} \quad \underbrace{+1 \quad -1 \quad +1}_{3^{\text{a}} \text{ linha}}]^T$$

Aplicando-se a expressão (7.12), com  $p = 2$  e  $n = 9$ , obtém-se a referida matriz de pesos da rede de Hopfield, ou seja:

$$W = \begin{bmatrix} 0 & -0.22 & 0.22 & -0.22 & 0 & -0.22 & 0.22 & -0.22 & 0.22 \\ -0.22 & 0 & -0.22 & 0.22 & 0 & 0.22 & -0.22 & 0.22 & -0.22 \\ 0.22 & -0.22 & 0 & -0.22 & 0 & -0.22 & 0.22 & -0.22 & 0.22 \\ -0.22 & 0.22 & -0.22 & 0 & 0 & 0.22 & -0.22 & 0.22 & -0.22 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.22 & 0.22 & -0.22 & 0.22 & 0 & 0 & -0.22 & 0.22 & -0.22 \\ 0.22 & -0.22 & 0.22 & -0.22 & 0 & -0.22 & 0 & -0.22 & 0.22 \\ -0.22 & 0.22 & -0.22 & 0.22 & 0 & 0.22 & -0.22 & 0 & -0.22 \\ 0.22 & -0.22 & 0.22 & -0.22 & 0 & -0.22 & 0.22 & -0.22 & 0 \end{bmatrix}$$

Desse forma, considerando uma versão parcial ou ruídosas dos padrões  $\mathbf{x}^{(1)}$  ou  $\mathbf{x}^{(2)}$ , na qual será apresentada às entradas da rede, esta convergirá para uma das respectivas versões que foram originalmente armazenadas em sua estrutura.

#### 7.4.2 – Método da matriz pseudo-inversa

A obtenção da matriz de pesos  $W$ , por meio do método da matriz pseudo-inversa, é realizada a partir da minimização do erro entre uma versão parcial (ou ruídosas) frente à sua respectiva versão original. Para tanto, sejam as seguintes matrizes:

$$\mathbf{Z} = [\mathbf{z}^{(1)} \quad \mathbf{z}^{(2)} \quad \mathbf{z}^{(3)} \quad \dots \quad \mathbf{z}^{(p)}] \quad (7.13)$$

$$\mathbf{R} = [\mathbf{r}^{(1)} \quad \mathbf{r}^{(2)} \quad \mathbf{r}^{(3)} \quad \dots \quad \mathbf{r}^{(p)}] \quad (7.14)$$

onde  $\mathbf{Z}$  é uma matriz que contém todos os  $p$  padrões de dimensão  $n$ , representados pelos vetores  $\mathbf{z}^{(i)}$ , ao passo que  $\mathbf{R}$  é uma matriz que contém as versões ruidosas  $\mathbf{r}^{(i)}$  desses  $p$  padrões.

O erro existente entre  $\mathbf{Z}$  e  $\mathbf{R}$  pode ser computado por intermédio da norma euclidiana, que é dada por:

$$Erro = \|\mathbf{Z} - \mathbf{V}\| \quad (7.15)$$

onde  $\mathbf{V}$  é a matriz representando a saída da rede em relação à matriz  $\mathbf{R}$ , que

foi apresentada em suas entradas. Em situação de convergência, tem-se o seguinte relacionamento para a memória associativa:

$$\mathbf{V} = \mathbf{W} \cdot \mathbf{R} \quad (7.16)$$

- Substituindo o valor da expressão (7.16) em (7.15), obtém-se:

$$Error = \|\mathbf{Z} - \mathbf{W} \cdot \mathbf{R}\| \quad (7.17)$$

Portanto, deseja-se obter valores para os elementos da matriz de pesos  $\mathbf{W}$  que minimize a expressão (7.17). Por meio da álgebra linear e das técnicas de identificação de sistemas [Ljung, 1998], a minimização desta norma pode ser obtida por:

$$\mathbf{W} = \mathbf{Z} \cdot (\mathbf{R}^T \cdot \mathbf{R})^{-1} \cdot \mathbf{R}^T \quad (7.18)$$

Em que a parcela  $(\mathbf{R}^T \cdot \mathbf{R})^{-1} \cdot \mathbf{R}^T$  é a pseudo-inversa da matriz  $\mathbf{R}$ , conhecida também como pseudo-inversa de Penrose-Moore.

#### 7.4.3 – Capacidade de armazenamento das memórias

Um dos questionamentos frequentes a respeito da rede de Hopfield, quando usada como memória associativa, diz respeito à quantidade de padrões que podem ser inseridos em sua estrutura, visando-se, posteriormente, as respectivas recuperações destes frente à apresentação de uma amostra incompleta ou distorcida.

Baseado em inúmeros experimentos computacionais, descreve-se, em Hopfield (1982), que a capacidade de armazenamento  $\{C^{opt}\}$  de padrões das memórias associativas, objetivando uma recuperação de padrões relativamente com poucos erros, é dada por:

$$C^{Hopf} = 0,15 \cdot n \quad (7.19)$$

onde  $n$ , conforme definido anteriormente, é a quantidade de elementos (dimensão) de cada padrão a ser armazenado.

Por meio de análises de probabilidade, apresentam-se, em Amit (1992) e Haykin (1999), alguns resultados mais precisos e elaborados sobre a capacidade de armazenamento das redes de Hopfield. Os estudos realizados revelam que a capacidade máxima de armazenamento ( $C^{\text{Max}}$ ), considerando uma recuperação quase sem erros, seria definida por:

$$C^{\text{Max}} = \frac{n}{2 \cdot \ln(n)} \quad (7.20)$$

Já para uma recuperação perfeita, com probabilidade beirando 100% de acertos, a capacidade de armazenamento de padrões ( $C^{100\%}$ ) seria estimada em:

$$C^{100\%} = \frac{n}{4 \cdot \ln(n)} \quad (7.21)$$

Para efeitos comparativos, em função dos resultados advindos dessas três expressões anteriores, a tabela 7.1 mostra as capacidades de armazenamento da rede de Hopfield para diversos valores de  $n$ .

Tabela 7.1 – Capacidades de armazenamento da rede de Hopfield

Dimensão	$C^{\text{Max}}$	$C^{\text{Max}}$	$C^{100\%}$
$n = 20$	3,0	3,3	1,7
$n = 50$	7,5	8,4	3,2
$n = 100$	15,0	10,9	5,5
$n = 500$	75,0	40,2	20,1
$n = 1000$	150,0	72,4	36,2

Pela análise da tabela 7.1 é possível verificar que, quando a precisão requerida durante a recuperação deixar de ser um fator tão imperativo, pode-se então armazenar muito mais padrões na memória associativa, conforme os valores refletidos por  $C^{\text{Max}}$ . Em contraste, quando a precisão

da recuperação se torna muito elevada, a quantidade máxima de padrões a serem armazenados ( $C^{\max}$ ) decai substancialmente com o aumento de suas dimensões, atingindo-se valores mínimos de armazenamento quando se espera 100% de acertos na recuperação.

Com efeito, uma das principais fragilidades das redes de Hopfield, funcionando como memórias associativas, seriam as suas baixas capacidades de armazenamento, que são proporcionais às dimensões dos padrões a serem guardados.

Por outro lado, quando a capacidade de armazenamento despreza as quantidades recomendadas por (7.19) e (7.20), isto é, números excessivos de padrões passam a ser inseridos dentro da estrutura da rede, há então o aparecimento de estados espúrios. Embora sejam pontos de equilíbrio estacionais, tais estados não corresponderão a nenhum dos padrões previamente armazenados. Neste caso, a matriz  $W$  possuirá autovalores degenerados. Uma detalhada investigação teórica sobre este tópico, que envolve a convergência da rede de Hopfield frente às propriedades da matriz  $W$  e de seus subespaços, é examinado em Aiyer *et alii* (1990).

Diversas outras investigações têm sido ainda conduzidas com a finalidade de se incrementar a capacidade de armazenamento de padrões nas memórias associativas, quando da utilização de redes de Hopfield, tais como aquelas propostas em Garcia & Moreno (2004), Lee (2006), Li *et alii* (1989) e Muezzinoglu *et alii* (2005).

## 7.5 – Aspectos de projeto de redes de Hopfield

Diferentemente das arquiteturas de redes neurais artificiais apresentadas nos capítulos anteriores, tem-se que os parâmetros livres da rede de Hopfield, definidos pela matriz de pesos  $W$  e pelo vetor de limiares  $P$ , são obtidos de maneira explícita em grande parte de suas aplicações, o que implica na dispensa de algoritmos de treinamento.

De fato, a maioria dos problemas tratados pela rede de Hopfield é derivada da especificação de uma função de energia, conforme expressão definida em (7.5), a qual é representativa de seu comportamento dinâmico. Quanto mais conhecimento se tenha da área temática que envolve um problema específico, mais subsídios estarão disponíveis para se projetar adequadamente as respectivas funções de energia.

Em suma, o projeto de uma rede de Hopfield pode ser desenvolvido atentando-se para os seguintes aspectos:

- i. A função de energia do problema deve ser escrita na forma dada por (7.5);
- ii. Para problemas envolvendo otimização de sistemas, a função de energia normalmente é formulada por intermédio da função objetivo (função custo ou função de desempenho);
- iii. Para problemas de otimização restrita, tanto a função objetivo como as restrições estruturais devem ser também escritas conforme a expressão (7.5). Um exemplo deste mapeamento é apresentado no capítulo 20;
- iv. A matriz de pesos  $W$  pode ainda ser composta por elementos que sejam representados por funções algébricas. Entretanto, tais funções precisam ocupar posições simétricas dentro de  $W$ ;
- v. Independentemente do problema a ser mapeado pela rede de Hopfield, deve-se sempre observar a simetria em sua matriz de pesos;
- vi. A função de ativação da rede de Hopfield deve ser sempre monótona crescente, visando-se para tanto os propósitos de garantia de estabilidade;
- vii. Em diversas aplicações de cunho prático, os elementos do vetor de limitares  $\{P\}$  podem assumir valores nulos;
- viii. O domínio de variação de cada variável do problema a ser mapeado deve ser abrangido pelos limites da função de ativação considerada;
- ix. O resultado da rede de Hopfield, correspondente a alguma solução do problema que está sendo tratado, é sempre representado por um de seus pontos de equilíbrio;
- x. No caso de problemas de memórias associativas, deve-se atentar para as suas capacidades de armazenamento frente à precisão requerida para a recuperação dos padrões.

As instruções (em pseudocódigo) descrevendo a operação da rede de Hopfield discreta, representada pelas expressões (7.3) e (7.4), são fornecidas na sequência.

**Inicio {Algoritmo Hopfield – Operação em Tempo Discreto}**

<1> Especificar a matriz de pesos  $W$  e o vetor de limiares  $I^b$ ;  
 <2> Apresentar vetor inicial de entradas ( $x^{(0)}$ );  
 <3>  $v^{atual} \leftarrow x^{(0)}$ ;  
 <4> Repetir as instruções:  
 {<4.1>  $v^{anterior} \leftarrow v^{atual}$ ;  
 <4.2>  $u \leftarrow W \cdot v^{anterior} + I^b$ ; {conforme (7.3)}  
 <4.3>  $v^{atual} \leftarrow g(u)$ ; {conforme (7.4)}  
 Até que:  $v^{atual} \equiv v^{anterior}$   
 <5>  $v^{final} \leftarrow v^{atual}$  { $v^{final}$  representa um ponto de equilíbrio}

**Fim { Algoritmo Hopfield – Operação em Tempo Discreto}**

A função de ativação a ser utilizada no algoritmo depende do tipo de problema a ser mapeado pela rede de Hopfield, sendo que, para o caso das memórias associativas, a função sinal é uma das mais empregadas.

Em resumo, a partir de um estado inicial  $x^0$ , a rede tende a convergir para um de seus pontos de equilíbrio que sejam estáveis, cuja evolução permanecerá estática ( $v^{atual} \equiv v^{anterior}$ ) quando esta condição for atingida.

## 7.6 – Aspectos de implementação em hardware

A relativa facilidade de implementação em hardware análogo é também um dos principais motivos da popularidade da rede de Hopfield. Para tanto, em Hopfield (1984), além da proposição da rede em tempo contínuo, foi ainda demonstrado que esta poderia ser representada por componentes eletrônicos bem básicos, tais como amplificadores operacionais, resistores e capacitores. Nesta situação, os neurônios em si são modelados pelos próprios amplificadores operacionais, cuja relação entrada/saída destes faz o papel da função de ativação logística. A figura 7.7 ilustra o diagrama esquemático de implementação da rede de Hopfield em hardware.

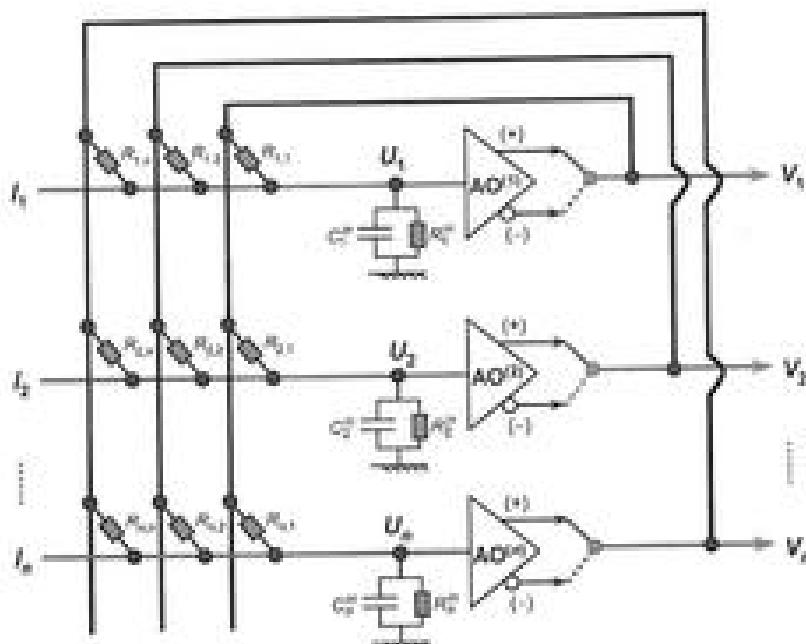


Figura 7.7 – Diagrama do hardware análogo da rede de Hopfield

A constante de tempo que define a evolução de cada  $j$ -ésimo amplificador operacional ( $AO^{(j)}$ ) é especificada pelo ajuste dos seguintes parâmetros internos:

$R_j^{(i)}$  → Resistor de entrada do  $j$ -ésimo amplificador operacional.

$C_j^{(i)}$  → Capacitor de entrada do  $j$ -ésimo amplificador operacional.

O parâmetro  $U_j$  na figura 7.7 indica a tensão na entrada do amplificador operacional, sendo que  $V_j$  é a sua respectiva tensão de saída. Os pesos  $W_{ij}$  da rede, representando a conexão entre dois neurônios, são definidos por uma corrente de intensidade que interliga a saída do  $j$ -ésimo amplificador à entrada do  $i$ -ésimo amplificador. Esta conexão é realizada por meio de um resistor cujo valor é dado por:

$$R_j = \frac{1}{|W_{ji}|} \quad (7.22)$$

Sendo que, se a sinapse (conexão) for excitatória ( $W_{ji} \geq 0$ ), então este resistor será diretamente conectado à saída convencional (+) do  $j$ -ésimo amplificador; caso

contrário, considerando-se uma sinapse inibitória ( $W_{ij} < 0$ ), o resistor será então conectado à saída inversora ( $-$ ) do respectivo amplificador. Portanto, os sinais dos elementos da matriz  $W$  definirão os pontos de conectividade entre todos os amplificadores operacionais (neurônios). A corrente de entrada de qualquer neurônio é formada pela soma das correntes fluindo através do conjunto de resistores que conecta suas entradas às saídas dos outros neurônios.

Conforme também indicado na figura 7.7, o circuito equivalente representativo da rede de Hopfield inclui ainda uma corrente de entrada  $I_j$ , a ser fornecida externamente para cada neurônio, que é correspondente ao respectivo valor de seu limiar, ou seja,  $I_j = I_j^*$ .

Finalmente, a equação que descreve a evolução deste circuito em relação ao tempo, constituído tipicamente por amplificadores operacionais, pode ser obtida mediante a aplicação das leis de Ohm e Kirchoff [Boylestad & Nashelsky, 2004]. Assumindo que todos os amplificadores operacionais possuem as mesmas configurações internas ( $R = R_j^*$  e  $C = C_j^*$ ), obtém-se:

$$C \cdot \frac{dU_j(t)}{dt} = -\eta \cdot U_j(t) + \sum_{j=1}^n W_{ji} \cdot V_i(t) + I_j, \text{ onde } j = 1, \dots, n \quad (7.23)$$

$$\text{onde: } \eta = \frac{1}{R} + \sum_{j=1}^n \frac{1}{R_j}$$

Normalizando (7.23) em relação ao valor da capacitância  $C$ , conclui-se que é similar àquela expressão definida em (7.1). Portanto, o diagrama eletrônico mostrado na figura 7.7 corresponde ao circuito equivalente, em hardware analógico, da rede de Hopfield. Consequentemente, qualquer problema de cálculo prático que seja mapeado por meio de simulações computacionais pode também ser convertido para o seu respectivo circuito equivalente.

## 7.7 – Exercícios

- 1) Explique quais são as condições essenciais que devem ser impostas às equações que definem a dinâmica da rede de Hopfield a fim de que produza soluções estáveis.
- 2) Discorra sobre a importância do parâmetro  $\beta$  para o processo de convergência da rede de Hopfield em direção aos pontos de equilíbrio estáveis.

- 3) Explique se a função de ativação do tipo gaussiana poderia ser utilizada nos neurônios da rede de Hopfield.
- 4) Mostre de maneira detalhada qual a sequência de passos necessários para projetar redes de Hopfield que sejam dinamicamente estáveis.
- 5) Considerando um problema relacionado com memórias associativas, discorra se a aplicação da rede de Hopfield pode produzir pontos de equilíbrio estáveis que não sejam uma solução válida para o problema.
- 6) Discorra se há alguma objeção a ser realizada nos parâmetros da rede de Hopfield em tempo discreto quando da sua inicialização com valores nulos ( $x^{(0)} = 0$ ).
- 7) Uma equipe de engenheiros e cientistas projetou uma rede de Hopfield visando uma aplicação específica. Contudo, todos os elementos da matriz de pesos possuíam os mesmos valores, que eram diferentes de zero. Explique se a formulação alcançada poderia estar incorreta.
- 8) Discorra como deve ser ajustada a função de ativação tangente hiperbólica com o intuito de utilizá-la no mapeamento de memórias associativas em hardware (rede de Hopfield contínua).
- 9) Considerando um problema de memórias associativas usando rede de Hopfield, em que cada padrão a ser armazenado é constituído de  $n$  componentes, determine qual seria a quantidade máxima de pesos sinápticos não nulos que a respectiva matriz de pesos poderia conter.
- 10) Seja um problema de otimização combinatória que utiliza uma rede de Hopfield para a obtenção da solução. Nas diversas simulações executadas observou-se que há somente dois pontos de equilíbrio estáveis. Discorra então sobre os seguintes itens:

- Quais são os possíveis parâmetros que condicionam a convergência para um desses pontos de equilíbrio?
- Que critério deve ser utilizado para escolher qual desses pontos de equilíbrio que corresponderá à solução final do problema.

## 7.8 – Projeto prático

Um sistema de transmissão de imagens (codificadas por 45 bits) envia seus sinais através de um link de comunicação. Ao chegar no sistema de

recepção, a informação é decodificada visando a recuperação fiel da imagem previamente enviada. As quatro imagens (informações) que estão sendo transmitidas são representadas pela figura 7.8.

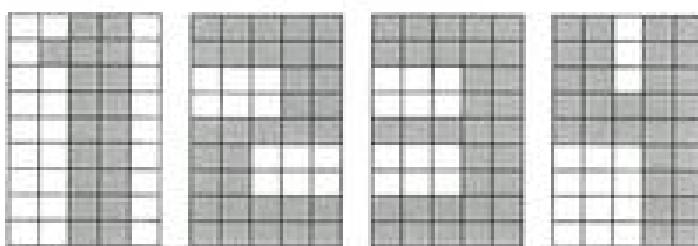


Figura 7.8 – Imagens a serem enviadas pelo transmissor

Entretanto, durante a transmissão, as imagens são corrompidas por ruídos, transformando-as em representações incompletas ou distorcidas, quando de suas decodificações pelo receptor de imagens.

Visando resolver o problema, pretende-se implementar uma memória associativa, por meio de uma rede de Hopfield constituída de 45 neurônios, a fim de armazenar e recuperar as imagens (padrões) definidas na figura 7.8. Para tanto, consideram-se as seguintes convenções:

- Pixel branco é codificado com valor -1;
- Pixel escuro é codificado com valor +1;
- Cerca de 20% dos pixels são corrompidos aleatoriamente durante a transmissão, ou seja, alguns que valiam -1 passam a valer +1 e vice-versa;
- A matriz de pesos  $W$  será definida por meio do método do produto externo;
- A função de ativação a ser utilizada em todos os neurônios será a função sinal (1.6), ou a tangente hiperbólica (1.11), com parâmetro de inclinação  $\{\beta\}$  igual a 100.

Utilizando-se tais informações, faça os exercícios seguintes quando do término da implementação computacional da rede de Hopfield.

- 1) Simule doze situações de transmissão (três para cada padrão), conforme esquema ilustrativo apresentado na figura 7.9.

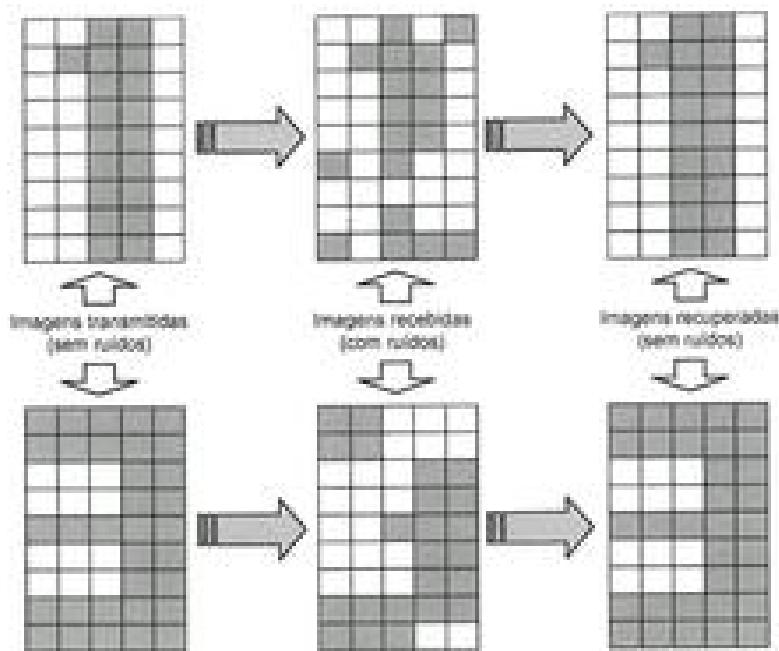


Figura 7.9 – Esquema ilustrativo de imagens transmitidas, recebidas e recuperadas.

- 2) Mostre, em cada situação, a imagem distorcida e a imagem limpa já recuperada.
- 3) Explique o que acontece quando se aumenta de maneira excessiva o nível de ruído.



Aleksandr Mikhajlovich Ljapunov  
(Александр Михайлович Ляпунов)

# Redes auto-organizáveis de Kohonen

## 8.1 – Introdução

Grande parte das estruturas de redes neurais artificiais apresentadas nos capítulos precedentes necessita de um conjunto de padrões de entradas e saídas para que possam ser devidamente treinadas. Os ajustes de seus parâmetros livres (matrizes de pesos e limiares) são realizados a partir das apresentações sucessivas dessas amostras de treinamento, configurando-se assim numa aprendizagem supervisionada.

Entretanto, em certas aplicações, somente o conjunto de padrões de entradas está disponível, inexistindo, para tanto, as respectivas saídas desejadas. Por outro lado, essas amostras possuem informações relevantes sobre o comportamento daquele sistema em que foram extraídas.

A maioria das redes utilizadas em problemas com essa configuração se auto-organizam por meio de métodos de treinamento competitivos, os quais têm a capacidade de detectar similaridades, regularidades e correlações entre os padrões do conjunto de entrada, agrupando-os em classes (*clusters*). Cada uma possui então características particulares que estão relacionadas com situações e condições que regem o funcionamento do processo. Assim, a identificação destes *clusters* é importante para o entendimento das relações entre os seus elementos constituintes, permitindo-se ainda identificar as funções de um componente ou amostra com base nos atributos dos outros elementos que fazem parte do grupo.

Uma das estruturas de redes neurais artificiais mais difundidas, em face ao contexto de auto-organização, são os mapas auto-organizáveis propostos pelo finlandês Teuvo Kohonen, os quais ficaram também conhecidos como redes auto-organizáveis de Kohonen, ou, simplesmente, redes de Kohonen, cujo trabalho pionero foi formulado em Kohonen (1982, 1984).

Assim como as redes recorrentes de Hopfield, as pesquisas desenvolvidas por Kohonen ainda atraíram a atenção da comunidade científica daquela época, contribuindo para o avivamento cada vez mais crescente da área de redes neurais artificiais, cujo renascimento definitivo se desdobrou com a proposição do algoritmo *backpropagation* por Rumelhart *et al.* (1986).

As redes de Kohonen encontram inspiração biológica no córtex cerebral, onde a ativação de uma região específica corresponde à resposta frente a determinado estímulo sensorial, por exemplo, estímulo motor, visual ou auditivo.

Embora a rede de Kohonen seja utilizada para diversas aplicações, em diferentes áreas do conhecimento, os trabalhos mais bem disseminados versam sobre problemas que envolvem classificação de padrões e agrupamento de dados (clusterização).

## 8.2 – Processo de aprendizado competitivo

Para descrever os passos envolvidos com o processo de aprendizado competitivo utilizado na rede de Kohonen, considera-se uma estrutura neural constituída de apenas uma camada neural, conforme ilustrada na figura 8.1.

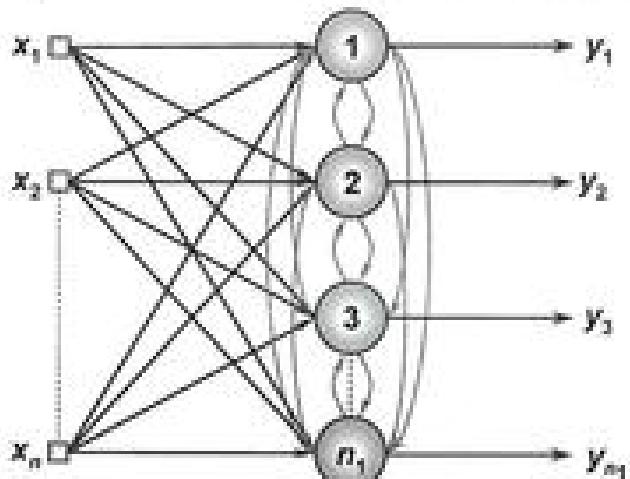


Figura 8.1 – Estrutura neural básica de rede competitiva

As conexões laterais entre os neurônios da figura 8.1 assumem aqui o papel de que um neurônio pode influenciar na resposta de saída produzida por outro neurônio. O significado dessas conexões laterais será detalhado na próxima seção.

Para efeitos de simplicidade de notação, visto que a estrutura neural ilustrada na figura 8.1 é constituída de apenas uma camada, assume-se então a seguinte convenção para os seus vetores de pesos:

$$\mathbf{w}^{(1)} = [W_{1,1} \ W_{1,2} \ \dots \ W_{1,n}]^T \quad \{\text{vetor de pesos do } 1^{\circ} \text{ neurônio}\}$$

$$\mathbf{w}^{(2)} = [W_{2,1} \ W_{2,2} \ \dots \ W_{2,n}]^T \quad \{\text{vetor de pesos do } 2^{\circ} \text{ neurônio}\}$$

$$\mathbf{w}^{(3)} = [W_{3,1} \ W_{3,2} \ \dots \ W_{3,n}]^T \quad \{\text{vetor de pesos do } 3^{\circ} \text{ neurônio}\}$$

(...)

$$\mathbf{w}^{(n)} = [W_{n,1} \ W_{n,2} \ \dots \ W_{n,n}]^T \quad \{\text{vetor de pesos do último neurônio}\}$$

O princípio básico para o processo de aprendizado competitivo é a concorrência entre os neurônios, tendo-se o objetivo de se sair vencedor da prova de competição, pois se ressalta que aqui o processo é não-supervisionado (sem saída desejada). O prêmio para aquele que venceu a competição com os outros neurônios será o ajuste de seus pesos, proporcionalmente aos valores do padrão de entrada apresentado, visando-se assim aperfeiçoar o seu estado para a próxima competição (próximo padrão a ser apresentado).

Nesta circunstância, se todas as conexões laterais deste neurônio vencedor forem nulas (ausência de conexões laterais), implica que somente os seus pesos serão ajustados, isto é, assume-se a estratégia do "vencedor leva tudo" (*winner-take-all*). Caso contrário, para a situação de haver valores para as conexões laterais com os seus vizinhos, então um ajuste proporcional a tais quantidades será também efetuado no vetor de pesos desses neurônios adjacentes.

Considerando tal contexto, haveria, portanto, a necessidade de se estabelecer uma regra que defina quem vai ser o neurônio vencedor. Uma das medidas mais utilizadas consiste em determinar o nível de proximidade existente entre o vetor de pesos de cada neurônio, frente ao vetor de entrada contendo os elementos da  $k$ -ésima amostra  $\{\mathbf{x}^k\}$ , a qual será apresentada nas

entradas da rede. Uma métrica de proximidade normalmente usada é a distância entre esses dois parâmetros, ou seja:

$$\text{dist}_j^{(k)} = \sqrt{\sum_{i=1}^n (x_i^{(k)} - w_i^{(j)})^2}, \quad \text{onde } j = 1, \dots, n_j \quad (8.1)$$

onde  $\text{dist}_j^{(k)}$  quantifica a distância (norma euclidiana) entre o vetor de entrada, representando a  $k$ -ésima amostra  $\{x^{(k)}\}$ , em relação ao vetor de pesos do  $j$ -ésimo neurônio  $\{w^{(j)}\}$ .

Assim, aquele neurônio  $j$  que obtiver a menor distância será declarado o vencedor da competição frente à amostra  $k$ . Como prêmio pela vitória, o vetor de pesos do neurônio vencedor será ajustado de maneira que este se aproxime ainda mais daquela amostra. Para tanto, utiliza-se o seguinte método de adaptação:

$$w_{\text{atual}}^{(v)} = w_{\text{anterior}}^{(v)} + \eta \cdot (x^{(k)} - w_{\text{anterior}}^{(v)}) \quad (8.2)$$

Em notação algorítmica, esta expressão (8.2) é equivalente a seguinte:

$$w^{(v)} \leftarrow w^{(v)} + \eta \cdot (x^{(k)} - w^{(v)}) \quad (8.3)$$

onde  $w^{(v)}$  denota o vetor de pesos do neurônio vencedor e o parâmetro  $\eta$  define a taxa de aprendizagem.

Visando obter uma maior eficiência do processo de aprendizagem, é desejável que se normalize (de forma unitária) todos os vetores de pesos dos neurônios, assim como todos aqueles vetores de amostras. A normalização unitária é efetuada bastando-se dividir cada vetor pelo seu respectivo módulo.

A figura 8.2 ilustra a regra de ajuste de pesos para o neurônio vencedor do processo competitivo, em que três neurônios (representados por  $w^{(1)}$ ,  $w^{(2)}$  e  $w^{(3)}$ ) disputam o torneio frente à apresentação do padrão  $x^{(k)}$ , sendo o mesmo composto por duas entradas  $x_1$  e  $x_2$ .

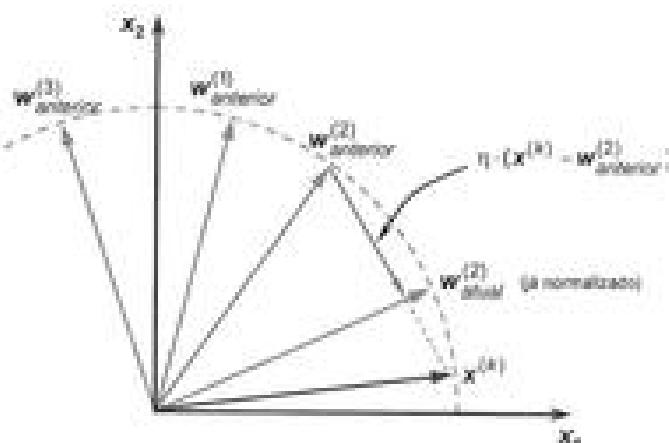


Figura 8.2 – Processo de ajuste do vetor de pesos do neurônio vencedor

A partir do exame desta figura, considerando que todos os vetores já se encontram unitariamente normalizados, observa-se que o vetor  $w^{(2)}$  é o vencedor da competição com  $w^{(1)}$  e  $w^{(3)}$ , pois é aquele que está mais perto do padrão  $x^{(2)}$ . Consequentemente, como recompensa, o seu vetor de pesos será ajustado, conforme a expressão (8.2), com o intuito de deixá-lo ainda mais próximo do vetor  $x^{(2)}$ . Por certo, o verdadeiro ajuste consistiu tão somente de rotacionar o neurônio vencedor em direção ao vetor representando a amostra.

**Início (Algoritmo Competitivo – Fase de Treinamento):**

- <1> Obter o conjunto de amostras de treinamento ( $x^{(1)}$ );
  - <2> Iniciar o vetor de pesos de cada neurônio considerando os valores das  $n_r$  primeiras amostras de treinamento;
  - <3> Normalizar os vetores de amostras e de pesos;
  - <4> Especificar a taxa de aprendizagem ( $\eta$ );
  - <5> Iniciar o contador de número de épocas (época  $\leftarrow 0$ );
  - <6> Repetir as instruções:
- <6.1> Para todas as amostras de treinamento ( $x^{(k)}$ ), fazer:
    - <6.1.1> Calcular as distâncias euclidianas entre  $x^{(k)}$  e  $w^{(j)}$ , conforme a expressão (8.1);
    - <6.1.2> Declarar como vencedor o neurônio  $j$  que contenha a menor distância euclidiana;
    - <6.1.3> Ajustar o vetor de pesos do neurônio vencedor conforme a expressão (8.3);
    - <6.1.4> Normalizar o vetor de pesos que foi ajustado na instrução precedente;
  - <6.2> época  $\leftarrow$  época + 1;
- Até que: não haja mudanças significativas nos pesos.
- Fim (Algoritmo Competitivo – Fase de Treinamento):**

Com efeito, após a convergência do algoritmo competitivo, cada um dos vetores de pesos dos neurônios estará posicionado nos centros dos aglomerados (*clusters*) que possuem características em comum. Tais aglomerados representarão as respectivas classes associadas ao problema de classificação de padrões, sendo que a quantidade destas ficará automaticamente vinculada ao número de neurônios utilizados na estrutura neural.

Considerando este ponto, vale ainda ressaltar que a quantidade inicial de neurônios a ser utilizado, visando representar as classes com características em comum, é de antemão desconhecido, pois o aprendizado é não-supervisionado. Assim, torna-se também de suma importância a obtenção de outras informações adicionais a respeito do problema a ser mapeado, seja por meio de especialistas ou pela aplicação de métodos estatísticos, que leve para uma estimativa inicial sobre a quantidade de possíveis classes associadas ao respectivo problema. Para a aplicação a ser formulada no capítulo 15, o número de neurônios foi definido em função dos possíveis tipos de adulterantes de café que poderiam ser identificados pela rede.

Como exemplo, em que se tenha quatro classes disponíveis, a figura 8.3 ilustra como ficaria uma eventual distribuição dos vetores de pesos da rede, após as suas devidas estabilizações. Cada um destes vetores, representativos das amostras do problema, era constituído por apenas dois sinais de entrada ( $x_1$  e  $x_2$ ).

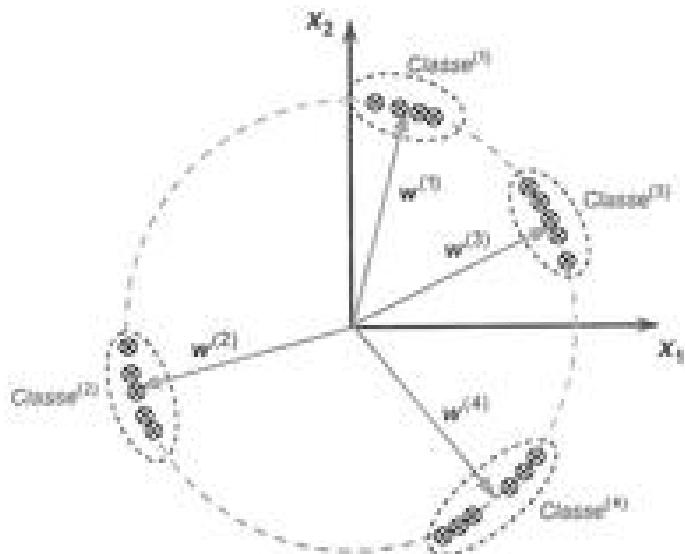


Figura 8.3 – Distribuição em espaço bidimensional dos vetores de pesos frente ao conjunto de amostras

Constata-se na figura 8.3, assim como na figura 8.2, que todos os vetores acabam ficando localizados dentro do círculo unitário, à medida que são bidimensionais e estão unitariamente normalizados. Verifica-se, ainda, que os quatro vetores de pesos se posicionaram nos centros dos aglomerados que representam as amostras (identificadas pelo símbolo “⊗”). Desta forma, depreende-se também que a estrutura neural representada nesta figura foi constituída por quatro neurônios, pois há somente quatro vetores de pesos ( $w^{(1)}$ ,  $w^{(2)}$ ,  $w^{(3)}$  e  $w^{(4)}$ ).

Nota-se ainda na ilustração da figura 8.3 que, se a estrutura neural fosse constituída por cinco neurônios, ao invés de quatro, as amostras pertencentes à Classe “0” seriam provavelmente divididas em duas classes, pois a mesma está atualmente agrupando dois subconjuntos (embora bem próximos) em apenas um único *cluster*.

Já para a aplicação representada na figura 8.4, cada amostra disponível é constituída de três sinais de entrada, sendo que todos os vetores de amostras e de pesos ficam confinados numa esfera de raio unitário, quando estão devidamente normalizados.

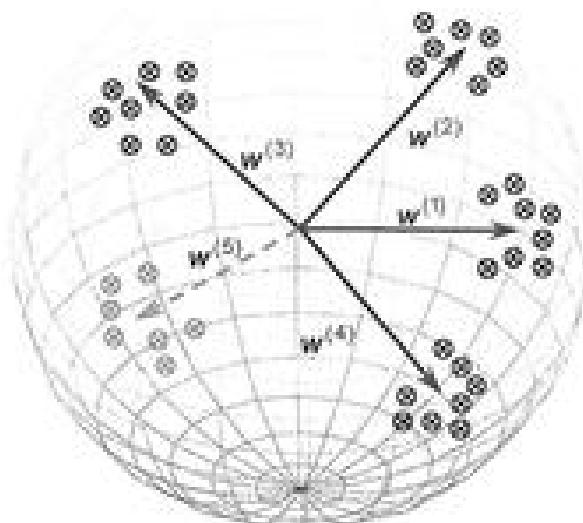


Figura 8.4 - Distribuição em espaço tridimensional dos vetores de pesos frente ao conjunto de amostras.

No caso da figura 8.4, percebe-se que cinco neurônios, representados pelos seus correspondentes vetores de pesos,  $\{w^{(1)}, w^{(2)}, w^{(3)}, w^{(4)} \text{ e } w^{(5)}\}$  estão posicionados no meio dos cinco agrupamentos (*clusters*) que configuraram o problema.

Logo, por meio das figuras 8.3 e 8.4, é possível verificar que os padrões a serem classificados são projetados no espaço onde se define os vetores de pesos, cuja dimensão é igual àquela em que os referidos padrões se encontram definidos. Assim, é possível também afirmar que um dos requisitos para melhoria da eficiência do processo de convergência é de que as variáveis de entrada sejam independentes, a fim de que a respectiva projeção ocorra em uma hiperesfera. Caso contrário, a superfície deixará de ser esférica, visto que os seus eixos constituintes não são mais ortogonais.

Portanto, após a convergência da estrutura neural, quando uma nova amostra for apresentada às entradas da rede para propósitos de classificação, basta então verificar qual neurônio será o vencedor, pois estará assim indicando a classe em que a amostra possuirá o maior nível de proximidade.

As instruções seguintes descrevem os passos algorítmicos empregados para a fase de operação da estrutura neural com aprendizado competitivo.

#### Início (Algoritmo Competitivo – Fase de Operação)

- <1> Apresentar a amostra ( $x$ ) a ser classificada;
- <2> Normalizar o vetor ( $x$ ) correspondente à amostra;
- <3> Asumir os vetores de pesos  $\{w^{(1)}\}$  já ajustados durante a fase de treinamento;
- <4> Execute as seguintes instruções:
  - <4.1> Calcular as distâncias euclidianas entre  $x$  e  $w^{(i)}$  conforme a expressão (8.1);
  - <4.2> Declarar como vencedor o neurônio / que contenha a menor distância euclidiana;
  - <4.3> Associar a amostra àquela classe que está sendo representada pelo neurônio vencedor;
- <5> Disponibilizar a classe em que a amostra foi associada.

#### Fim (Algoritmo Competitivo – Fase de Operação)

Assim como nas estruturas neurais apresentadas nos capítulos anteriores, não há alteração dos vetores de pesos sinápticos durante a fase de

operação da rede competitiva convencional, pois tais ajustes são somente realizados durante a fase de treinamento.

### 8.3 – Mapas auto-organizáveis de Kohonen (SOM)

Os mapas auto-organizáveis de Kohonen, também denominados de SOM (*self-organizing map*), são considerados uma arquitetura de redes neurais artificiais de estrutura reticulada, com aprendizado competitivo [Kohonen, 1984].

Os mapas auto-organizáveis, em sua essência, são estruturas neurais competitivas como aquela mostrada na figura 8.1. Adicionalmente, as conexões laterais, demonstrando o como a saída do neurônio vencedor influenciará, ou colaborará, com os demais neurônios da rede, são fornecidas por mapas topológicos de vizinhança. Nesta situação, os potenciais dessas conexões laterais serão maiores (excitatórias) para aqueles neurônios que estão mais próximos aos neurônios vencedores, ao passo que estas serão menores à medida que essas distâncias sejam também mais acentuadas.

Os mapas topológicos informam como estarão organizados espacialmente os neurônios da rede frente ao comportamento de seus vizinhos, sendo que normalmente são formados por uma dimensão (*array*) ou duas dimensões (*grid*).

A figura 8.5 mostra um mapa auto-organizável de uma dimensão. Assim como na figura 8.1, o vetor de entradas ( $\mathbf{x}$ ), correspondente às amostras, será apresentado a todos os neurônios do mapa. Aqui, portanto, todos os neurônios estão espacialmente organizados em uma única linha.

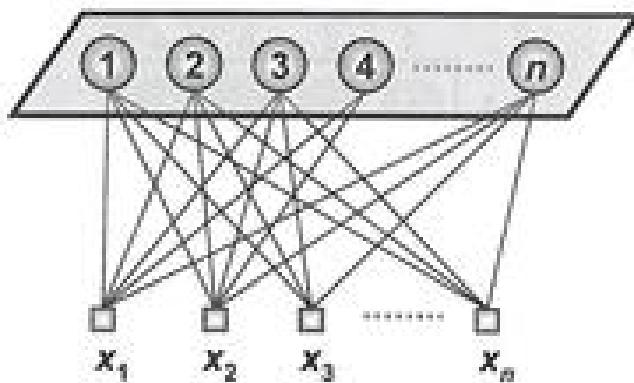


Figura 8.5 – Mapa topológico em formato unidimensional

Uma ilustração de mapa bidimensional, constituído por 16 neurônios, é apresentada na figura 8.6. É importante observar que para tal circunstância os neurônios estão espacialmente arranjados em linhas e colunas.

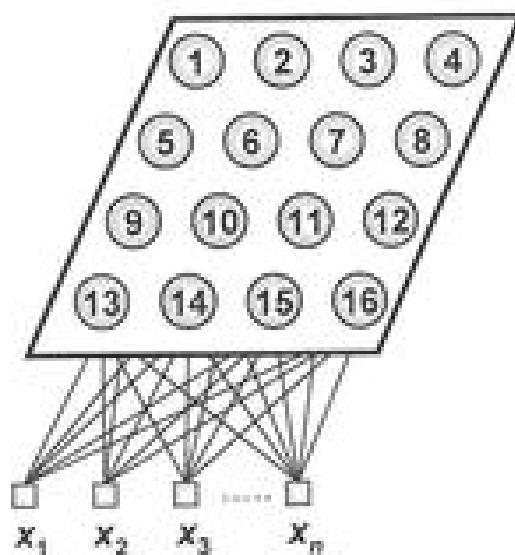


Figura 8.6 – Mapa topológico em formato bidimensional.

A figura 8.7 mostra uma representação matricial do mapa topológico ilustrado na Figura 8.6. Embora o número de linhas e colunas possa ser desigual, nota-se que, para o exemplo apresentado, o mapa é composto de quatro linhas e quatro colunas de neurônios.

	1	2	3	4
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12
4	13	14	15	16

Figura 8.7 – Representação matricial do mapa topológico bidimensional.

Definido o arranjo espacial dos neurônios para o mapa topológico, basta agora especificar o critério de vizinhança interneurônios que indique como estarão cooperando em relação aos seus vizinhos. Um dos principais critérios de vizinhança consiste de especificar um raio  $R$  de abrangência, que será utilizado pelos neurônios da rede visando definir seus respectivos vizinhos.

Portanto, para um determinado neurônio  $j$ , seus vizinhos serão todos aqueles que estarão a uma distância máxima que seja menor ou igual a  $R$ . Como exemplo, considerando o neurônio de número 6 (figura 8.7), tem-se que os seus vizinhos, para um raio de vizinhança igual a 1 (circunferência tracejada mais interna), seriam os neurônios de número 2, 5, 7 e 10. Caso se assumisse um raio de 1,75 (circunferência tracejada mais externa), sua vizinhança seria então composta pelos neurônios 1, 2, 3, 5, 7, 9, 10 e 11.

Para efeitos de ilustração, admitindo-se que uma vizinhança de raio unitário ( $R = 1$ ) seja assumida para todos os neurônios, a figura 8.8 mostra o mapa topológico com as interligações de cada um com os seus vizinhos correspondentes.

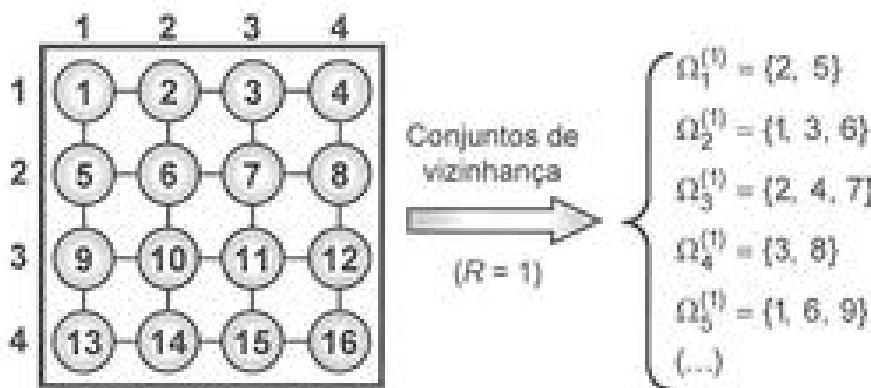


Figura 8.8 – Interligações de vizinhança considerando raio unitário para cada neurônio do mapa auto-organizável

Visando facilitar os procedimentos algorítmicos, define-se os conjuntos vizinhanças dos neurônios do mapa topológico pelo parâmetro  $\Omega_j^{(R)}$ , que indica quais são os neurônios que estarão na vizinhança do neurônio  $j$ , cujo raio de vizinhança seja  $R$ . Na figura 8.8 é apresentado alguns valores dos conjuntos de

vizinhança entre os neurônios, quando se assume um raio unitário ( $R = 1$ ). Caso fosse adotado um raio no valor de 1,75, ter-se-iam os seguintes conjuntos:

$$\Omega_1^{(1,75)} = \{2, 5, 6\}$$

$$\cap \Omega_2^{(1,75)} = \{1, 3, 5, 6, 7\}$$

$$\Omega_1^{(1,75)} = \{2, 4, 6, 7, 8\}$$

$$\cap \Omega_3^{(1,75)} = \{3, 7, 8\}$$

$$\Omega_1^{(1,75)} = \{1, 2, 6, 9, 10\}$$

$$\cap \Omega_4^{(1,75)} = \{1, 2, 3, 5, 7, 9, 10, 11\}$$

(...)

$$\Omega_1^{(1,75)} = \{10, 11, 12, 14, 16\}; \quad \Omega_5^{(1,75)} = \{11, 12, 15\}$$

Assim, considerando que um neurônio  $j$  venceu a competição frente à apresentação de uma amostra, tanto o seu vetor de pesos como os de seus vizinhos serão também ajustados. Todavia, os neurônios que estão na vizinhança do neurônio vencedor serão ajustados com taxas menores aquelas usadas para o ajuste do vetor de pesos do neurônio vencedor. Uma tática razoável, quando da utilização de raio de vizinhança unitário, é a de assumir um ajuste para os vetores de pesos dos neurônios vizinhos ao vencedor igual à metade da taxa de aprendizagem. Nesta condição, a expressão dada em (8.3) passa a ser representada por duas regras:

$$\left\{ \begin{array}{l} \text{Regra 1: } w^{(v)} \leftarrow w^{(v)} + \eta \cdot (x^{(k)} - w^{(v)}), \text{ para o neurônio vencedor} \\ \text{Regra 2: } w^{(0)} \leftarrow w^{(0)} + \frac{\eta}{2} \cdot (x^{(k)} - w^{(0)}), \text{ para os seus vizinhos} \end{array} \right. \quad (8.4)$$

onde a primeira regra é aplicada ao neurônio vencedor, enquanto a segunda aplica-se a todos os neurônios vizinhos ao vencedor, os quais são especificados pelo respectivo conjunto de vizinhança.

Como exemplo da utilização da expressão (8.4), em relação à configuração de mapa auto-organizável da figura 8.8, caso o neurônio de número 10 fosse o vencedor, o seu vetor de pesos seria ajustado pela primeira regra da expressão (8.4), de modo que os vetores de pesos de seus vizinhos {6, 9, 11 e 14} seriam ajustados pela segunda regra.

Para regiões de vizinhança que sejam maiores que uma unidade, a taxa de aprendizagem poderia ser ponderada por uma função do tipo gaussiana, sendo que quanto maiores forem as distâncias dos neurônios vizinhos em relação ao neurônio vencedor, menores então seriam os seus ajustes. Em tal instância, a expressão (8.4) torna-se:

$$\begin{cases} \text{Regra 1: } w^{(v)} \leftarrow w^{(v)} + \eta \cdot (x^{(k)} - w^{(v)}), \text{ para o neurônio vencedor} \\ \text{Regra 2: } w^{(B)} \leftarrow w^{(B)} + \eta \cdot \alpha^{(B)} \cdot (x^{(k)} - w^{(B)}), \text{ para os seus vizinhos} \end{cases} \quad (8.5)$$

onde  $\alpha^{(B)}$  é um operador de vizinhança dado por:

$$\alpha^{(B)} = e^{-\frac{|w^{(v)} - w^{(B)}|^2}{2\sigma^2}} \quad (8.6)$$

Assim sendo, a aplicação do operador gaussiano acaba proporcionando uma taxa de decaimento que fica bem intensificada para aqueles neurônios mais afastados do vencedor, mas que também devem ser contemplados por pertencerem ao conjunto de vizinhança. A figura 8.9 mostra uma ilustração deste decaimento gaussiano.

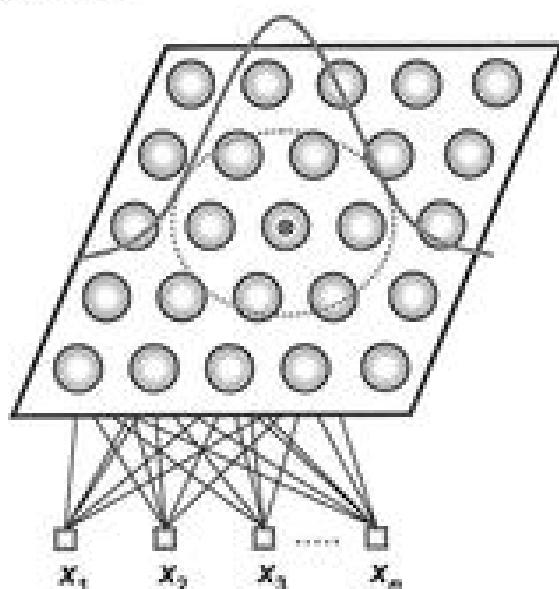


Figura 8.9 – Ilustração de decaimento gaussiano para os pesos dos neurônios vizinhos ao vencedor.

Como exemplo, considerando o arranjo topológico da figura 8.7, com raio de 1,75 (circunferência traçada mais externa), o conjunto vizinhança do neurônio 6 seria dado por  $\Omega_6^{(1,75)} = \{1, 2, 3, 5, 7, 9, 10, 11\}$ . Nesta circunstância, os ajustes dos vetores de pesos dos neurônios 2, 5, 7 e 10 seriam maiores que aqueles dos neurônios 1, 3, 9 e 11, pois estes estão mais afastados do neurônio vencedor.

Adicionalmente, conforme estudos relatados em Ritter *et al.* (1992), uma outra estratégia utilizada, também visando aumentar a eficiência do processo de ajuste dos pesos, consiste em diminuir gradativamente a taxa de aprendizagem no decorrer do treinamento.

As instruções computacionais do algoritmo que expõe a fase de treinamento de mapas auto-organizáveis de Kohonen são apresentadas na sequência.

#### Início (Algoritmo Kohonen – Fase de Treinamento)

- <1> Definir o mapa topológico da rede;
- <2> Montar os conjuntos de vizinhança ( $\Omega^{(l)}$ );
- <3> Iniciar o vetor de pesos de cada neurônio ( $w^{(l)}$ ) considerando os valores das  $n_l$  primeiras amostras de treinamento;
- <4> Obter o conjunto de amostras de treinamento ( $x^{(k)}$ );
- <5> Normalizar os vetores de amostras e de pesos;
- <6> Especificar a taxa de aprendizagem ( $\eta$ );
- <7> Iniciar o contador de número de épocas (época  $\leftarrow 0$ );
- <8> Repetir as instruções:
  - /<8.1> Para todas as amostras de treinamento ( $x^{(k)}$ ), fazer:
    - <8.1.1> Calcular as distâncias euclidianas entre  $x^{(k)}$  e  $w^{(l)}$ , conforme a expressão (8.1);
    - <8.1.2> Declarar como vencedor o neurônio  $j$  que contenha a menor distância euclidiana:  

$$\text{vencedor} = \arg \min_j \|x^{(k)} - w^{(l)}\|$$
    - <8.1.3> Ajustar o vetor de pesos do vencedor, conforme a regra 1 da expressão (8.4);
    - <8.1.4> Ajustar o vetor de pesos dos neurônios vizinhos ao vencedor, definidos em  $\Omega^{(l)}$ , conforme regra 2 da expressão (8.4) ou (8.5);
    - <8.1.5> Normalizar o vetor de pesos que foi ajustado na instrução anterior;
  - <8.2> época  $\leftarrow$  época + 1;
- Até que não haja mudanças significativas nos vetores de pesos;
- <9> Analisar o mapa visando extração de características;
- <10> Identificar regiões que possibilitem a definição de classes.

#### Fim (Algoritmo Kohonen – Fase de Treinamento)

Como os mapas auto-organizáveis não possuem de antemão as saídas (classes) desejadas, os passos <9> e <10> da fase de treinamento podem requerer o uso de ferramentas estatísticas e de conhecimento especialista, a fim de identificar as possíveis classes que agrupam dados com características em comum. Por conseguinte, o mapa topológico pode ser partitionado em regiões que definem as respectivas classes, sendo tal arranjo denominado de mapa de contexto. Como exemplo, a figura 8.10 ilustra um eventual resultado possível de mapa de contexto para a topologia bidimensional mostrada na figura 8.7, confeccionado mediante as análises efetuadas depois da fase de treinamento.



Figura 8.10 – Exemplo de mapa de contexto para uma topologia bidimensional

Para o exemplo considerado nesta figura, observa-se que o mapa de contexto acabou sendo representado por apenas três classes representativas do problema, as quais são constituídas pelos seguintes conjuntos de neurônios:

$$\text{Classe A} \rightarrow \{1, 2, 5, 6, 9, 10\}$$

$$\text{Classe B} \rightarrow \{3, 4, 7, 8\}$$

$$\text{Classe C} \rightarrow \{11, 12, 13, 14, 15, 16\}$$

Desta forma, quando da apresentação de uma nova amostra a ser classificada, basta, em sequência, determinar qual foi o neurônio vencedor da competição. A seguir, recorre-se ao mapa de contexto com a finalidade de

resgatar a classe que este estará incumbido de representar. Finalmente, a amostra será então classificada como pertencente à respectiva classe do neurônio vencedor.

Assim sendo, pode-se ainda deduzir de que os mapas podem ser também utilizados como ferramenta de pré-processamento para outras técnicas de classificação de padrões ou identificação de sistemas.

As instruções seguintes exibem os passos algorítmicos para a fase de operação quando da utilização dos mapas de Kohonen.

#### Início (Algoritmo Kohonen – Fase de Operação)

- {<1> Apresentar a amostra ( $x$ ) a ser classificada e normalizar;
- <2> Assumir os vetores de pesos ( $w^{(j)}$ ) já ajustados durante a fase de treinamento;
- <3> Executar as seguintes instruções:
- <3.1> Calcular as distâncias euclidianas entre  $x$  e  $w^{(j)}$ , conforme a expressão (8.1);
  - <3.2> Declarar como vencedor o neurônio  $j$  que contenha a menor distância euclidiana;
  - <3.3> Localizar o neurônio vencedor dentro do mapa auto-organizável;
  - <3.4> Associar a amostra à classe que foi identificada a partir da confecção do mapa de contexto;
- <4> Disponibilizar a eventual classe em que a amostra foi associada.

#### Fim (Algoritmo Kohonen – Fase de Operação)

Em suma, mediante aquilo que foi apresentado nesta seção, três aspectos devem ser levados em conta para a configuração de um mapa auto-organizável de Kohonen, ou seja:

- Definição da organização espacial (*grid*) dos neurônios;
- Delimitação dos conjuntos de vizinhança de cada neurônio;
- Especificação do critério de ajuste do vetor de pesos do neurônio vencedor e de seus vizinhos.

Finalmente, toda a dinâmica envolvida com o processo de treinamento da rede de Kohonen pode ser ainda sistematizada pelos seguintes passos:

- 1) Cada neurônio da rede computa o nível de proximidade de seu vetor de pesos em relação a cada padrão de entrada;
- 2) Um mecanismo de competição entre os neurônios é aplicado com o objetivo de escolher o vencedor;
- 3) A partir da definição do neurônio vencedor, resgata-se, por meio do mapa topológico, o conjunto de vizinhança que informa sobre quais são os neurônios vizinhos ao vencedor;
- 4) Os pesos do neurônio vencedor e de seus vizinhos são incrementados com o objetivo de aumentar o nível de proximidade com a respectiva entrada;
- 5) Após a convergência, é possível identificar regiões do mapa (contexto) que correspondem às classes do problema.

A sua estrutura de configuração simples, assim como sua dinâmica de treinamento diferenciada, faz dos mapas auto-organizáveis de Kohonen uma ferramenta sofisticada para aplicações em problemas que envolvem classificação de padrões e identificação de agrupamentos de dados (*clustering*) com características próprias.

Conforme pode também ser observado, o algoritmo de aprendizado competitivo acaba tendo algumas similaridades com o algoritmo *k-máx*, idealizado por Steinhaus (1956), em que as amostras disponíveis devem ser divididas em  $k$  grupos (representados pelos seus centros), tendo-se como critério de alocação a norma euclidiana entre a amostra e os referidos centros.

Todavia, em conformidade com o descrito na subseção 6.2.1, uma das limitações do método *k-máx* é a necessidade de se especificar *a priori* qual a quantidade de grupos a ser considerada. Em contraste, o algoritmo de aprendizado competitivo implementado nos mapas organizáveis se configura como um diferencial, pois facilita a identificação do número de classes após a sua convergência.

#### 8.4 - Exercícios

- 1) Explique por que é recomendável a normalização, tanto dos vetores

res de pesos como dos vetores de amostras, quando da aplicação do aprendizado competitivo.

2) Discorra sobre as eventuais dificuldades que se podem enfrentar, durante o decorrer do processo de convergência da rede competitiva, quando se opta pela não-normalização dos vetores de pesos e de amostras.

3) Qual o significado do parâmetro "taxa de aprendizagem" durante a fase de treinamento da rede competitiva. Discorra sobre as implicações de se assumir valores bem pequenos, assim como valores bem altos, para este parâmetro.

4) Considerando a expressão (8.2), discorra passo a passo (de forma geométrica) como se processa o ajuste dos pesos do neurônio vencedor.

5) Para o mapa topológico ilustrado na figura 8.7, especifique os conjuntos de vizinhança de todos os neurônios, cujo raio de vizinhança seja igual a 2 ( $R = 2$ ).

6) Qual o significado das conexões laterais entre os neurônios quando da utilização de um mapa auto-organizável de Kohonen.

7) Explique quais são as principais diferenças existentes entre a rede neural competitiva, apresentada na figura 8.1, em relação à rede SOM.

8) Em relação aos diagramas das figuras 8.3 e 8.4, discorra sobre o que poderia acontecer nessas distribuições no caso de ser especificada uma quantidade de neurônios que fosse inferior aos respectivos números de classes.

9) Explique por que a concepção da rede SOM pode ser associada a algumas características estruturais do córtex cerebral.

10) Considerando uma amostra específica  $x^{(k)}$ , demonstre que a regra de ajuste do vetor de pesos, baseada na norma euclidiana (expressão 8.2), é obtida a partir da minimização da função erro quadrático definida por:

$$E_{w0} = \frac{1}{2} \sum_{i=1}^n (w_i^{(k)} - x_i^{(k)})^2$$

onde  $w^{(k)}$  é o neurônio vencedor e  $n$  é quantidade de sinais de entrada que compõe a amostra.

## 8.5 – Projeto prático

No processo industrial de fabricação de pneus sabe-se que o compo-

to que forma a borracha pode apresentar imperfeições que impedem a sua utilização. Diversas amostras dessas anomalias foram coletadas, sendo que também foram realizadas as medidas referentes a três grandezas  $\{x_1, x_2, x_3\}$ , as quais fazem parte do processo de fabricação das respectivas borrachas. Contudo, a equipe de engenheiros e cientistas não tem percepção técnica de como essas variáveis podem estar relacionadas.

Assim, pretende-se aplicar uma rede de Kohonen, constituída de 16 neurônios (figura 8.11), com o objetivo de detectar as eventuais similaridades e correlações existentes entre essas variáveis, pois se tem como objetivo final o posterior agrupamento das amostras imperfeitas em classes.

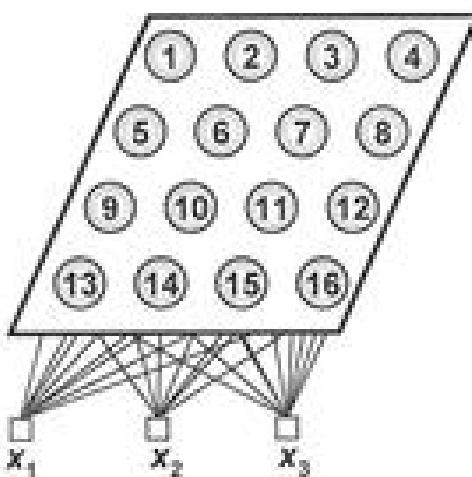


Figura 8.11 – Estrutura espacial do mapa auto-organizável do projeto prévio.

Portanto, baseado nos dados fornecidos no apêndice V, treine um mapa auto-organizável de Kohonen, conforme a estrutura espacial apresentada na figura 8.11, utilizando taxa de aprendizagem no valor de 0,001.

O diagrama esquemático da grade bidimensional ilustrando as vizinhanças interneurônicos, assumindo-se aqui um raio de vizinhança unitário ( $R = 1$ ), é apresentado na figura 8.12.

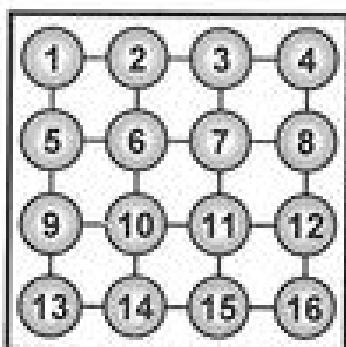


Figura 8.12 – Arranjo de vizinhança entre os neurônios ( $R = 1$ )

De posse dos resultados advindos do treinamento da rede, efetuou-se uma análise neste conjunto e verificou-se que as amostras de 1 a 20, de 21 a 60 e aquelas de 61 a 120 possuem particularidades em comum, podendo ser então consideradas três classes distintas, denominadas de classe *A*, *B* e *C*, respectivamente.

Por conseguinte, cabem os seguintes exercícios:

- 1) Realize a identificação sobre quais os conjuntos de neurônios representados na grade da figura 8.12 fornecem respostas relativas às classes *A*, *B* e *C*.
- 2) Para as amostras da tabela 8.1, indique a classe à qual cada uma pertence.

Tabela 8.1 – Amostras para identificação de classes pelo mapa auto-organizável

Amostra	$x_1$	$x_2$	$x_3$	Classe
1	0,2471	0,1778	0,2905	
2	0,8240	0,2223	0,7041	
3	0,4960	0,7231	0,5866	
4	0,2923	0,2041	0,2234	
5	0,8118	0,2668	0,7484	
6	0,4837	0,8200	0,4792	
7	0,3248	0,2629	0,2375	
8	0,7209	0,2116	0,7821	
9	0,5259	0,6522	0,5957	
10	0,2075	0,1689	0,1745	
11	0,7830	0,3171	0,7888	
12	0,5393	0,7510	0,5682	



Hugo Steinhaus



## Redes LVQ e counter-propagation

### 9.1 – Introdução

Neste capítulo serão apresentadas mais duas arquiteturas de redes neurais artificiais com treinamento supervisionado. Uma delas, a rede *LVQ* (*learning vector quantization, quantização vetorial por aprendizado*), possui certa similaridade em seu processo de aprendizado quando comparada com a rede auto-organizável de Kohonen, sendo tipicamente aplicada em problemas de classificação de padrões. O treinamento dessa rede neural, composta em sua forma convencional por uma única camada de neurônios, realiza um processo de quantização vetorial, frente ao espaço onde as amostras se encontram, a fim de ponderar as regiões de domínio de cada uma das classes. Uma apresentação detalhada do processo de quantização vetorial será desenvolvida na próxima seção.

A outra arquitetura de redes neurais artificiais, que também será analisada nesta unidade, é denominada de rede *counter-propagation* (contra-propagação), constituída basicamente de duas camadas neurais, cujo treinamento é uma combinação de aprendizado auto-organizado (utilizado na primeira camada) com aprendizado supervisionado (aplicado na segunda camada). Entretanto, a rede *counter-propagation* é classificada como sendo uma arquitetura cujo treinamento é supervisionado, visto que necessita *a priori* de um

conjunto de pares desejados de entradas e saídas a fim de ajustar os seus parâmetros internos.

Diferentemente da rede *LJ/Q*, a rede *counter-propagation* pode ser utilizada tanto em extração de características visando classificação de padrões como em problemas relacionados à aproximação de funções e processamento de sinais, tais como nas aplicações apresentadas em Chang & Su (2005), Morns & Dlay (2003), Yang *et alii* (2004) e Lin *et alii* (1990).

## 9.2 – Processo de quantização vetorial

Considerando um problema de classificação de padrões, cujas amostras são divididas em  $n$  classes, o processo de quantização vetorial consiste em atribuir a cada uma, um único vetor, denominado de quantizador (vetor referência), que possa representar o perfil que permeia o respectivo grupo frente às operações de classificação de uma nova amostra. Assim sendo, esta nova amostra será classificada como pertencente àquela classe em que o vetor quantizador esteja mais próximo.

Neste caso, levando em conta uma classe específica, o objetivo da quantização vetorial seria a obtenção de um quantizador, que esteja alocado numa posição estratégica, cuja soma de sua distância em relação a todas as amostras que compõem a classe seja a menor possível. A figura 9.1 ilustra a ideia envolvida com o processo de quantização vetorial frente a um conjunto de amostras constituídas de duas componentes  $\{x_i, e x_j\}$ .

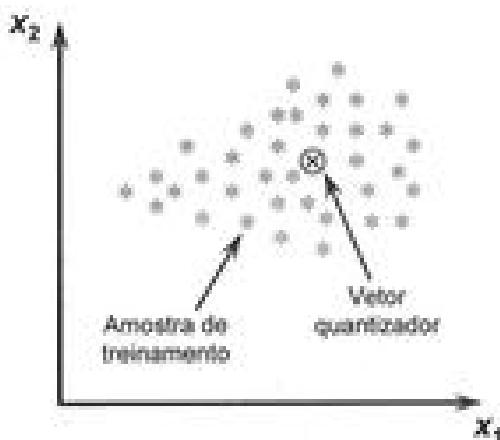


Figura 9.1 – Ilustração de um vetor quantizador de classe

Observa-se na figura 9.1 que o vetor quantizador ficou posicionado numa localização em que o somatório de sua distância aos elementos da classe é o mínimo possível. Na situação ilustrada, verifica-se ainda que o quantizador está alocado mais proximamente ao local onde há o maior adensamento de amostras. Por outro lado, se as amostras da classe tiverem uma distribuição normal, o vetor quantizador tenderia então à média de seus valores.

Considerando agora a existência de um total de  $n$  classes, haverá então  $n$  vetores quantizadores que serão responsáveis por representar os perfis discriminantes de cada uma. A figura 9.2 ilustra cinco classes distintas com seus respectivos vetores quantizadores já previamente calculados.

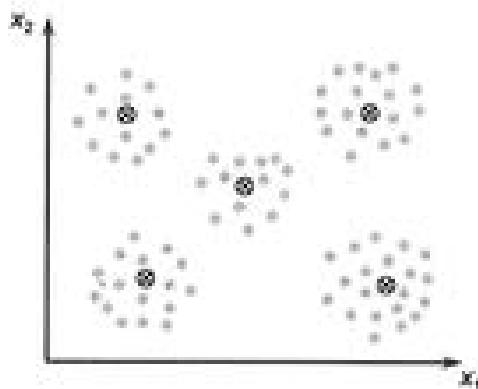


Figura 9.2 – Ilustração de vetores quantizadores

Assim, cada um dos vetores quantizadores, ilustrados na figura 9.2, estará representando o perfil característico que domina cada uma das classes que estes estão representando.

Considerando agora a situação de que uma nova amostra tenha que ser classificada, dentre uma das cinco classes definidas pela figura 9.2, o único parâmetro que deve ser então usado para a referida tarefa será o próprio vetor quantizador representativo de cada classe. Deste modo, a principal medida utilizada para tal propósito é o nível de proximidade, sendo que o vetor quantizador mais próximo à amostra é que diretamente determinará a classe em que deve ser alocada. De fato, quanto mais próximo esteja um vetor quantizador de determinada amostra, mais parecido estará.

Consequentemente, levando-se em conta este critério de distância mínima para definir o vetor quantizador dominante à determinada amostra, tornar-se-á

então possível delimitar as fronteiras de dominação frente aos demais quantizadores. Para tanto, dado que o ponto médio entre dois quaisquer vetores quantizadores seria um divisor natural de proximidade, concluir-se-ia que bastaria obter o hiperplano perpendicular ao segmento de reta que liga tais vetores, impondo-se como condição a sua passagem pelo respectivo ponto médio. Como ilustração, a figura 9.3 mostra as fronteiras de dominância de classe associadas a cada um dos vetores quantizadores definidos pela figura 9.2.

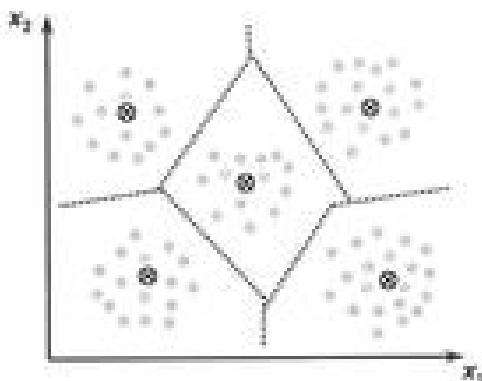


Figura 9.3 – Ilustração de regiões de dominação dos vetores quantizadores

Observa-se que as fronteiras de dominância, quando se assume o critério de proximidade, possibilitam decompor todo o espaço amostral métrico em sub-regiões que delimitam as referidas classes. O conjunto total destes reticulados derivados do referido particionamento, e que permite também a redução da dimensionalidade do problema, é conhecido como diagrama de Voronoy [Aurenhammer, 1991] ou mosaico de Dirichlet [Bowyer, 1981].

Em termos práticos, após a alocação dos vetores quantizadores, a definição da classe a que pertence uma nova amostra é realizada a partir do cálculo de sua distância em relação a todos os vetores quantizadores. A amostra será então classificada como pertencente àquela classe que estará sendo integralmente representada pelo vetor quantizador com menor distância.

Desta forma, a principal tarefa da quantização vetorial está em determinar a posição espacial de cada quantizador dentro do conjunto amostral de cada classe, sendo também este o alvo das redes  $LVQ$ .

Em termos comparativos, a quantização vetorial é bem diferente da clusterização, cujo propósito está apenas em encontrar e agrupar as amostras em

classes que possuem qualidades em comum. Em contraste, além de encontrar grupos com características similares, a quantização vetorial consiste principalmente em converter todo o espaço de entradas, onde as amostras estão definidas, para um espaço discreto de saída, sendo que, nesta condição, a própria saída se torna uma representação discreta daquele espaço de entradas, havendo ainda uma considerável redução de sua dimensionabilidade.

### 9.3 – Redes LVQ (*learning vector quantization*)

A rede LVQ foi também idealizada por Teuvo Kohonen [Kohonen, 1990], sendo considerada uma versão supervisionada dos mapas auto-organizáveis (*self-organizing maps – SOM*). Assim, diferentemente da topologia SOM, exige-se então, para propósitos de treinamento das redes LVQ, um conjunto de pares de entradas e saídas representativos do processo a ser mapeado.

O treinamento de redes LVQ é também executado de maneira competitiva, similarmente àquelas utilizados para as redes SOM, de modo que os vetores de pesos dos neurônios estarão representando os respectivos vetores quantizadores de classes, conforme ilustrado na seção 9.2. Assim, para a utilização desta topologia, as diversas classes associadas à representação do processo devem ser conhecidas.

A figura 9.4 mostra uma rede LVQ composta de  $n$  entradas e  $n_t$  neurônios, os quais estarão representando todas as classes envolvidas com o referido problema de classificação de padrões a ser mapeado.

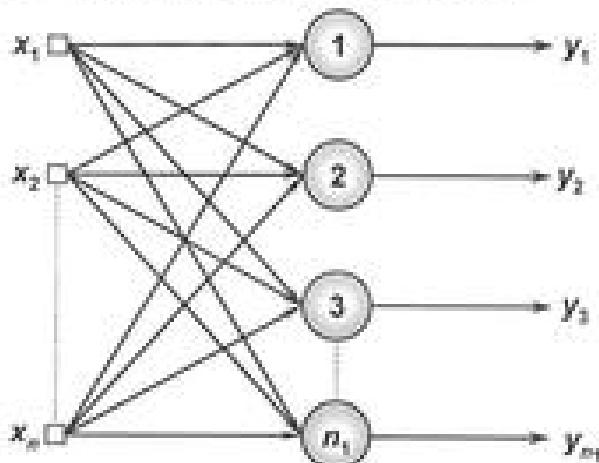


Figura 9.4 – Estrutura neural básica da rede LVQ

Conforme exibido na figura 9.4, em contraste à arquitetura de Kohonen, a rede LVQ não possui conexões laterais entre neurônios, sendo que este aspecto implica que aqueles neurônios vizinhos ao vencedor deixarão também de ter os seus pesos ajustados. De fato, a existência de conexões laterais é desnecessária, pois os estímulos excitatório e inibitório serão computados pelo algoritmo de treinamento.

Dois algoritmos de treinamentos das LVQ's, denominados de LVQ-1 e LVQ-2, são normalmente utilizados para ajuste dos pesos dos neurônios vencedores. O algoritmo LVQ-1 ajusta os pesos somente do neurônio vencedor, enquanto que o algoritmo LVQ-2 ajusta os pesos tanto do vencedor como do vice. Outros procedimentos de ajuste mais elaborados, inspirados nesses algoritmos, têm também sido propostos na literatura visando otimizar o processo frente a algumas aplicações, tais como aqueles apresentados em Lee & Kim (1995) e Vakil-Baghmash & Pavesic (2003).

### 9.3.1 – Algoritmo de treinamento LVQ-1

Os passos envolvidos com o algoritmo de treinamento LVQ-1 são implementados de maneira a ajustar somente os pesos sinápticos dos neurônios vencedores, frente aos processos de competição que envolvem cada uma das amostras disponíveis.

Neste caso, considera-se que cada vetor de entrada  $\{x^i\}$ , utilizado durante o treinamento da LVQ, pertence somente a uma das classes  $j$  previamente conhecidas, pois o mecanismo de aprendizado é do tipo supervisionado.

Os procedimentos inerentes ao algoritmo de aprendizagem são, portanto, idênticos àqueles da rede de Kohonen, modificando apenas o critério de seleção dos neurônios que terão os seus pesos ajustados, sendo que no algoritmo de treinamento LVQ-1 somente os pesos do neurônio vencedor serão devidamente sintonizados.

O dois passos principais do algoritmo consistem da obtenção do neurônio vencedor, assim como da especificação da regra de ajuste de seus respectivos pesos sinápticos.

Em relação à obtenção do vencedor, aquele neurônio cujo vetor de pesos  $\{w^j\}$  tiver o maior nível de proximidade com uma determinada amostra  $\{x^i\}$ , será declarado vitorioso. Similarmente à rede de Kohonen, uma das

medidas de proximidade mais utilizadas é a norma euclidiana entre esses dois parâmetros, ou seja:

$$\text{dist}_j^{(k)} = \sqrt{\sum_{i=1}^n (x_i^{(k)} - w_j^{(j)})^2}, \text{ onde } j = 1, \dots, n_j \quad (9.1)$$

onde  $\text{dist}_j^{(k)}$  fornece a distância entre o vetor de entrada representando a  $k$ -ésima amostra  $\{x^{(k)}\}$  em relação ao vetor de pesos do  $j$ -ésimo neurônio  $\{w^{(j)}\}$ .

Após a declaração do neurônio vencedor, aplica-se então a regra de ajuste de seus pesos. Quando a situação do neurônio vencedor  $\{w^{(j)}\}$  estiver representando a própria classe atribuída à respectiva amostra  $\{x^{(k)}\}$ , isto é,  $x^{(k)} \in C^j$ , ajustam-se então os seus pesos com a finalidade de aproximar-lo ainda mais daquela amostra, pois, nesta circunstância, o mesmo terá como recompensa uma grande chance de vencer a competição quando a referida amostra for novamente apresentada no decorrer do treinamento. Caso contrário, quando a condição do neurônio vencedor  $\{w^{(j)}\}$  não estiver representando a classe da referida amostra  $\{x^{(k)}\}$ , ou seja,  $x^{(k)} \notin C^j$ , então os seus pesos serão ajustados com o intuito de se afastar, possibilitando, consequentemente, a chance de que outro neurônio possa vencer a competição na próxima época de treinamento. Em termos algorítmicos, tais procedimentos podem ser sintetizados por meio da seguinte regra:

Se  $x^{(k)} \in C^j$

$$\text{Então: } w^{(j)} \leftarrow w^{(j)} + \eta \cdot (x^{(k)} - w^{(j)}) \quad (9.2)$$

$$\text{Senão: } w^{(j)} \leftarrow w^{(j)} - \eta \cdot (x^{(k)} - w^{(j)})$$

onde o parâmetro  $\eta$  define a taxa de aprendizagem.

A figura seguinte (9.5) ilustra os mecanismos de ajustes dos pesos do neurônio declarado como vencedor, quando da aplicação do algoritmo de treinamento LVCQ-1, onde se considera que uma amostra  $\{x^{(k)}\}$  seja constituída de duas componentes  $\{x_1$  e  $x_2\}$ , sendo que esta poderá pertencer a uma das duas classes  $\{C^0\}$  ou  $\{C^1\}$  existente ao problema.

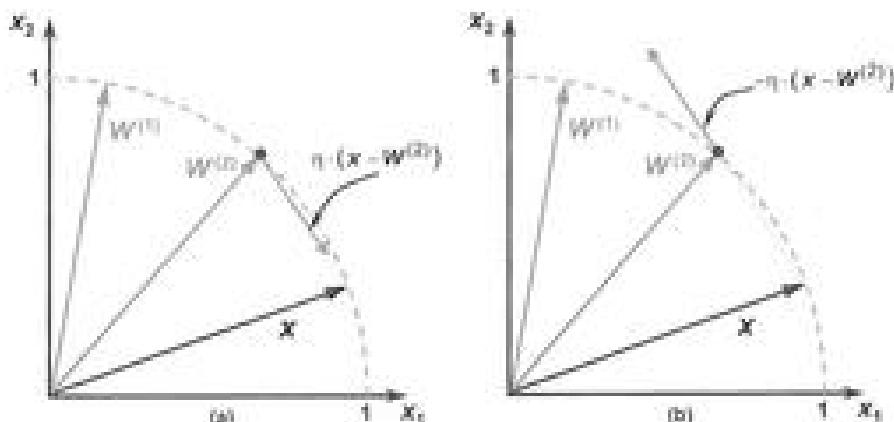


Figura 9.5 – Mecanismo de ajuste de pesos do algoritmo L1/Q-1

Assim, assumindo a premissa de que a referida amostra pertence à classe tipo 2, ou seja,  $x^0 \in C^2$ , então o vetor de pesos do neurônio 2  $\{w^{(2)}\}$ , designado para representar a referida classe, será movido em direção ao vetor  $x^0$ , conforme mostra a figura 9.5a. Caso contrário, de acordo com a figura 9.5b, na situação de que a amostra  $x^0 \notin C^2$ , infere-se então que o vetor de pesos do neurônio vencedor  $\{w^{(2)}\}$  deverá se afastar de  $x^0$ , a fim de que o outro neurônio possa vencer a prova de proximidade no decorrer das próximas iterações.

Em suma, conclui-se que, se o neurônio vencedor  $w^{(2)}$  estiver representando a classe à qual a amostra  $x^0$  pertence, então este será atraído em direção à  $x^0$ ; senão, o vetor  $w^{(2)}$  será repelido, dando assim chance para que o neurônio que esteja representando a referida classe possa vencer nas etapas posteriores.

Os passos básicos do algoritmo L1/Q-1 (fase de treinamento) são exibidos na sequência, por meio de instruções em pseudocódigo.

**Inicio {Algoritmo LVQ-1 II Fase de Treinamento}**

- <1> Obter o conjunto de amostras de treinamento { $x^{(k)}$ };
- <2> Associar cada amostra ( $x^{(k)}$ ) com sua respectiva classe (saída) desejada ( $C^{(j)}$ ), com  $j$  variando de um até o total de classes;
- <3> Iniciar o vetor de pesos de cada neurônio ( $w^{(j)}$ ) com um dos valores das amostras em { $C^{(j)}$ }.
- <4> Normalizar os vetores de amostras { $x^{(k)}$ } e de pesos { $w^{(j)}$ };
- <5> Especificar a taxa de aprendizagem ( $\eta$ );
- <6> Iniciar o contador de número de épocas { $\text{época} \leftarrow 0$ };
- <7> Repetir as instruções:
- {<7.1> Para todas as amostras de treinamento ( $x^{(k)}$ ), fazer:
  - <7.1.1> Calcular as distâncias euclidianas entre  $x^{(k)}$  e  $w^{(j)}$ , conforme a expressão (9.1);
  - <7.1.2> Declarar como vencedor aquele neurônio  $j$  que contenha a menor distância euclidiana:  
 $vencedor = \arg \min_j \|x^{(k)} - w^{(j)}\|$ ;
  - <7.1.3> Ajustar o vetor de pesos do vencedor, conforme a regra explicitada na expressão (9.2);
  - <7.1.4> Normalizar o vetor de pesos que foi ajustado no passo anterior;
- <7.2>  $\text{época} \leftarrow \text{época} + 1$ .

Até que: não haja mudanças significativas nos vetores de pesos.

**Fim {Algoritmo LVQ-1 II Fase de Treinamento}**

A fase de operação da rede, quando da apresentação de uma nova amostra, sendo aplicada após a fase de treinamento, consiste em determinar tão somente o neurônio vencedor, cujo rótulo estará representando a classe a ser atribuída à respectiva amostra. O algoritmo seguinte fornece as instruções básicas da fase de operação.

**Início { Algoritmo LVQ-1 – Fase de Operação}**

- { <1> Apresentar a amostra ( $x$ ) a ser classificada e normalizar;  
 <2> Asumir os vetores de pesos ( $w^{(l)}$ ) já ajustados durante a fase de treinamento;  
 <3> Executar as seguintes instruções:  
     <3.1> Calcular as distâncias euclidianas entre  $x$  e  $w^{(l)}$ , conforme a expressão (9.1);  
     <3.2> Declarar como vencedor o neurônio  $j$  que contenha a menor distância euclidiana;  
     <3.3> Associar a amostra à classe em que o neurônio vencedor estiver representando;  
 <4> Disponibilizar a eventual classe em que a amostra foi associada.

**Fim {Algoritmo LVQ-1 – Fase de Operação}**

Consoante ao mencionado anteriormente, a partir da inspeção das instruções do algoritmo LVQ-1, observa-se sua similaridade ao algoritmo de Kohonen, sendo que uma das principais diferenças está justamente no fato de se conhecer antecipadamente a classe em que cada amostra está atribuída, viabilizando-se assim a aplicação deste método de treinamento supervisionado.

**9.3.2 – Algoritmo de treinamento LVQ-2**

Em relação ao algoritmo de treinamento LVQ-2, os procedimentos de ajuste são agora aplicados tanto para o neurônio vencedor como para o seu vice.

Assim, considera-se que o vetor de pesos do neurônio vencedor seja definido por  $w^v$  e que o vetor de pesos do vice-vencedor seja dado por  $w^w$ . Neste caso, assim como ocorre para o algoritmo LVQ-1, para a situação do neurônio vencedor ( $w^v$ ) estar representando a própria classe atribuída à respectiva amostra ( $x^v$ ), isto é,  $x^v \in C^v$ , os seus pesos serão então ajustados com o propósito de se aproximar ainda mais daquela amostra. Contrariamente, atuando-se de forma antecipada, o vetor de pesos do vice-vencedor será repelido com a finalidade de se afastar de  $x^v$ , pois o mesmo estará representando outra classe, que é diferente daquela assinalada a  $x^v$ , ou seja,  $x^w \notin C^v$ . Em termos algorítmicos, tem-se a seguinte regra:

**Se ( $x^{(k)} \in C^{(l)}$ ) AND ( $x^{(k)} \in C^{(m)}$ )**

$$\text{Então: } \begin{cases} w^{(l)} \leftarrow w^{(l)} + \eta_j \cdot (x^{(k)} - w^{(l)}) \\ w^{(m)} \leftarrow w^{(m)} - \eta_j \cdot (x^{(k)} - w^{(m)}) \end{cases} \quad (9.3)$$

$$\text{Sendo: } \begin{cases} w^{(l)} \leftarrow w^{(l)} - \eta_j \cdot (x^{(k)} - w^{(l)}) \end{cases}$$

Alternativamente, se o neurônio vice-vencedor estiver representando a referida classe em que pertence a amostra  $\{x^{(k)}\}$ , basta então alterar a regra de ajuste, a qual visa agora tanto aproximar o vice-vencedor como também afastar o vencedor, ou seja:

$$\begin{aligned} \text{Se } & (x^{(k)} \in C^{(l)}) \text{ AND } (x^{(k)} \in C^{(m)}) \\ \text{Então: } & \begin{cases} w^{(l)} \leftarrow w^{(l)} - \eta \cdot (x^{(k)} - w^{(l)}) \\ w^{(m)} \leftarrow w^{(m)} + \eta \cdot (x^{(k)} - w^{(m)}) \end{cases} \\ \text{Senão: } & w^{(l)} \leftarrow w^{(l)} - \eta \cdot (x^{(k)} - w^{(l)}) \end{aligned} \quad (9.4)$$

Outro aperfeiçoamento que pode ainda ser inserido no algoritmo *LVQ-2*, denominado algoritmo *LVQ-2.1* [Kohonen, 1990], consiste em aplicar as regras anteriores somente para o caso da amostra  $x^{(k)}$  estar contida dentro de uma janela de pertinência  $\lambda$ , definida em torno da fronteira entre  $w^k$  e  $w^{(m)}$ , sendo a mesma dada por:

$$\min\left(\frac{d_l}{d_m}, \frac{d_m}{d_l}\right) > \frac{1-\lambda}{1+\lambda}, \text{ onde } \lambda \in [0.2; 0.3] \quad (9.5)$$

onde  $d_l$  e  $d_m$  são respectivamente as distâncias entre  $x^{(k)}$  em relação a  $w^k$  e  $w^{(m)}$ . Portanto, essa variante do algoritmo *LVQ-2* procura então deslocar os vetores  $w^k$  e  $w^{(m)}$  para a vizinhança das fronteiras que separaram as classes, pois essa pode ser considerada como aquela que minimiza o erro da classificação (fronteiras de Bayes). A imposição desta condição deixa a rede bem mais robusta para os propósitos de classificar padrões, após o seu devido treinamento. Além disso, eventuais instabilidades de convergência observadas durante a fase de treinamento é também minimizada.

A sequência de instruções relacionadas ao algoritmo *LVQ-2* são apresentados em seguida.

**Inicio (Algoritmo LVQ-2 // Fase de Treinamento)**

- <1> Obter o conjunto de amostras de treinamento ( $x^{(k)}$ );
  - <2> Associar cada amostra ( $x^{(k)}$ ) com sua respectiva classe (saída) desejada ( $C^{(j)}$ ), com  $j$  variando de um até o total de classes;
  - <3> Iniciar o vetor de pesos de cada neurônio ( $w^{(j)}$ ) com um dos valores das amostras em ( $C^{(j)}$ );
  - <4> Normalizar os vetores de amostras ( $x^{(k)}$ ) e de pesos ( $w^{(j)}$ );
  - <5> Especificar a taxa de aprendizagem ( $\eta$ );
  - <6> Iniciar o contador de número de épocas { época  $\leftarrow 0$  };
  - <7> Repetir as instruções:
  - <7.1> Para todas as amostras de treinamento ( $x^{(k)}$ ), fazer:
    - <7.1.1> Calcular as distâncias euclidianas entre  $x^{(k)}$  e  $w^{(j)}$ , conforme a expressão (9.1);
    - <7.1.2> Obter o neurônio vencedor e o vice-vencedor;
    - <7.1.3> Ajustar o vetor de pesos do vencedor e do vice-vencedor, conforme as regras explicitadas em (9.3) e (9.4);
    - <7.1.4> Normalizar os vetores de pesos dos neurônios que foram ajustados no passo anterior;
  - <7.2> época  $\leftarrow$  época + 1;
- Até que não haja mudanças significativas nos vetores de pesos.

**Fim (Algoritmo LVQ-2 // Fase de Treinamento)**

As instruções da fase de operação associada ao algoritmo *LVQ-2* são aquelas mesmas que foram utilizadas para a *LVQ-1*.

Complementarmente, assim como verificado para o algoritmo de treinamento da rede de Kohonen, as taxas de aprendizagem { $\eta$ } nos algoritmos *LVQ-1* e *LVQ-2* podem ser também decrementadas à medida que o processo de treinamento avança, evitando-se eventuais instabilidades de convergência.

Além dos algoritmos *LVQ-1* e *LVQ-2*, há ainda a proposição do algoritmo *LVQ-3* [Kohonen, 1997], em que são formulados alguns procedimentos adicionais, cuja finalidade é a melhoria no processo de convergência dos vetores quantizadores em relação às classes associadas ao problema a ser mapeado.

#### **9.4 – Redes counter-propagation**

As redes com arquitetura *counter-propagation*, idealizada por Hecht-Nielsen (1987a), são normalmente constituídas de duas camadas neurais, cujo treina-

mento pode ser também considerado híbrido, tal como acontece com as redes RBF. A figura 9.6 exibe uma rede *counter-propagation* constituída de  $n$  entradas,  $n$ , neurônios na camada intermediária e  $n_2$  neurônios em sua camada de saída.

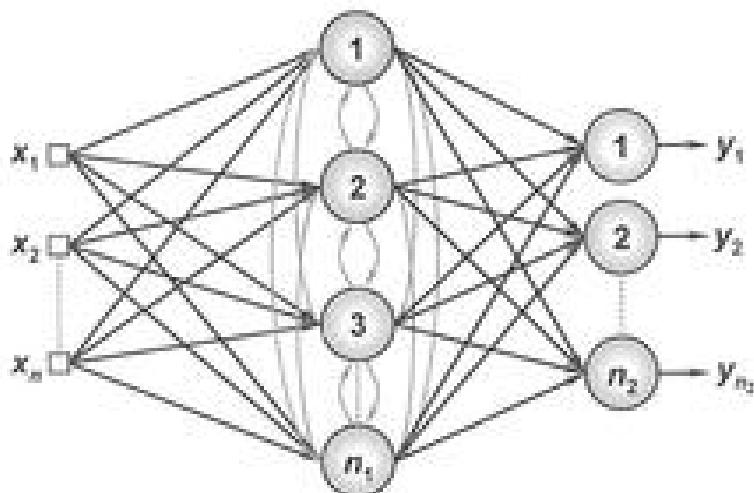


Figura 9.6 – Arquitetura básica da rede *counter-propagation*

Como o seu próprio nome tem sugerido (contra-propagação) e diferentemente das redes PCC, a rede *counter-propagation* não faz retropropagação de erro com o intuito de ajustar os pesos dos neurônios da camada intermediária. Em conformidade com o arranjo estrutural mostrado na figura 9.6, a primeira camada da rede *counter-propagation* é tipicamente um arranjo de Kohonen, sendo que seu treinamento ocorre de forma similar ao algoritmo de aprendizado competitivo apresentado na seção 8.2.

Em relação à segunda camada neural (camada de saída), os seus neurônios constituem um arranjo de Grossberg, denominado também de estrutura *Oscitar* [Grossberg, 1974], com função de ativação rampa e ausência de limiar. O treinamento desta camada se processa de forma supervisionada, aparentemente similar à regra de Hebb, conforme explicitado na subseção seguinte.

Assim sendo, o treinamento da rede *counter-propagation* é realizado em dois estágios. Primeiramente, a camada escondida é ajustada a fim de realizar a identificação dos agrupamentos de dados. Em seguida, a camada de saída (*Oscitar*) é treinada com o intuito de associar as respostas da rede com as respectivas saídas desejadas. Para tanto, as entradas da cama-

da de saída serão os valores produzidos pela própria camada escondida (Kohonen), em que haverá apenas um neurônio vencedor em relação a um estímulo de entrada.

#### 9.4.1 – Aspectos da camada *Outstar*

O treinamento da camada *Outstar* da rede *counter-propagation* deve ser realizado somente após a finalização do aprendizado da camada de Kohonen.

Os procedimentos envolvidos com o ajuste dos pesos dos neurônios da camada *Outstar* são relativamente bem compactos, sendo que somente serão ajustados os pesos daqueles cujas entradas (advindas da camada de Kohonen) sejam diferentes de zero. Para fins de ilustração, considerando a figura 9.6, assume-se que o neurônio 2 da camada de Kohonen seja o vencedor em relação a um estímulo de entrada ( $x$ ). Nesta circunstância, a figura 9.7 ilustra aqueles pesos (setas em linha contínua) a serem ajustados em relação à camada *Outstar*.

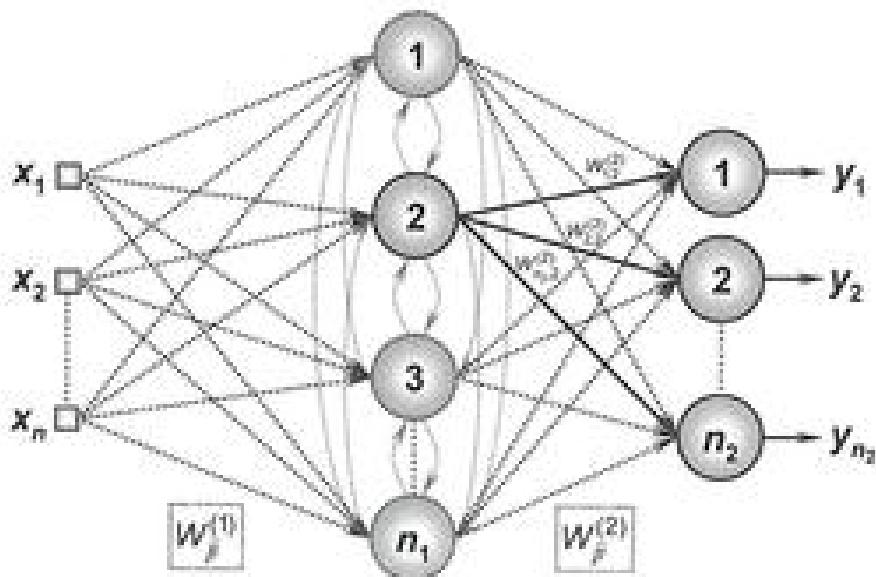


Figura 9.7 – Ilustração do processo de ajuste da camada *Outstar*

A regra de ajuste a ser aplicada a cada um desses pesos é sintetizada pela seguinte expressão:

$$W_j^{(2)} \leftarrow W_j^{(2)} + \eta \cdot (d_j^{(k)} - W_j^{(2)}) \cdot u_j^{(k)} \quad (9.6)$$

onde  $u_j^{(k)}$  é o valor de saída do  $j$ -ésimo neurônio da camada de Kohonen que venceu a competição (em relação à  $k$ -ésima amostra de treinamento),  $d_j^{(k)}$  é a  $j$ -ésima componente do vetor de saídas desejadas (em relação à  $k$ -ésima amostra de treinamento), e o parâmetro  $\eta$  é a taxa de aprendizagem, que tem a finalidade de guiar o processo de convergência da rede.

Para propósitos de eficiência de convergência, sugere-se também que a taxa de aprendizagem  $\{\eta\}$  seja gradualmente decrementada ao longo do processo de treinamento.

Percebe-se, ainda, que a regra de ajuste apresentada em (9.6) simplesmente realiza a atualização dos pesos dos neurônios de referência, de forma proporcional às diferenças existentes entre as suas respectivas componentes de peso frente às componentes de saída desejada. Este princípio acaba tendo certa similaridade com a regra de aprendizado de Hebb apresentada na seção 3.3. Para o caso da situação apresentada na figura 9.7, os pesos a serem atualizados seriam apenas aqueles representados por  $W_{1,2}^{(2)}, W_{2,2}^{(2)}, \dots, W_{n_2,2}^{(2)}$ .

A partir da análise do processo de convergência da camada *Output*, constata-se então que os pesos de seus neurônios convergirão para os valores médios das saídas desejadas, ao passo que os pesos dos neurônios da camada de Kohonen serão ajustados em relação aos valores médios das entradas [Wasserman, 1989].

#### 9.4.2 – Algoritmo de treinamento da rede counter-propagation

O processo de treinamento da rede *counter-propagation* é dividido em dois estágios. O primeiro refere-se aos ajustes dos pesos dos neurônios da camada intermediária (camada de Kohonen), o qual segue os mesmos passos do algoritmo competitivo apresentado na seção 8.2.

Já o segundo estágio, referente aos ajustes dos pesos dos neurônios da camada de saída (camada *Output*), assume a regra de atualização dada por (9.6), que é então aplicada somente aos pesos que se conectam com a saída do neurônio vencedor advindo da camada intermediária. Neste segundo estágio,

o treinamento passa a ser supervisionado, pois há necessidade de se utilizar as saídas desejadas para realizar as sintonizações dos pesos.

O algoritmo seguinte ilustra a sequência de procedimentos computacionais, em termos de pseudocódigo, visando o treinamento da rede *counter-propagation*.

**Inicio {Algoritmo Counter-Propagation – Fase de Treinamento}**

- <1> Obter o conjunto original de amostras de treinamento { $x^{(k)}$ };
  - <2> Obter o vetor de saída desejada ( $d^{(k)}$ ) para cada amostra;
  - <3> Executar o treinamento da camada intermediária (Kohonen) usando o algoritmo competitivo // (conforme Subseção 8.2);
  - <4> Iniciar  $W_j^{(2)}$  com valores aleatórios pequenos;
  - <5> Especificar taxa de aprendizagem { $\eta$ };
  - <6> Iniciar o contador de número de épocas {época  $\leftarrow 0$ };
  - <7> Repetir as instruções:
    - <7.1> Para todas as amostras { $x^{(k)}$ }, fazer:
      - <7.1.1> Obter os valores de saída do neurônio vencedor da camada intermediária (Kohonen) em relação a cada amostra  $x^{(k)}$  // (conforme Subseção 8.2);
      - <7.1.2> Ajustar somente os pesos individuais daqueles neurônios da camada de saída (Outstar) que estão conectados ao neurônio vencedor da camada intermediária (Kohonen) // (conforme expressão (9.6));
    - <7.2> época  $\leftarrow$  época + 1;
- Até que não haja mudanças significativas na matriz de pesos  $W_j^{(2)}$ .

**Fim {Algoritmo Counter-Propagation – Fase de Treinamento}**

Após a aplicação dos procedimentos de treinamento da *counter-propagation*, esta já estará apta para ser colocada em operação quando da apresentação de novas amostras que devem ser identificadas. A sequência de passos é dada pelo algoritmo que se segue.

**Início {Algoritmo Counter-Propagation – Fase de Operação}**

{<1> Apresentar uma amostra ( $x$ );  
 <2> Obter os valores de saída do neurônio vencedor ( $u^{(V)}$ ) da camada intermediária (Kohonen) // (conforme Subseção 8.2);  
 <3> Calcular a resposta da rede, advinda dos próprios neurônios da camada Outstar, por meio da seguinte expressão:  

$$y_j = W_j^{(2)} \cdot u_j^{(V)}, \text{ onde } j = 1 \dots n_2;$$
  
 <4> Disponibilizar as saídas da rede dadas pelos elementos contidos em  $y_j$ .

**Fim {Algoritmo Counter-Propagation – Fase de Operação}**

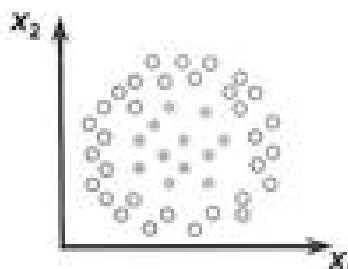
Em conformidade também com o testemunho do próprio idealizador da rede *counter-propagation* em Hecht-Nielsen (1987b), constata-se que a capacidade da mesma em resolver tanto problemas de classificação de padrões como aproximação de funções é bem limitada quando comparada com o PMC e RBF. Entretanto, uma de suas vantagens reside na simplicidade de implementação computacional, sendo aplicada com sucesso em diversos tipos de problemas de engenharia e ciências, principalmente naqueles relacionados com a compactação de informações e imagens [Dong *et alii*, 2008; Mishra *et alii*, 2006; Sygnowski & Maeukow, 1996].

## 9.5 – Exercícios

- 1) Explique as diferenças existentes entre clusterização de dados e quantização vetorial.
- 2) Considerando uma das classes a ser identificada usando quantização vetorial, explique em que eventuais situações se podem adotar os valores médios de suas amostras para propósitos de alocação do respectivo vetor quantizados.
- 3) Explicite as principais diferenças existentes entre o PMC e a *LVQ*, em relação às suas fronteiras de separabilidade de classes, quando aplicadas em problemas de classificação de padrões.
- 4) Elabore uma instrução algorítmica que poderia ser utilizada como critério de parada para o processo de treinamento de redes *LVQ*.
- 5) Discorra sobre a importância de se recomendar que os vetores de pesos dos neurônios da *LVQ* sejam inicializados com um daqueles valores de amostras

dos respectivos conjuntos em que estarião representando.

6) Sejam duas classes que estão representadas conforme o diagrama seguinte. Explique se a rede  $LLQ$  pode ser treinada visando à classificação das amostras dentro dessas duas classes.



7) Explique as principais diferenças entre os algoritmos de treinamento associados às redes  $LLQ$ -1 e  $LLQ$ -2.

8) Discorra sobre as características fundamentais que envolvem as duas camadas neurais que constituem a arquitetura das redes *backward-propagation*.

9) Explique se é possível aplicar a regra delta generalizada no ajuste dos pesos dos neurônios pertencentes à última camada da rede *backward-propagation*.

10) Comente sobre o porquê da impossibilidade de se utilizar as redes  $LLQ$  convencionais em aproximação de funções.

## 9.6 – Projeto prático

A previsão de demanda de potência elétrica é de suma importância para o planejamento operacional de um sistema de energia elétrica. A programação da manutenção do sistema, o planejamento de sua expansão e a análise de despacho são normalmente executados levando em consideração a potência elétrica prevista para o respectivo sistema.

A previsão para um dia específico pode ser efetuada levando-se em conta apenas a potência medida nas primeiras horas do dia. Baseando-se nessas informações, por intermédio de um processo de classificação de curvas, é possível traçar o perfil da demanda de potência necessária para todas as outras horas do dia.

A tabela 9.1 fornece a potência medida em 16 dias e que foram agrupadas em quatro classes (perfis) de demanda.

Tabela 9.1 – Dados para treinamento da rede LVQ-1

Amostra	7 horas	8 horas	9 horas	10 horas	11 horas	12 horas	Classe
1	2,3978	1,5328	1,9044	1,1937	2,4184	1,8649	1
2	2,3936	1,4804	1,9907	1,2732	2,2719	1,8110	1
3	2,2880	1,4585	1,9867	1,2451	2,3389	1,8099	1
4	2,2904	1,4786	1,8876	1,2706	2,2966	1,7744	1
5	1,1201	0,0587	1,3154	5,3783	3,1849	2,4276	2
6	0,9913	0,1524	1,2700	5,3808	3,0714	2,3331	2
7	1,0915	0,1881	1,1387	5,3701	3,2561	2,3383	2
8	1,0535	0,1229	1,2743	5,3226	3,0950	2,3193	2
9	1,4871	2,3448	0,9918	2,3160	1,5783	5,0850	3
10	1,3312	2,2553	0,9618	2,4702	1,7272	5,0845	3
11	1,3646	2,2945	1,0562	2,4763	1,8051	5,1470	3
12	1,4392	2,2296	1,1278	2,4230	1,7259	5,0876	3
13	2,9364	1,5233	4,6109	1,3160	4,2700	6,8749	4
14	2,9034	1,4640	4,8061	1,4598	4,2912	6,9142	4
15	3,0181	1,4918	4,7051	1,3521	4,2623	6,7966	4
16	2,9374	1,4896	4,7219	1,3977	4,1863	6,8336	4

Assim sendo, implemente e treine uma LVQ-1 que detecte as possíveis similaridades e regularidades entre todos os vetores pertencentes a cada uma das classes, objetivando-se para tanto a classificação futura de perfis de potência de outros dias (amostras).

Para as simulações, considera-se aqui uma taxa de aprendizagem  $\{\eta\}$  no valor de 0,05. Após o treinamento da rede, utilize-a para classificação dos perfis de potência para os dias apresentados na tabela 9.2.

Tabela 9.2 – Dados para teste da rede  $L^1Q_1$ 

Dia	7 horas	8 horas	9 horas	10 horas	11 horas	12 horas	Classe
1	2,9817	1,5656	4,8391	1,4311	4,1916	6,9718	
2	1,5537	2,2615	1,3169	2,5873	1,7570	5,0958	
3	1,2240	0,2445	1,3595	5,4192	3,2027	2,5675	
4	2,5828	1,5146	2,1119	1,2859	2,3414	1,8695	
5	2,4168	1,4857	1,8959	1,3013	2,4500	1,7888	
6	1,0604	0,2276	1,2806	5,4732	3,2133	2,4839	
7	1,5246	2,4254	1,1353	2,5325	1,7569	5,2640	
8	3,0565	1,6259	4,7743	1,3654	4,2904	6,9808	



Georgy Fedosovich Voronoi  
(Георгий Федосович Вороной)

## Redes ART (*adaptive resonance theory*)

### 10.1 – Introdução

A teoria da ressonância adaptativa (*adaptive resonance theory - ART*), proposta originalmente por Grossberg (1976a, 1976b), foi desenvolvida a partir da observação de alguns fenômenos biológicos envolvendo a visão, a fala, o desenvolvimento cortical e as interações cognitivo-emocionais. A motivação desta teoria é baseada em três princípios biológicos que são destacados pelas seguintes características:

- *Normalização de sinal* → capacidade dos sistemas biológicos em se adaptar a ambientes que se alteram em todo momento. Como exemplo, tem-se o sistema oftalmológico humano que se adapta rapidamente frente à quantidade de luz disponível;
- *Intensificação de contraste* → potencialidade de identificar detalhes sutis a partir de sucessivas observações realizadas sobre o ambiente. Como exemplo, o sistema respiratório é capaz de diferenciar de forma quase que instantânea um ambiente limpo que começou a ser poluído com monóxido de carbono;
- *Memória de curto prazo* → possibilidade de armazenar momentaneamente as informações sensoriais advindas do mecanismo de intensificação de contraste, antes que possam ser decodificadas visando à tomada de decisões.

Assim, por intermédio da proposição desta teoria, vários modelos de redes neurais artificiais, com treinamento tipicamente não-supervisionado (com característica recorrente), foram sequencialmente propostas.

Um dos principais atributos das redes ART é a sua habilidade de aprender novos padrões sem destruir os conhecimentos já anteriormente extraídos, por ocasião da apresentação de outras amostras. Esta característica está diretamente associada ao intrigante dilema plasticidade/estabilidade [Carpenter & Grossberg, 1988], no qual o sistema deve ser flexível (adaptativo) e suficiente para incorporar mudanças ocorridas no ambiente, ao passo que deve ser ainda bem estável a fim de preservar o conhecimento já adquirido no decorrer do tempo.

As topologias baseadas na arquitetura ART podem ser divididas em cinco grupos principais que são classificadas em função do tipo de suas entradas e seus processos de treinamento, ou sejam:

- ART-1 → Padrões com entradas binárias e treinamento não-supervisionado [Carpenter & Grossberg, 1987a];
- ART-2 → Padrões com entradas binárias ou analógicas (contínuas) e treinamento não-supervisionado [Carpenter & Grossberg, 1987b];
- ART-3 → Padrões com entradas binárias ou analógicas e treinamento não-supervisionado, usando-se aqui topologia multinível e "transmissores químicos" para controle do processo de busca pela melhor solução [Carpenter & Grossberg, 1990];
- ART-Map → Padrões com entradas binárias ou analógicas e treinamento supervisionado em tempo-real [Carpenter *et alii*, 1991a], utilizando-se duas redes ART em sua estrutura;
- Fuzzy-ART → Versão fuzzy da ART-MAP e que possui também aprendizado supervisionado [Carpenter *et alii*, 1991b].

Em função da elevada aplicabilidade, assim como de sua relativa simplicidade de implementação, abordar-se-á neste livro somente a topologia ART-1, sendo que as propriedades e os princípios relacionados à teoria de ressonância adaptativa são os mesmos para os cinco grupos de redes ART.

Por certo, conforme enfatizado anteriormente, a essência de todas as

topologias ART está baseada no princípio de como incorporar conhecimentos associados a novos padrões de entrada, sem a necessidade de modificar substancialmente o aprendizado já inserido anteriormente na rede por intermédio da apresentação de outras amostras.

As aplicações das redes ART-1 são as mais variadas possíveis, sendo encontradas em diversas áreas do conhecimento. Como exemplos, em Young *et alii* (2007) foi utilizada uma arquitetura ART-1 para detectar regiões espaciais de interesse astronómico, assim como caracterizar os ruídos que estão presentes nas informações coletadas.

Já em Cho *et alii* (2006) foi descrita uma metodologia utilizando redes ART-1 para classificação baseada em gráf computacional visando aplicações em problemas de bioinformática. Em Tateyama & Kawata (2004) foi desenvolvida uma tecnologia de agrupamento usando redes ART-1, a fim de dividir as máquinas de uma planta industrial em células de fabricação, tendo-se o intuito de se aumentar sua produtividade.

Em Rajasekaran & Raj (2004) foi apresentada outra aplicação de redes ART-1 para extração de características e reconhecimento de imagens monocromáticas de satélites. Em Baraldi & Alpaydin (2002) foi também proposta uma estrutura neural baseada na rede ART-1 visando clusterização de dados em tempo-real.

Finalmente, em Srinivasan & Batur (1994) foi apresentada uma abordagem integrando redes de Hopfield e redes ART-1 para detectar e isolar faltas em sistemas dinâmicos lineares.

## 10.2 – Estrutura topológica da rede ART-1

A estrutura topológica representando uma rede do tipo ART-1, idealizada por Carpenter & Grossberg (1987a), é ilustrada na figura 10.1, cujas amostras (vetores) de entrada são tipicamente compostas por valores binários.

A principal aplicabilidade das redes ART-1 está na solução de problemas de classificação de padrões, sendo também considerada como um sistema classificador de vetores.

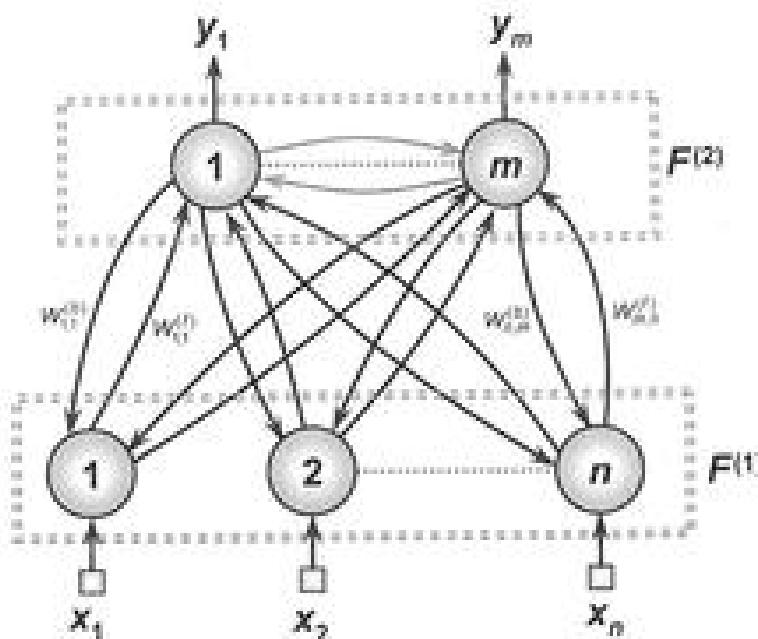


Figura 10.1 – Estrutura topológica da rede ART-1.

Em resumo, considerando uma amostra representada pelo vetor de entrada  $\{x\}$ , sua finalidade consiste em classificá-la em uma das classes representativas do problema, utilizando-se para tanto uma medida de similaridade deste vetor em relação aos outros vetores que já compõem as referidas classes. É válido ressaltar que, em se tratando de aprendizado não-supervisionado, há então a inexistência da saída desejada.

Em complemento, observa-se por meio da figura 10.1 que a rede ART-1 é constituída de duas camadas de neurônios (campos representativos  $P^1$  e  $P^2$ ), que estão totalmente conectados uns aos outros, ou seja, todos os neurônios da camada  $P^1$  estão conectados a todos os neurônios da camada  $P^2$  e vice-versa.

A camada  $P^1$ , denominada de camada de entrada ou camada de comparação, possui uma quantidade de neurônios que coincide com a dimensão das amostras de entrada (dimensão do vetor  $x$ ), sendo que sua finalidade está também incumbida de repassar tais padrões originais para os neurônios da camada  $P^2$ . Assim, conforme representado na figura 10.2, o fluxo de informações impetrado pela camada  $P^1$  é continuamente adiante (*feedforward*), isto é, sempre em direção à camada  $P^2$ .



Figura 10.2 – Fluxo de informações na rede ART-1

Por conseguinte, as informações realimentadas (*feedback*), a partir da camada  $F^2$ , constituem assim os seus respetivos sinais de entrada, os quais serão então ponderados pela sua matriz de pesos (definida por  $W^{21}$ ), composta por elementos binários.

Portanto, a missão principal da camada  $F^2$  consiste em realizar uma operação de comparação a fim de verificar se o vetor de entrada  $\{x\}$ , previamente associado a uma das classes representadas pelos neurônios da camada de saída, possui similaridade com o conjunto de vetores já armazenados.

Já a camada  $F^0$ , denominada de camada de saída ou camada de reconhecimento, tem a eventual missão de reconhecer e incorporar o conhecimento trazido por um vetor de entrada para uma das classes que estão sendo representadas pelos seus neurônios. Portanto, a quantidade de neurônios desta camada é dinâmica, e varia de acordo com a necessidade de se adicionar novos neurônios, para que novas classes possam ser representadas. Esta característica dinâmica é uma das principais potencialidades de redes ART.

Outra particularidade desta camada  $F^0$ , diferentemente do comportamento entre os neurônios da camada de entrada  $F^1$ , é que estes são competitivos entre si (apresentam conexões laterais). O fluxo de informações desta camada ( $F^0$ ) realimenta (*feedback*) os neurônios da camada de entrada ( $F^1$ ). Sua matriz de pesos (definida por  $W^0$ ), constituída de elementos reais, fica incumbida de ponderar as características das componentes do vetor de entrada que é apresentado, de tal modo a estimular que um dos neurônios re-

presentativos das classes do problema possa ser então ativado como sinal de reconhecimento do referido vetor.

Similarmente às redes de Kohonen, apenas um único neurônio da camada de reconhecimento da ART-1 estará ativo em cada ciclo, sendo este considerado o vencedor da competição de proximidade.

### 10.3 – Princípio da ressonância adaptativa

O funcionamento da rede ART-1 é baseado no próprio princípio da ressonância adaptativa, quando da apresentação de um novo vetor (representando uma amostra) em sua camada de entrada.

Em primeira instância, cada vetor  $\{x\}$  apresentado será propagado da camada de entrada  $P^0$  em direção à camada de saída  $P^1$ , cujos valores de seus sinais componentes serão ponderados pela respectiva matriz de pesos  $\{W^1\}$  da camada  $P^1$ , tendo-se o intuito de extrair as suas características discriminantes, as quais visam a estimulação de uma das classes que estão sendo representadas pelos seus neurônios.

Em segunda instância, quando da chegada do referido padrão (ponderado por  $W^1$ ) junto à camada de saída  $P^1$ , os neurônios desta camada, os quais estarão representando as classes do problema a ser mapeado, entrarião, portanto, em competição com o objetivo de verificar qual deles possui maior proximidade com aquele vetor que acabou de chegar, sendo assim declarado um vencedor para esta competição.

Em terceira instância, o neurônio ganhador encaminha a classe vencedora de volta à camada de entrada  $P^0$ . Durante este processo, os neurônios da camada  $P^1$ , por intermédio dos pesos armazenados na matriz  $W^1$ , realizam então o teste de similaridade a fim de verificar a aderência daquele vetor de entrada à classe representada pelo neurônio vencedor de  $P^1$ .

Em quarta instância, em decorrência do resultado do teste de similaridade, se o vetor de pesos do neurônio vencedor for “parecido” com o respectivo vetor de entrada, levando-se em consideração um limiar de aderência (parâmetro de vigilância), haverá então o processo de ressonância entre ambos. Em consequência, o referido vetor de entrada será assim associado àquela classe que o neurônio vencedor esteja representando, sendo que a finalização deste ciclo de reconhecimento é executada por meio da atualização de ambas as matrizes de pesos.

Caso contrário, em quinta instância, o atual neurônio vencedor (ativo), embora tenha ganhado a competição com os demais neurônios em  $P^0$ , não é aquele mais apropriado para representar a classe associada ao referido vetor de entrada. Nesta situação, no próximo ciclo, este deve ser desabilitado para que outro neurônio da camada  $P^0$  possa então ter a chance de vencer a competição, repassando-se (em seguida) a nova classe vencedora para a realização do teste de similaridade. Tal processo se repete a cada novo ciclo até que haja definição de uma classe que seja compatível com o referido vetor de entrada.

Finalmente, em última instância, caso nenhum dos neurônios da camada  $P^0$  seja aprovado no teste de similaridade, um novo neurônio deve ser então adicionado/habilitado na camada  $P^0$  a fim de identificar esta outra classe representada pelo vetor de entrada em questão. De fato, tratar-se-á então de uma nova classe, pois todas as outras já existentes, as quais estão sendo representadas pelos outros neurônios da camada  $P^0$ , não são passíveis de englobar as novas características de conhecimento e/ou comportamento trazido pelo referido vetor de entrada.

Na realidade, o processo de ressonância realiza basicamente um teste de hipótese. Nesta circunstância, uma hipótese sobre a classe apropriada para incorporar o vetor de entrada é formulada pela camada de reconhecimento  $P^0$ . Em seguida, a camada de comparação  $P^1$  entra em operação a fim de validar a hipótese proposta, ou seja, se o vetor de entrada tiver um grau de similaridade aceitável à classe sugerida, há então a validação da hipótese; caso contrário, a hipótese previamente formulada estava equivocada e outras classes (neurônios) devem ser então avaliadas, ou até mesmo criadas, nos próximos ciclos.

#### 10.4 – Aspectos de aprendizado da rede ART-1

O processo de aprendizado de novos conhecimentos pela rede ART-1 pode ser mais bem entendido por meio do fluxograma sistematizado na figura 10.3.

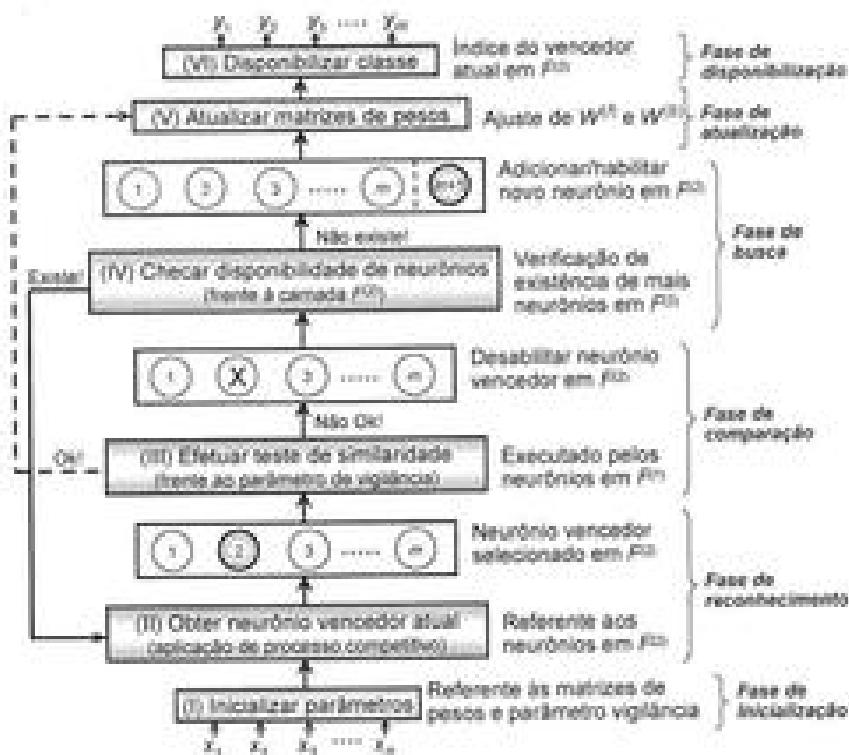


Figura 10.3 – Fluxograma de funcionamento da rede ART-1

Baseado nos blocos deste fluxograma poder-se-á então dividir o processo de aprendizado da rede ART-1 em seis etapas principais, as quais coincidem com os blocos sombreados na figura 10.3.

#### (I) Primeira etapa: fase de inicialização de parâmetros

As instruções que compõem esta fase compreendem a atribuição de valores para as matrizes de pesos  $W^0$  e  $W^1$ , assim como a especificação do parâmetro de vigilância ( $\rho$ ), cuja missão é arbitrar o teste de similaridade a ser executado pelos neurônios da camada de entrada  $P^0$ .

Os elementos da matriz de pesos  $W^0$ , referentes à camada de saída  $P^0$ , devem ser inicializados com valores pequenos, tendo-se o objetivo de evitar a saturação dos resultados produzidos pelos neurônios desta camada, quando da apresentação dos vetores de entrada. Assim, têm-se os seguintes valores para os elementos de  $W^0$ :

$$W_j^{(l)} = \frac{1}{1+n}, \text{ onde } \begin{cases} j = 1, \dots, m \\ j = 1, \dots, n \end{cases} \quad (10.1)$$

onde  $n$  é o número de neurônios na camada de entrada  $P^0$  e  $m$  é a quantidade de neurônios habilitados na camada de saída  $P^0$ .

Já os elementos da matriz de pesos  $W^0$ , referentes à camada de entrada  $P^0$ , são todos inicializados com valores unitários, os quais indicam que cada neurônio pertencente à camada  $P^0$  estará a princípio habilitado para enviar sua resposta aos neurônios da camada  $P^0$ . Portanto, tem-se:

$$W_j^{(0)} = 1, \text{ onde } \begin{cases} j = 1, \dots, n \\ j = 1, \dots, m \end{cases} \quad (10.2)$$

Em relação ao parâmetro de vigilância  $\{\rho\}$ , o seu valor deve estar compreendido entre 0 e 1 ( $0 < \rho < 1$ ). É também válido ressaltar que quanto mais elevado for o valor de  $\rho$ , mais detalhes discriminantes serão levados em consideração pelos neurônios da camada  $P^0$  quando do teste de similaridade. Esta estratégia permite que diferenças sutis entre padrões de entrada possam implicar na criação de novas classes, isto é, necessitar-se-á adicionar ou habilitar mais neurônios junto à camada  $P^0$ .

Por outro lado, valores muito baixos para o parâmetro de vigilância implicam em relaxar a capacidade de discriminação da rede, que estará avaliando somente diferenças que sejam muito estruturais entre as amostras, sendo que nesta condição o número de classes será também bem pequeno.

## (II) Segunda etapa: fase de reconhecimento

A fase de reconhecimento consiste de tentar categorizar (enquadrar) um vetor de entrada apresentado à rede dentro de uma das classes já disponíveis, representadas pelos neurônios pertencentes à camada de saída  $P^0$ .

Este processo de reconhecimento é similar ao método competitivo utilizado na rede de Kohonen, ou seja, o neurônio da camada  $P^0$  que tiver a maior ativação (casamento), frente ao vetor de entrada  $\{x\}$  que foi apresentado, será então declarado o vencedor da competição, isto é, aplica-se a estratégia do "vencedor leva tudo" (*winner-take-all*), conforme apresentado na seção

8.2. O grau de ativação relativo ao  $j$ -ésimo neurônio da camada  $F^0$  é então produzido pela ponderação dos sinais do vetor de entrada pelos respectivos pesos. Em termos matemáticos, tem-se:

$$v_j = \sum_{i=1}^n W_j^{(l)} \cdot x_i , \text{ onde } j = 1, \dots, m \quad (10.3)$$

Consequentemente, o neurônio vencedor ( $k$ ) é então aquele que produzir o maior valor de ativação, isto é:

$$k = \arg \max_j \{v_j\} \quad (10.4)$$

onde  $k$  está recebendo um valor inteiro que representa o índice do neurônio vencedor atual (ativo), o qual por sua vez inibe a ação dos demais neurônios da camada  $F^0$  por meio de suas conexões laterais. Neste caso, o neurônio vencedor estará representando a classe candidata em que o respectivo vetor de entrada poderá ser eventualmente inscrito.

Adicionalmente, o neurônio vencedor produzirá valor igual a 1 em sua saída, cuja finalidade aqui é tanto de indicar a classe candidata do vetor de entrada apresentado, como também realimentar de forma excitatória os neurônios da camada  $F^0$ . Por outro lado, por meio das conexões laterais, todos os outros neurônios da camada  $F^0$  produzirão valores iguais a 0, indicando que as classes que estão representando não foram as mais apropriadas para incorporar o referido vetor, resultando assim numa realimentação inibitória para os neurônios da camada  $F^0$ . Em termos matemáticos, tem-se:

$$y_j = \begin{cases} 1, & \text{se } j = k \\ 0, & \text{se } j \neq k \end{cases}, \text{ onde } j = 1, \dots, m \quad (10.5)$$

onde  $k$  é o índice do neurônio vencedor.

Em virtude do fato de que a quantidade de neurônios da camada  $F^0$  é dinâmica, pois mais neurônios serão adicionados ou habilitados conforme a necessidade de se criar novas classes, infere-se que, quando da apresentação do primeiro vetor de entrada, um único neurônio estará então presente ou habilitado na camada  $F^0$ .

### (III) Terceira etapa: fase de comparação

Após o procedimento de categorização do vetor de entrada frente à aplicação da fase de reconhecimento, passar-se-á então para a fase de comparação, cujo propósito é verificar se a classe candidata representada pelo neurônio vencedor é a mais apropriada para receber o referido vetor.

Para tal propósito, os resultados produzidos pelos neurônios da camada  $P^{(1)}$  são realimentados aos neurônios da camada  $P^{(2)}$  a fim de se proceder ao teste de similaridade. Como somente o neurônio vencedor  $k$  produziu valor igual a 1 em sua saída, os valores que serão então realimentados da camada  $P^{(1)}$  aos neurônios da camada  $P^{(2)}$  são os próprios pesos associados às interligações deste neurônio vencedor  $k$  aos respectivos neurônios de  $P^{(2)}$ . A figura 10.4 ilustra este processo de realimentação entre os neurônios das camadas  $P^{(1)}$  e  $P^{(2)}$ .

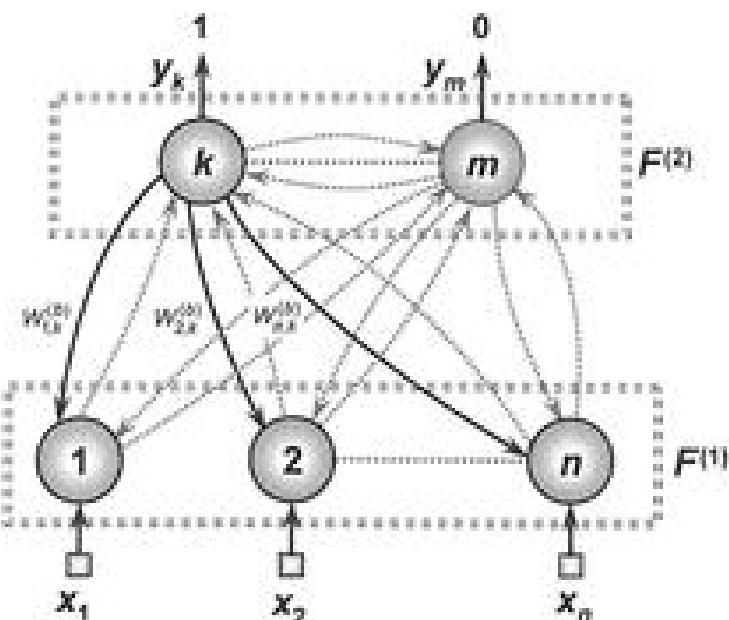


Figura 10.4 – Fluxo de realimentação do neurônio vencedor para as entradas

O teste de similaridade é, por conseguinte, realizado por meio da comparação dos valores de  $W_{1,k}^{(2)}$  com  $x_1$ , de  $W_{2,k}^{(2)}$  com  $x_2$ , e assim por diante. Conforme definido anteriormente, tanto os elementos da matriz  $W^{(2)}$  como aqueles também do vetor  $x$  são binários, o que facilita o processo comparativo. A regra normalmente utilizada consiste em comparar o valor do parâmetro

de vigilância com a razão  $R$ , que é obtida da quantidade de elementos unitários que sejam comuns {considerando  $W_{j,k}^{(b)}$  e  $\mathbf{x}_j$ } em relação à quantidade de elementos unitários presentes no vetor  $\mathbf{x}$ . Tal razão de similaridade pode ser formalmente descrita por:

$$R = \frac{\sum_{j=1}^n W_{j,k}^{(b)} \cdot x_j}{\sum_{j=1}^n x_j} \quad (10.6)$$

onde  $k$  é o índice do neurônio vencedor e  $n$  é a quantidade de componentes do vetor  $\mathbf{x}$ . Como exemplo, assumindo que o vetor  $\mathbf{x}$  seja constituído de oito componentes de entrada ( $n = 8$ ), e que o neurônio com índice 1 tenha sido o vencedor ( $k = 1$ ) da competição de proximidade, considera-se aqui os seguintes valores que foram apresentados para os vetores  $\mathbf{x}$  e  $W^{(b)}$ :

$$\mathbf{x}_j = [ \begin{array}{cccccccc} j=1 & j=2 & j=3 & j=4 & j=5 & j=6 & j=7 & j=8 \end{array} ]^T \quad (10.7)$$

$$W_{j,1}^{(b)} = [ \begin{array}{cccccccc} j=1 & j=2 & j=3 & j=4 & j=5 & j=6 & j=7 & j=8 \end{array} ]^T \quad (10.8)$$

Para este caso específico, o valor do numerador de (10.6) será igual a 4, pois a operação envolvida verifica somente quantos elementos unitários são comuns entre os dois vetores dados em (10.7) e (10.8), cujas posições ocupadas estão em  $j = 1, j = 3, j = 6$  e  $j = 8$ . Esta operação obtém uma medida inspirada na distância de Hamming [Moon, 2005], que computa a quantidade de bits diferentes entre duas cadeias (*strings*) binárias ao invés de computar a quantidade de bits iguais. Já o denominador de (10.6) indica quantos elementos unitários estão compondo o vetor  $\mathbf{x}$ , cujo valor retornado quando aplicado em (10.7) será igual a 5. Consequentemente, considerando esta situação, o valor resultante da razão  $R$  será igual a 0,8.

Por conseguinte, se o resultado desta razão  $R$  for maior que o valor especificado para o parâmetro de vigilância ( $R > \rho$ ), considerar-se-á então que o vetor de entrada foi aprovado no teste de similaridade, pois a classe simbolizada pelo neurônio vencedor  $k$  de  $F^{(b)}$  é de fato a mais apropriada para representar as

características envolvidas com o referido padrão. Em seguida, haverá assim a atualização das matrizes de pesos  $W^a$  e  $W^b$ , por meio da aplicação da fase (V), a fim de também representar os atributos deste novo vetor de entrada.

Caso contrário, se o resultado da razão  $R$  for menor ou igual que o parâmetro de vigilância ( $R \leq \beta$ ), é possível assumir que a classe representada pelo neurônio vencedor  $k$  estará incapaz de representar o comportamento do referido vetor de entrada, pois há diferenças significativas com os que já estão inseridos dentro daquele agrupamento. Consequentemente, outro neurônio da camada  $P^B$  deverá ser selecionado como vencedor, ficando agora incumbido de tentar representar a classe candidata, tendo assim também a chance de tentar mostrar que a sua classe tem mais similaridade com a amostra apresentada.

#### (IV) Quarta etapa: fase de busca

Neste estágio de processamento, decorrente da rejeição do neurônio vencedor frente ao teste de similaridade, buscar-se-á outro neurônio (pertencente a  $P^B$ ) que tenha mais afinidade com a amostra inserida. Para tanto, o neurônio vencedor do estágio anterior deverá ser desabilitado, a fim de se evitar que possa novamente ganhar a competição. Uma das formas de se proceder tal operação está em atribuir o valor zero ao potencial de ativação ( $u_i$ ) do vencedor, pois assim outro neurônio será o ganhador da competição.

Desta forma, conforme se pode verificar no fluxograma da figura 10.3, procede-se sucessivas vezes com a aplicação sequencial das fases (II), (III) e (IV) até que seja possível a obtenção de um neurônio vencedor de  $P^B$  que seja aprovado no teste de similaridade.

Deve-se também ressaltar que esta capacidade de as redes ART incorporar, de forma automática, novos neurônios em sua camada de saída tem sido uma de suas principais características, pois a rede se torna flexível o suficiente para inserir novos conhecimentos em sua estrutura sem destruir tudo que já tenha sido aprendido. Os procedimentos algorítmicos para esta adição de novos neurônios na camada  $P^B$  podem ser implementados de duas formas possíveis.

A primeira consiste da utilização de estruturas de dados estráticas, tais como um vetor contendo os atributos dos neurônios, em que de antemão é declarada a sua dimensão máxima (quantidade máxima de neurônios que pode ter  $P^B$ ). Nesta circunstância, um dos atributos do neurônio seria informar seu *status*, indicando-se se o mesmo estará ativo (habilitado) ou inativo (desabilitado) em determinado ciclo.

Alternativamente, pode-se também definir um contador que sempre indicará a quantidade de neurônios que estarão ativos na camada  $F^n$  em cada ciclo. Como exemplo, se for definido que o número máximo de neurônios em  $F^n$  seja 50, o contador necessariamente estará indicando que apenas 01 neurônio estará ativo quando da apresentação da primeira amostra. Este número será automaticamente incrementado, conforme novos neurônios sejam habilitados para representação de outras classes. A figura 10.5 ilustra esta diagramação estrutural, levando-se em conta que cinco neurônios estarão ativos para representação de classes, ou seja, somente esses cinco neurônios poderão ser considerados para a competição de proximidade.

Neurônio	1	2	3	4	5	6	7	...	50
Status	Ativo	Ativo	Ativo	Ativo	Ativo	Inativo	Inativo		Inativo
$u_i$	0,32	0,87	0,45	0,93	0,15	—	—		—
$y_i$	0	0	0	1	0	—	—		—

Figura 10.5 – Representação dos neurônios da ART-1 usando estruturas de dados estáticas

Para o caso hipotético representado na figura 10.5, observa-se também que o neurônio de número 4 venceu a competição de proximidade, pois possui o maior valor de ativação ( $u_i$ ), sendo que nesta situação somente sua saída ( $y_i$ ) terá valor igual a um. Caso este neurônio e, posteriormente, todos os demais que estão ativos não sejam aprovados no teste de similaridade, habilitar-se-á então o neurônio 6 por meio da alteração de seu status para a condição "ativo".

A segunda forma consiste do uso de estruturas de dados dinâmicas, como uma lista encadeada ou ligada [Wirth, 1989] que conterá também cada um dos atributos dos neurônios. Para esta estrutura de dados, cada um dos neurônios é considerado um objeto, os quais serão interligados por ponteiros a fim de formar a estrutura lógica da lista.

Assim, inicialmente, um único objeto neurônio estará disponibilizado, sendo que outros serão gerados e dinamicamente interligados à medida que houver necessidade de se inserir novas classes. Diferentemente das estruturas de dados estáticas, o parâmetro de *status* é aqui desnecessário, pois todo neu-

rônio que compõe a lista estará ativo a partir do momento de sua inclusão. A figura 10.6 ilustra esta estrutura de dados dinâmica que pode também ser utilizada para a representação dos neurônios da rede ART-1.



Figura 10.6 – Representação dos neurônios da ART-1 usando estrutura de dados dinâmica.

Nota-se então, por meio da figura 10.6, que quando uma nova classe for necessária, um novo objeto neurônio (número 6) será criado e incorporado à estrutura dinâmica já formada anteriormente. Desta forma, tal estrutura tem a característica de ir aumentando conforme haja necessidade.

#### (V) Quinta etapa: fase de atualização

Após a conclusão bem sucedida da fase de comparação (III), procede-se com a fase de atualização dos pesos das matrizes  $W^b$  e  $W^a$ , a fim de que incorporem o conhecimento trazido pela última amostra apresentada. Para tanto, somente aqueles elementos referentes ao neurônio vencedor  $\{k\}$  serão atualizados por meio da aplicação das seguintes expressões:

$$W_{kj}^{(b)}(t+1) = \frac{W_{jk}^{(b)}(t) \cdot x_j}{2 + \sum_{i=1}^n W_{ik}^{(b)}(t) \cdot x_i}, \text{ onde } j = 1, \dots, n \quad (10.9)$$

$$W_{jk}^{(b)}(t+1) = W_{jk}^{(b)}(t) \cdot x_j, \text{ onde } j = 1, \dots, n \quad (10.10)$$

A atualização dos pesos do neurônio vencedor em relação à matriz  $W^b$ , em (10.9), os quais podem ser visualizados na figura 10.7, se processa por intermédio da realização de uma simples operação lógica de conjunção booleana (porta AND) entre os elementos  $W_{jk}^{(b)}$  e  $x_j$ . Na tabela 10.1 é mostrado o resultado da expressão (10.9) quando aplicada em uma rede ART-1 hipotética, constituída de oito entradas ( $n = 8$ ), em que tem sido assumido que o neurônio vencedor de número 2 ( $k = 2$ ) foi aquele aprovado no teste de similaridade.

Tabela 10.1 – Exemplo de valores de ajuste para  $W^{(k)}$ 

		$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
Pesos do vencedor	$W_{j,2}^{(k)}(t)$	1	0	0	1	1	1	0	1
Entradas da ART-1	$x_j$	1	0	1	1	1	0	1	1
Ajuste de $W^{(k)}$ (Porta AND)	$W_{j,2}^{(k)}(t+1)$	1	0	0	1	1	0	0	1

Já o ajuste dos pesos do neurônio vencedor em relação à matriz  $W^k$ , cujas conexões de interesse são destacadas na figura 10.4, se processa por meio de duas operações.

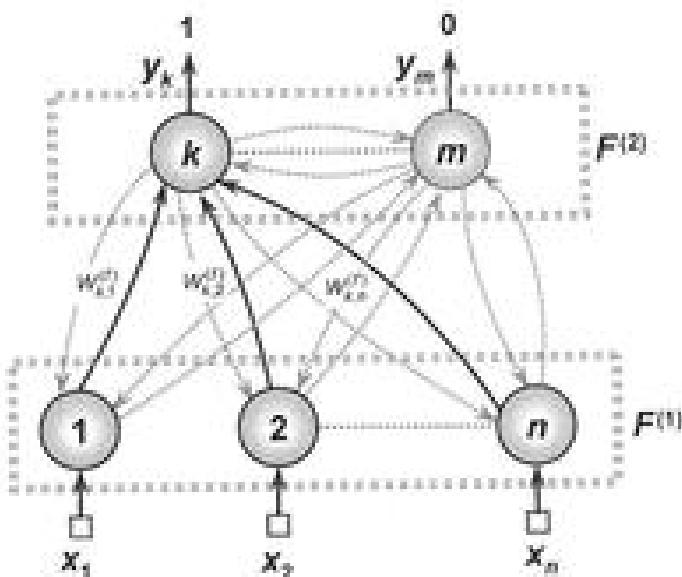


Figura 10.7 – Fluxo de alimentação das entradas para o neurônio vencedor

A primeira operação, realizada pelo numerador de (10.10), efetua novamente a aplicação da porta AND fornecida em (10.9). Já para a segunda operação, executada pelo denominador de (10.10), a somatória obtém como resultado a quantidade de elementos unitários que estejam justapostos, tanto nos pesos do neurônio vencedor (em referência a  $W^k$ ) como nas componentes do vetor de entrada.

Na tabela 10.2 é ilustrado um exemplo numérico visando o ajuste de

$W^h$ , considerando-se aqui as mesmas condições e parâmetros utilizados para a obtenção dos valores na tabela 10.1.

Tabela 10.2 – Exemplo de valores de ajuste para  $W^h$

		$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
<b>Pesos do vencedor</b>	$W_{j,2}^{(b)}(t)$	1	0	0	1	1	1	0	1
<b>Entradas da ART-1</b>	$x_j$	1	0	1	1	1	0	1	1
<b>Ajuste de <math>W^h</math></b>	$W_{j,2}^{(f)}(t+1)$	$\frac{2}{9}$	0	0	$\frac{2}{9}$	$\frac{2}{9}$	0	0	$\frac{2}{9}$

Conforme se verifica no exemplo apresentado, constata-se que haverá somente estímulo excitatório em  $W_{j,2}^{(f)}(t+1)$  para aquelas conceções cujas componentes  $W_{j,2}^{(b)}(t)$  e  $x_j$  possuam concomitantemente valores unitários.

#### (VI) Sexta etapa: fase de disponibilização

De acordo com o fluxograma ilustrado na figura 10.3, a sexta etapa do processo de treinamento da rede ART-1 consiste em disponibilizar o rótulo da classe em que a amostra apresentada vai ser inserida. Neste caso, após a execução de todas as fases anteriores, basta aplicar a expressão dada em (10.5) a fim de se obter a classe selecionada, pois será aquela representada pelo neurônio vencedor da competição de proximidade.

### 10.5 – Algoritmo de treinamento da rede ART-1

O processo de treinamento da rede ART-1 é estruturado nas seis etapas apresentadas na seção anterior, cujas principais instruções em pseudocódigo são descritas na sequência.

Diferentemente das demais arquiteturas neurais estudadas ao longo deste livro, verifica-se que, tanto a fase de treinamento como a fase de operação da rede ART-1, estão incluídas no mesmo algoritmo, pois tal topologia sempre precisará executar o teste de similaridade para proceder com a categorização da referida amostra apresentada. Além disso, o aprendizado se processa de ma-

neira não-supervisionada, possibilitando a inserção de conhecimentos dentro das classes já representadas pelos neurônios existentes, ou ainda, avaliando-se a necessidade de incluir ou habilitar outros neurônios à medida que novos conhecimentos relevantes são trazidos pelas amostras inseridas.

#### Início {Algoritmo ART-1 II Fase de Treinamento e Operação}

- <1> Apresentar o vetor de entradas ( $x$ );
- <2> Inicializar as matrizes  $W^{(1)}$  e  $W^{(2)}$ ; {conforme (10.1) e (10.2)}
- <3> Especificar o parâmetro de vigilância ( $\rho$ ); {definido entre 0 e 1}
- <4> Repetir as instruções:
  - <4.1> Calcular as ativações neurais ( $u_i$ ); {conforme (10.3)}
  - <4.2> Obter o neurônio vencedor atual ( $k$ ); {conforme (10.4)}
  - <4.3> Calcular a razão de similaridade ( $R$ ); {conforme (10.6)}
  - <4.4> Se ( $R > \rho$ ) fazer:
    - <4.4.1> Atualizar  $W^{(1)}$  e  $W^{(2)}$ ; {conforme (10.9) e (10.10)}
    - <4.4.2> Atribuir  $x$  à classe do neurônio vencedor;
    - <4.4.3> Vencedor  $\leftarrow$  "Aprovado";
  - <4.5> Se ( $|R \leq \rho$ ) e (Existe neurônio disponível) fazer:
    - <4.5.1> Desabilitar neurônio vencedor atual ( $u_k = 0$ );
    - <4.5.2> Vencedor  $\leftarrow$  "Reprovado";
  - <4.6> Se ( $|R \leq \rho$ ) e (Não existe neurônio disponível) fazer:
    - <4.6.1> Incluir ou habilitar um novo neurônio;
    - <4.6.2> Atualizar  $W^{(1)}$  e  $W^{(2)}$ ; {conforme (10.9) e (10.10)}
    - <4.6.3> Atribuir  $x$  à classe do novo neurônio;
    - <4.6.4> Vencedor  $\leftarrow$  "Aprovado";
- <5> Até que: Vencedor = "Aprovado";
- <5> Habilitar novamente todos os neurônios em  $F^{(2)}$ .

Fim {Algoritmo ART-1 II Fase de Treinamento e Operação}

Na realidade, a rede ART-1 acaba implementando o algoritmo de clusterização conhecido como *leader-follower* [Duda *et al.*, 2001], que é também denominado de estratégia *fóllas-the-leader* [Hartigan, 1975], cuja missão primeira é tentar encontrar uma classe apropriada que se possa agrupar uma amostra apresentada. Caso a amostra possua um desvio muito elevado, em relação aos elementos já armazenados naquela classe candidata, o objetivo do algoritmo está agora em criar uma nova classe para armazenar os seus atributos.

## 10.6 – Aspectos da versão original da rede ART-1

A versão original da rede ART-1, conforme concebida por Carpenter & Grossberg (1987a), é ainda formulada em termos de equações diferenciais de primeira ordem, na qual há também três parâmetros binários de controle ou de ganho ( $G_x$ ,  $G_y$ , *Reset*), que são inseridos no seu algoritmo de aprendizado, a fim de controlar o fluxo de execução entre as camadas  $P^{in}$  e  $P^{out}$ , ou seja:

- Parâmetro  $G_x \rightarrow$  responsável pelo *status* de ativar o fluxo de execução para a camada  $P^{in}$ , tendo ainda a finalidade de ativar a alocação de novos neurônios nesta camada. A saída deste parâmetro estará ativa (1) se pelo menos uma das componentes do vetor de entradas  $\{x\}$  for igual a 1. Esta saída de  $G_x$  pode ser visualizada como o resultado da aplicação da porta lógica "OU" (OR) entre todas as componentes do vetor de entradas;
- Parâmetro  $G_y \rightarrow$  responsável pelo *status* de ativar o fluxo de execução para a camada  $P^{in}$ . A sua resposta estará ativa (1) se também pelo menos uma das componentes do vetor de entradas  $\{x\}$  for igual a 1, estando ainda todas as saídas dos neurônios da camada  $P^{in}$  produzindo valores que são iguais a 0;
- Parâmetro *Reset*  $\rightarrow$  responsável pelo *status* obtido pela aplicação do teste de similaridade. Fornece resposta em estado ativo (1) se o neurônio vencedor de  $P^{in}$  for reprovado no teste de similaridade.

Conforme observado na figura 10.8, cada um dos neurônios da camada  $P^{in}$  recebe três valores binários, isto é, a respectiva componente do vetor de entrada, a respectiva soma ponderada dos valores advindos dos neurônios da camada de saída  $P^{out}$ , e o valor do parâmetro de controle  $G_x$ , que será o mesmo para todos.

O fluxo de execução estará em si na camada  $P^{in}$  quando for observada a concretização da regra dos dois terços, ou seja, pelo menos dois destes três valores devem estar ativos (produzindo saídas iguais a um). Uma descrição mais bem detalhada deste mecanismo é apresentada em Carpenter & Grossberg (1987b).

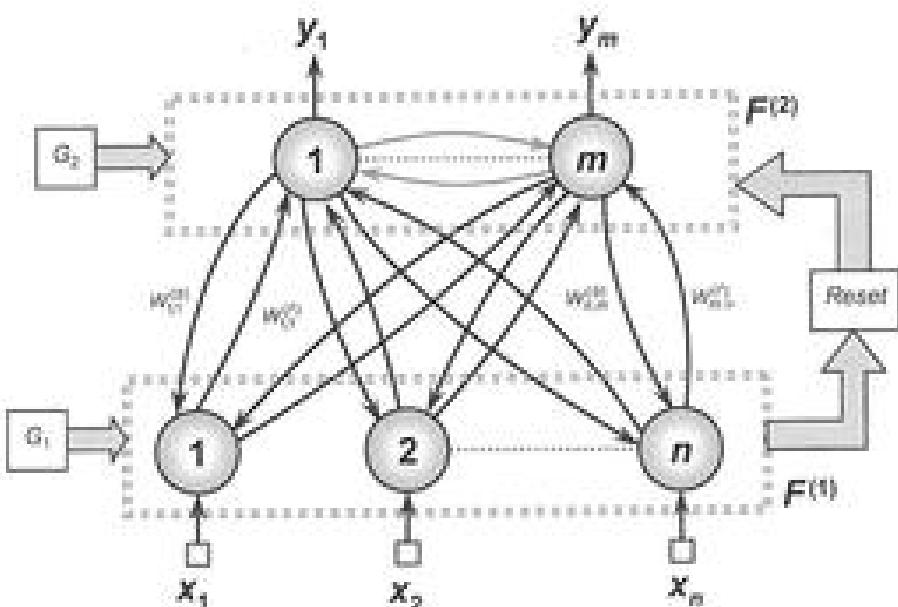


Figura 10.8 – Parâmetros de controle da versão original da rede ART-1

A operação da versão original da ART-1, com os seus parâmetros de controle, acaba realizando as mesmas instruções regidas pelo algoritmo apresentado na seção 10.5, a qual pode ser sintetizada pelos seguintes passos:

- Inicialmente, quando a amostra é apresentada, todas as saídas dos neurônios de  $F^{(2)}$  estarão produzindo valores iguais a zero. Nesta situação, a camada que estará ativa será  $F^{(1)}$ , pois há a verificação da regra dos dois terços, visto que tanto  $G_1$  como  $G_2$  estarão produzindo valores iguais a 1;
- Uma cópia da amostra é então encaminhada para a camada  $F^{(2)}$ , na qual será definido o neurônio vencedor e que produzirá valor igual a 1 em sua saída, sendo que nos outros serão produzidos valores de resposta iguais a 0. Neste momento, em virtude da saída do neurônio vencedor estar produzindo valor igual a 1, o parâmetro  $G_1$  passa então a valer 0 {similar à aplicação de (10.4) e (10.5)};
- O fluxo de execução retorna para  $F^{(1)}$ . Em seguida, somente aqueles neurônios de  $F^{(1)}$  que possuírem componentes de  $x$  iguais a 1, assim como também receberam valores iguais a 1 a partir das saídas ponderadas dos neurônios de  $F^{(2)}$ , serão ativados (produzindo valores iguais a 1 em suas saídas). Este padrão casado  $\{v\}$  será então enviado para o teste de similaridade;

- Na condição em que as quantidades de componentes com valores iguais a 1, que integram tanto  $v$  como  $x$ , forem bem próximas (maiores que o parâmetro de vigilância), então o valor do *Reset* será 0, havendo consequentemente a atualização das matrizes  $W^v$  e  $W^x$  (Similar à aplicação de (10.6), (10.9) e (10.10)). Caso contrário, o parâmetro *Reset* assumirá valor igual a 1, indicando que o atual neurônio vencedor deverá ser desabilitado.

Portanto, constata-se que esses quatro passos operativos estão também integralmente implementados no algoritmo apresentado na seção anterior, cujos três parâmetros de controle estão implícitos nas condições impostas para as estruturas de comparação e repetição.

Conforme pôde ser ainda verificado no decorrer deste capítulo, uma das potencialidades da rede ART-1 é a sua flexibilidade em incorporar novos conhecimentos, sem, contudo destruir o que já foi aprendido. Esta qualidade distintiva a transforma em uma das melhores arquiteturas de redes neurais artificiais que consegue tratar, de maneira coerente e sistemática, o dilema estabilidade/plasticidade. Complementarmente, algumas outras características de destaque são (Carpenter & Grossberg, 1987a):

- A arquitetura ART possui uma forte plausibilidade biológica lastreada pelos princípios da normalização de sinal, intensificação de contraste e memória de curto prazo;
- O treinamento da rede é sempre estável, sendo que, após o alcance desta estabilização (convergência), a aplicação de um padrão que se encaixa em uma das categorias já criadas ativará de forma direta o respectivo neurônio representativo daquele agrupamento, sem a necessidade de acionamento da fase de busca. Neste caso, a rede acaba funcionando como uma memória associativa autônoma;
- O processo de seleção do neurônio vencedor na fase de reconhecimento é também sempre estável. Uma vez que um vetor de entradas já foi associado ao agrupamento representado por um daqueles neurônios da camada  $F^v$ , esta mesma unidade sempre vencerá a competição, independentemente de eventuais ajustes posteriores que sejam realizados em  $W^v$  e  $W^x$ , por ocasião da apresentação de outras amostras de treinamento;
- A ocorrência do estado de ressonância adaptativa depende, principal-

mente, do qual próximo está à amostra apresentada frente ao vetor representativo do cluster, o que é indicado pelo neurônio vencedor. Se a distância for aceitável, ponderada pelo parâmetro de vigilância ( $p$ ), então haverá o estado de ressonância adaptativa que, em termos biológicos, corresponde à amplificação e ao prolongamento da atividade neural;

• O nível de detalhamento de cada nova classe inserida dentro da estrutura da rede fica, portanto, em função do valor especificado para o parâmetro de vigilância. Quanto maior este valor, mais detalhes ponderados, assim como mais características diferenciais a respeito dos padrões a serem inseridos nas classes, serão então considerados.

## 10.7 – Exercícios

- 1) Enumere as principais características envolvidas com as camadas  $F^0$  e  $F^1$  da rede ART-1.
- 2) Explique quais as diferenças existentes, do ponto de vista de plasticidade, entre as arquiteturas de Kohonen e ART-1.
- 3) Determine o que é o princípio de ressonância adaptativa e como pode ser avaliado nas redes ART-1.
- 4) Explique de maneira resumida como se processa o mecanismo de aquisição de novos conhecimentos pela rede ART-1.
- 5) Discorra sobre o dilema plasticidade/estabilidade quando se avalia a rede Paraplex multicamadas.
- 6) Explique quais as estruturas de dados computacionais que podem ser utilizadas visando implementar a adição ou a habilitação de novos neurônios na camada de saída da rede ART-1.
- 7) Discorra sobre qual seria a principal finalidade do parâmetro de vigilância da rede ART-1.
- 8) Considerando a resposta formulada para a questão anterior, explique quais as implicações em se assumir o valor unitário para o parâmetro de vigilância.
- 9) Seja um vetor de entradas dado por  $\mathbf{x} = [1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0]^T$ . Considerando que o neurônio de índice  $k$  seja o vencedor da camada  $F^0$ , tem-se então os seguintes elementos associados aos pesos dos neurônios da camada  $F^1$ :

$$W_{i,k}^{(2)} = [1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0]^T$$

Discorra se o referido padrão estaria aprovado no teste de similaridade quando se assume um valor de 0,8 para o parâmetro de vigilância.

10) Em relação ao exercício anterior, qual seria o valor mínimo a ser assumido para o parâmetro de vigilância a fim de que o padrão seja aprovado no teste de similaridade.

### 10.8 - Projeto prático

O comportamento de um processo industrial pode ser analisado levando-se em consideração diversas variáveis de *status* relativas às fases do processo. Na tabela 10.3 são fornecidas 10 situações do comportamento do sistema a partir dos valores coletados de 16 variáveis de *status* ( $x_1, x_2, \dots, x_{16}$ ).

Tabela 10.3 – Conjunto de treinamento para a rede ART-1

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$	$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$	$x_{15}$	$x_{16}$
Situação 1	0	1	0	1	1	0	1	0	1	1	0	1	1	1	1	1
Situação 2	1	0	1	0	1	1	1	1	1	1	1	0	1	0	0	0
Situação 3	1	0	1	1	1	1	1	0	1	1	0	1	1	0	1	1
Situação 4	1	1	1	0	1	0	1	0	1	1	1	1	0	1	0	0
Situação 5	0	0	1	1	1	1	1	1	0	1	1	0	0	0	0	1
Situação 6	1	1	0	1	0	0	1	0	1	1	0	1	1	1	1	1
Situação 7	1	0	1	0	1	1	0	1	1	1	1	0	1	1	1	0
Situação 8	1	0	1	1	1	1	1	0	1	1	0	1	1	0	1	1
Situação 9	0	1	1	0	1	0	1	0	1	1	0	1	0	1	0	1
Situação 10	0	0	1	1	1	1	1	1	0	1	1	0	0	0	0	1

Assim, o objetivo do projeto é implementar e treinar uma rede ART-1, que classifique e agrupe em classes as situações que são “parecidas”, de modo que se tenha um provável diagnóstico em uma eventual necessidade de manutenção.

Em sequência, proceder com as simulações que visam a classificação dessas variáveis de *status*, considerando-se aqui os seguintes valores para o parâmetro de vigilância: i)  $\rho = 0,5$ ; ii)  $\rho = 0,8$ ; iii)  $\rho = 0,9$ ; iv)  $\rho = 0,99$ .

Finalmente, após cada simulação, indicar a quantidade de classes que ficarão ativas pelos seus neurônios representantes, listando-se ainda quais as situações da tabela 10.3 que estarião incorporando em seus respectivos agrupamentos.



Richard Wesley Hamming

## **Parte II**

**Aplicações de redes neurais artificiais em problemas  
de engenharia e ciências aplicadas**



## **Estimação da qualidade global de café utilizando o Perceptron multicamadas**

### **11.1 – Introdução**

O alvo desta aplicação é empregar redes neurais artificiais para classificar lotes, ou marcas de café, a partir de um conjunto de sensores à base de polímeros condutores, os quais foram desenvolvidos pela Embrapa Instrumentação Agropecuária de São Carlos. O conjunto, conhecido como “língua eletrônica”, é ilustrado na figura 11.1.



Figura 11.1 – Conjunto “língua eletrônica”

A utilização deste dispositivo possibilita a caracterização da qualidade do café analisado, que é algo tradicionalmente realizado por degustadores profissionais na forma de atribuição de notas de 1 a 10.

A língua eletrônica realiza medições de reatância capacitiva nas amostras (em meio líquido) por meio de 55 frequências distintas. Mediante o uso de redes neurais artificiais, deseja-se então extrair, a partir desse conjunto de informações, todas as características que possam servir de subsídios para a criação de um modelo capaz de atribuir notas às marcas ou lotes de café.

Portanto, este problema consiste basicamente em aproximar uma função, cujo domínio é definido pelas frequências advindas da operação dos sensores da língua eletrônica, que fornecerá como resposta uma medida de caracterização da qualidade do café, a qual será tabulada em uma escala de 1 até 10. Tais valores são referentes às notas atribuídas pelos degustadores especializados, sendo que a nota 10 reflete o patamar máximo de qualidade.

## 11.2 – Características da Rede PMC

Os sinais de entrada foram normalizados para um melhor condicionamento das saídas. A RNA empregada nesta aplicação foi um *Perceptron* multicamadas (PMC).

Diversas topologias para o PMC foram testadas, chegando-se finalmente às duas candidatas com bons resultados, isto é, topologia T1 (com uma camada escondida) e topologia T2 (com duas camadas escondidas), cujas configurações são as seguintes:

### Topologia T1

- Camada de entrada → 55 sinais de entrada, referentes às medições de capacitâncias em 55 diferentes frequências;
- Camada neural escondida → 28 neurônios;
- Camada neural de saída → 1 neurônio.

### Topologia T2

- Camada de entrada → 55 sinais de entrada, referentes às medições de capacitâncias em 55 diferentes frequências;

- 1<sup>a</sup> Camada neural escondida → 10 neurônios;
- 2<sup>a</sup> Camada neural escondida → 2 neurônios;
- Camada neural de saída → 1 neurônio.

O algoritmo de treinamento empregado na fase de aprendizagem foi o *backpropagation*. Utilizou-se, em primeira instância, a validação cruzada com um conjunto de dados formado por 236 exemplos, dos quais 90% para o treinamento e 10% para a validação.

Em seguida, aplicou-se a validação cruzada com  $k$ -partições a fim de aumentar a capacidade de generalização da rede. As amostras foram divididas em 10 subconjuntos, sendo seis conjuntos com 24 amostras e quatro com 23 amostras.

A taxa de aprendizado empregada foi de 0,1. O número de iterações especificado para a convergência da rede foi de 500 épocas. As topologias testadas podem ser mais bem compreendidas por meio das figuras 11.2 e 11.3.

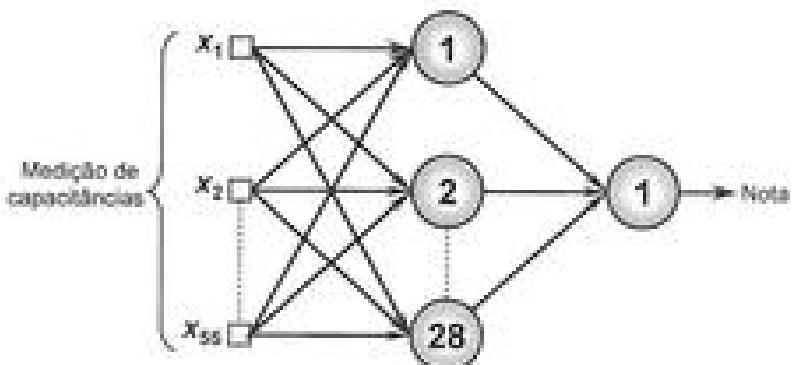


Figura 11.2 – Ilustração da topologia T1 com uma camada escondida

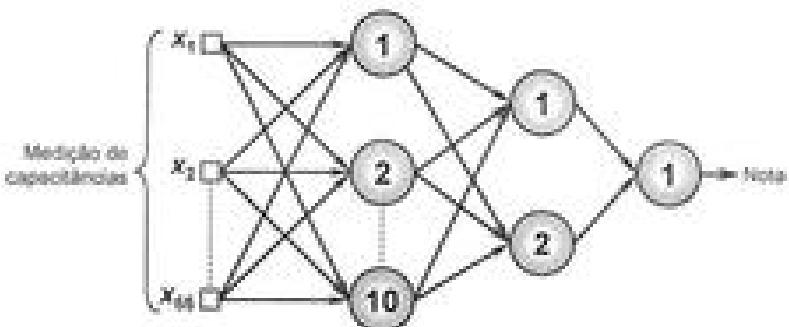


Figura 11.3 – Ilustração da topologia T2 com duas camadas escondidas

### 11.3 – Resultados computacionais

Os fatores de correlação ( $r^2$ ) entre as saídas desejadas e aquelas obtidas após os treinamentos, considerando-se aqui os conjuntos de validação, são apresentados na tabela 11.1.

Tabela 11.1 – Resultados dos treinamentos

Método de validação	Fator de correlação ( $r^2$ ) (topologia T1)	Fator de correlação ( $r^2$ ) (topologia T2)
Validação cruzada convencional	0,8437	0,8648
Validação cruzada com $k$ -partições	0,7998	0,8184

Na figura 11.4 é ilustrada uma comparação entre a aproximação funcional realizada pela topologia T1 em relação aos valores desejados.

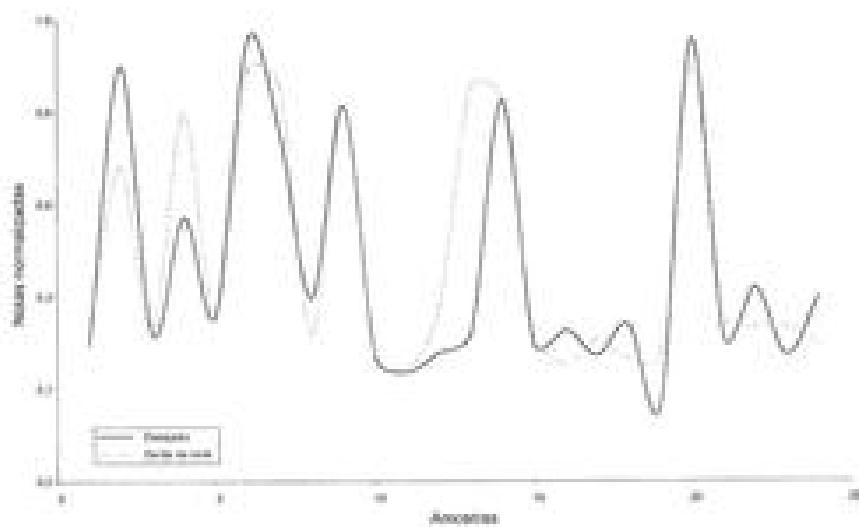


Figura 11.4 – Resultados comparativos (topologia T1)

Já na figura 11.5 é também possível visualizar um comparativo entre a reconstrução da função por meio da aplicação da topologia T2 em relação aos valores desejados.

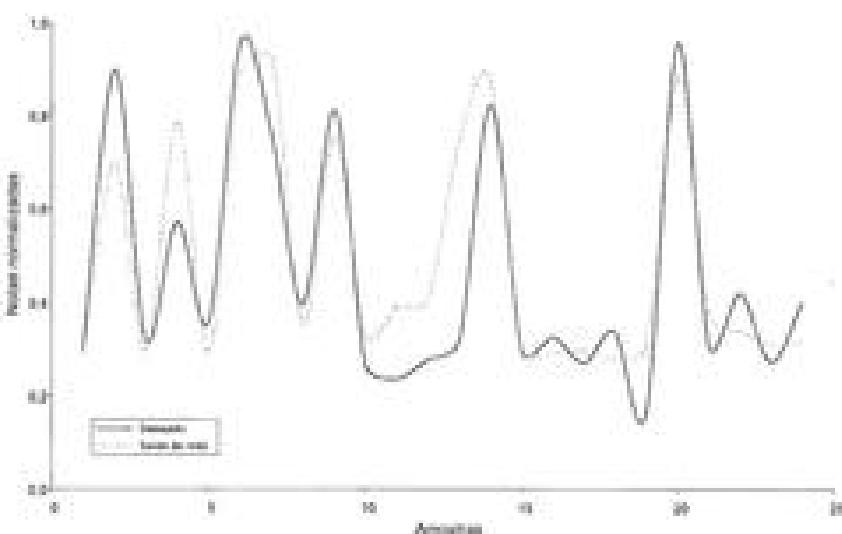
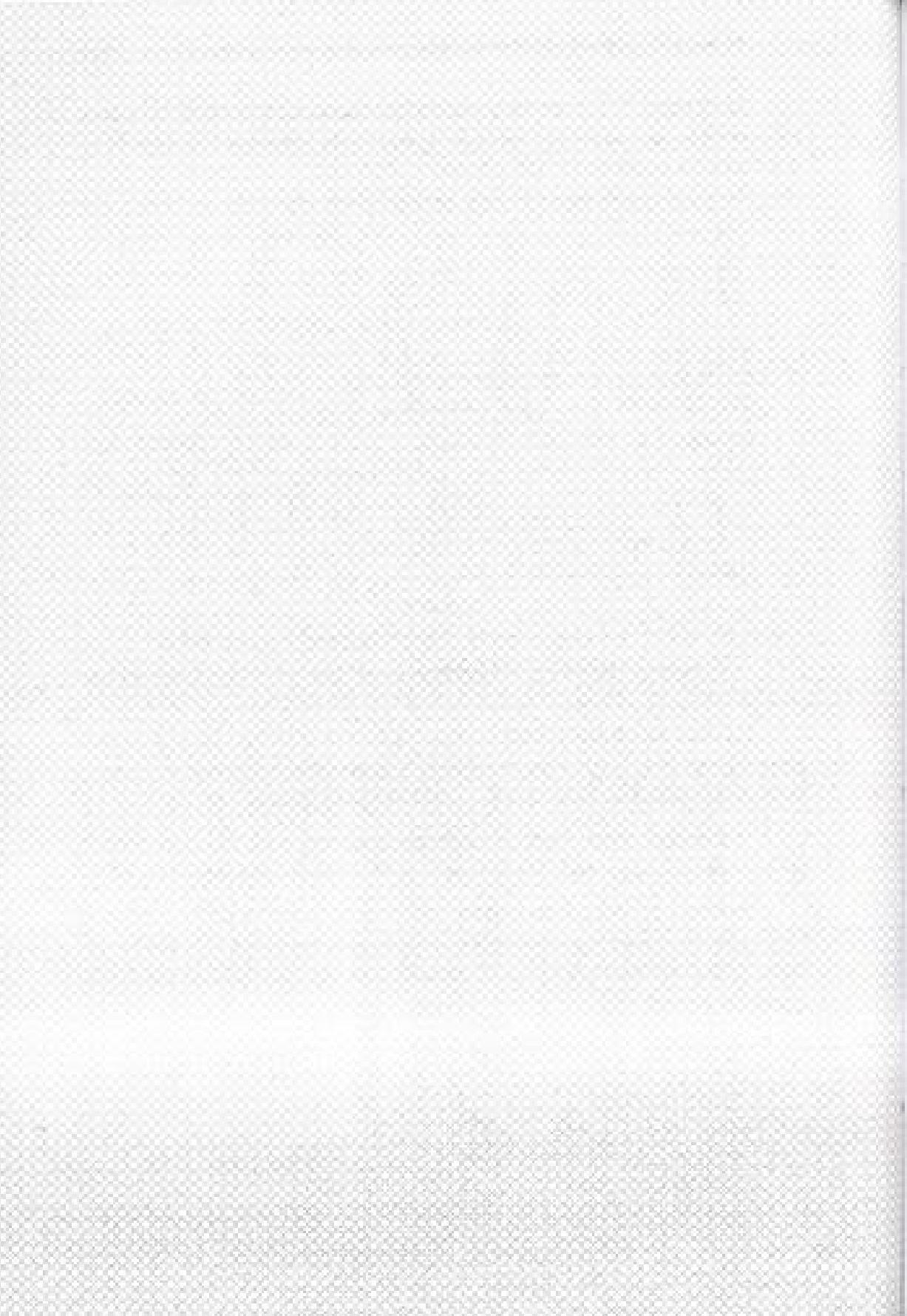


Figura 11.5 – Resultados comparativos (topologia T2)

As duas topologias testadas foram capazes de manter o fator de correlação ( $r^2$ ) próximo a 0,8. Este resultado é considerado promissor se comparado às atribuições de notas por degustadores profissionais.

A topologia que apresentou melhores resultados foi a T1 e, por possuir apenas uma camada escondida, denotou melhor eficiência computacional.

Assim, o modelo desenvolvido mostrou-se capaz de desempenhar a função de estimador, em conjunto com a "língua eletrônica", podendo auxiliar os degustadores no processo de atribuição de notas para avaliação da qualidade de café.



# Análise de tráfego de redes de computadores utilizando protocolo SNMP e rede LVQ

## 12.1 – Introdução

A investigação do fluxo de dados de uma rede de computadores é fundamental para o planejamento da expansão do sistema, assim como para a resolução de problemas.

A principal dificuldade no gerenciamento de uma rede de computadores é analisar os resultados disponibilizados por ferramentas de monitoramento de tráfego, pois os dados não se encontram de forma intuitiva, exigindo a inspeção de um especialista.

Os testes envolvendo o tráfego de informações em uma rede de computadores podem ser realizados sob três formas de análise:

- Das transmissões de dados feitas na rede inteira ou, em determinado segmento, por uma interface em modo promíscuo;
- Do tráfego de uma determinada interface, pertencente tanto a um *host* (dispositivo ou equipamento) como a um computador, que é suspeita de inundação de tráfego de rede;
- Das estatísticas de tráfego dos dispositivos de rede ou dos servidores presentes no segmento.

Já o *SNMP* (*simple network management protocol*) é um protocolo de gerenciamento de rede que, atuando sobre a camada de aplicação do modelo *OSI* [Kurose & Ross, 2006], efetua um monitoramento ativo da rede, além de fornecer serviços de configuração de equipamentos.

As informações são obtidas com o *SNMP* por intermédio de uma requisição de um gerente a um ou mais agentes, como pode ser examinado na figura 12.1.



Figura 12.1 – Relacionamento entre gerente e agente baseado no modelo TCP/IP

Cada máquina na rede é representada como um conjunto de variáveis referentes ao seu estado atual. Tais informações ficam disponíveis ao gerente, que também pode alterá-las. Cada máquina gerenciada deve possuir um agente e uma base de dados *MIB* (*management information base*).

O agente é um processo em execução na máquina gerenciada, enquanto que o gerente é um processo em execução no servidor, que permite a comunicação com diversos agentes gerenciados, conforme pode ser observado na figura 12.2.

O gerenciamento do *link* de internet, que fornece a banda de transmissão, geralmente é feito de maneira visual por meio de gráficos fornecidos por ferramentas especializadas, como ilustrado pela figura 12.3.

Assim, a classificação da rede em um dado momento deve ser executada por um especialista, podendo gerar atrasos na tomada de decisão no caso de haver alguma falha humana de interpretação.

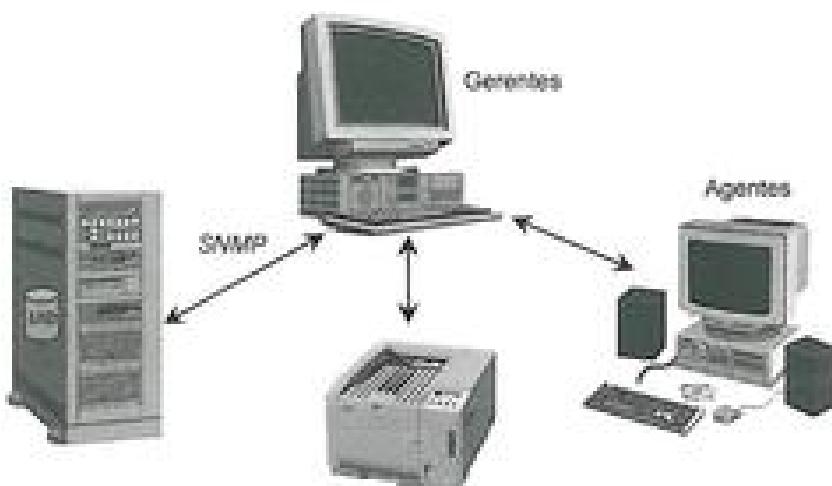


Figura 12.2 – Mecanismo de gerenciamento de comunicação entre dispositivos

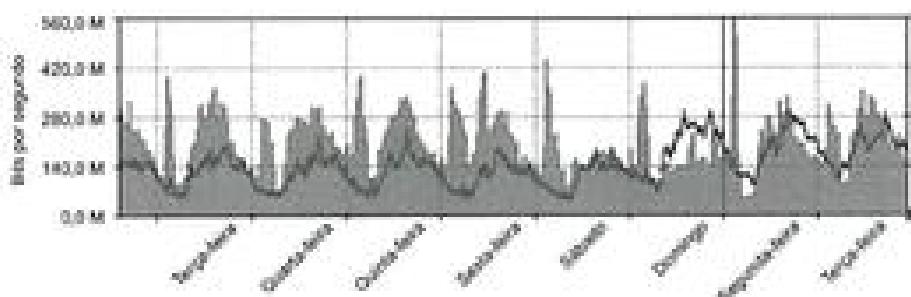


Figura 12.3 – Exemplo de um gráfico de tráfego

Para deixar a classificação das redes de computadores de maneira automática, investigou-se aqui a aplicação de redes *LLQ* no processo de classificação dos dados.

## 12.2 – Características da rede LVQ

A classificação automática do tráfego de rede será realizada por uma *LLQ*-1. Na figura 12.4 visualiza-se o comportamento da rede analisada para o período de uma semana, em que o tráfego de saída de bits da rede se manteve abaixo de 200 kbps (kilobit por segundo). Para todos os estudos envolvendo o tráfego de redes, analisou-se aqui uma única interface ethernet em um servidor.

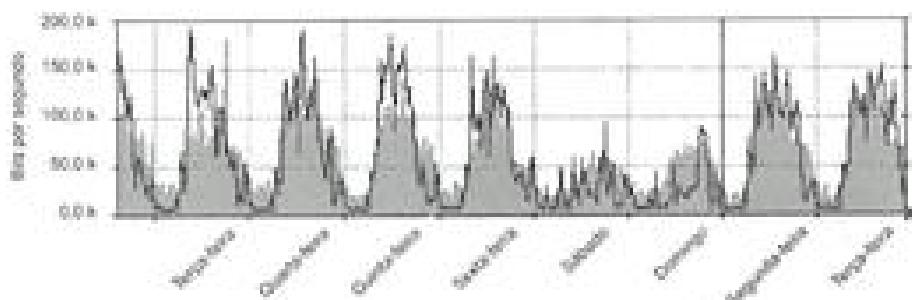


Figura 12.4 – Visualização do comportamento de rede analisada

A topologia de rede  $LJ/Q\text{-}1$  foi constituída de quatro neurônios quantizadores para representar as seguintes classes de tráfego:

*Classe A* → Tráfego de até 43 kbps;

*Classe B* → Tráfego entre 43 kbps e 86 kbps;

*Classe C* → Tráfego entre 86 kbps e 129 kbps;

*Classe D* → Tráfego acima de 129 kbps.

A figura 12.5 ilustra a estrutura da rede  $LJ/Q\text{-}1$ , em que estão representadas em suas saídas as respectivas classes de tráfego.

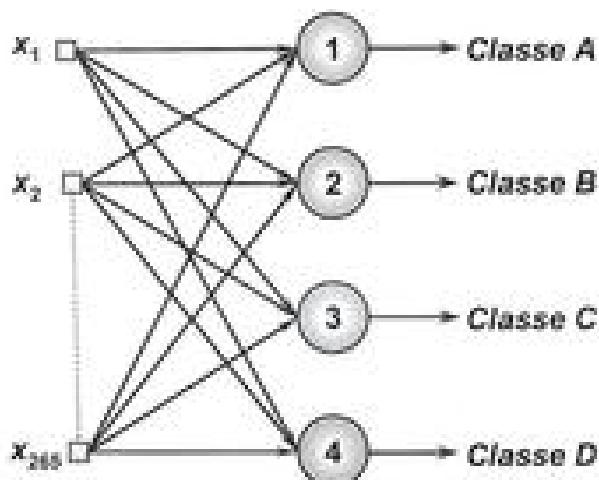


Figura 12.5 – Topologia da rede  $LJ/Q\text{-}1$  para a classificação de tráfego

Um total de 36 amostras colhidas em laboratório foi empregado para o treinamento, as quais representam nove situações de comportamento de tráfego em relação a cada uma das classes previamente definidas. A captura de bits da saída da rede de computadores foi realizada a cada cinco minutos, durante um intervalo de 24 horas, totalizando assim 265 sinais de entrada para a rede  $LLQ$ -1.

### 12.3 – Resultados computacionais

Utilizaram-se 10 treinamentos diferentes ( $t1, t2, \dots, t10$ ) e, em cada um, foram gerados novos pesos na rede. Com base nos acertos obtidos foi então possível construir um gráfico com a performance dos treinamentos, para cada dia da semana, conforme exibido na figura 12.6.

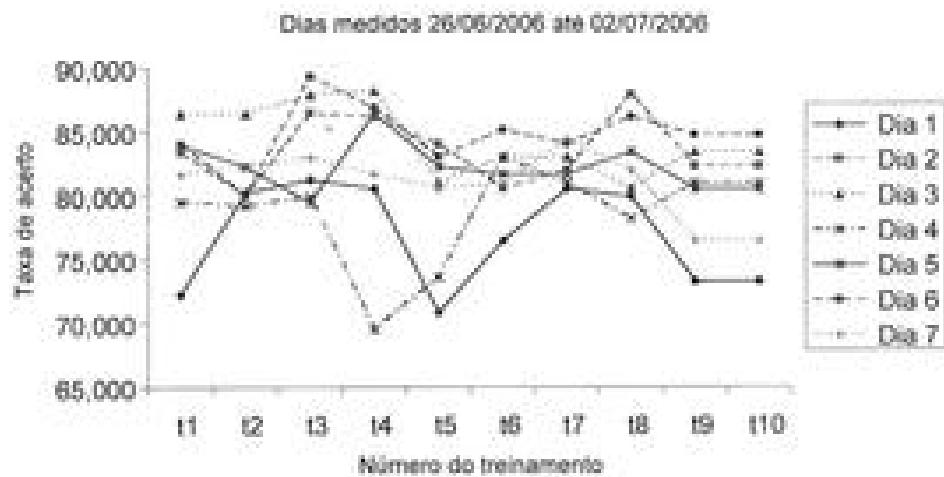


Figura 12.6 – Performance durante sete dias.

Já para o teste de classificação foram utilizados 288 padrões, colhidos também em intervalos de cinco minutos cada.

Observa-se uma taxa de acertos média de 81,57%, sendo 69,44% o valor mínimo e 89,23% o valor máximo. Na figura 12.7 contata-se a aplicabilidade da  $LLQ$  em interpretar corretamente as classes de tráfegos de rede, fornecendo-se aqui um resultado com relatório da classe atual para o usuário.

Análise do tráfego de redes de computadores  
através do protocolo SNMP<sup>3</sup> e redes neurais artificiais

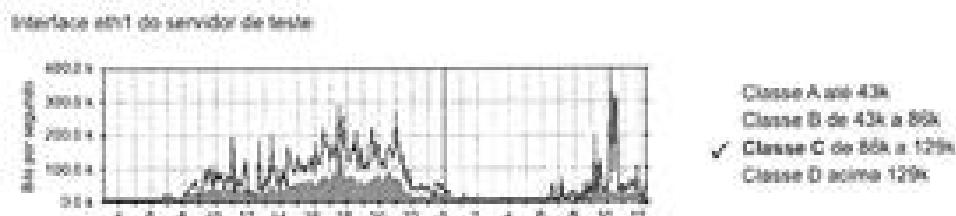


Figura 12.7 – Visualização da implementação proposta

Para esta aplicação, a grande vantagem de se utilizar uma *LVQ* para a classificação de redes foi o seu baixo custo computacional, pois os resultados são compilados em tempos muito curtos (menor que meio segundo).

Adicionalmente, as taxas de acertos alcançadas pela abordagem proposta são também consideradas bem favoráveis para o propósito de análise de tráfego em redes de computadores.

# Previsão de tendências do mercado de ações utilizando redes recorrentes

## 13.1 – Introdução

Métodos convencionais de predição do comportamento de títulos financeiros fundamentam-se na análise e tomada de decisão de especialistas, indispensável para a maioria das situações de métodos automáticos capazes de realizar tal tarefa.

Dentro deste contexto, a utilização de redes neurais artificiais recorrentes se constitui como uma alternativa à tomada de decisão no mercado acionário de títulos financeiros.

Esta aplicação irá contribuir com as ferramentas já existentes para a tomada de decisão do mercado acionário, uma vez que irá analisar a influência de variáveis macroeconómicas.

No mercado de capitais existem dois modelos que regem as estratégias de compras e vendas de ações. O modelo fundamentalista analisa os aspectos económicos e contábeis das empresas, bem como a influência de variáveis macroeconómicas. Já o modelo técnico se fundamenta em dados históricos a respeito da empresa, tentando assim inferir o futuro a partir de comportamentos passados.

Os dois modelos não são mutuamente exclusivos, mas sim complementares, sendo que qualquer decisão no mercado de ações é tomada avaliando-se as duas vertentes de estudo de uma empresa.

Diversos tipos de ações são colocados à venda no mercado financeiro pelas empresas, tais como a PN e a ON. A PN é um tipo de ação que dá direito na forma de recebimento do lucro da empresa. Já a ON dá direito de voto nas assembleias da empresa.

Desta forma, o histórico de preço de uma determinada ação produz uma sequência numérica, cuja previsão de comportamento é de fundamental importância para negociações futuras. Na figura 13.1 observa-se o comportamento de ações do grupo Pão de Açúcar no ano de 2005.

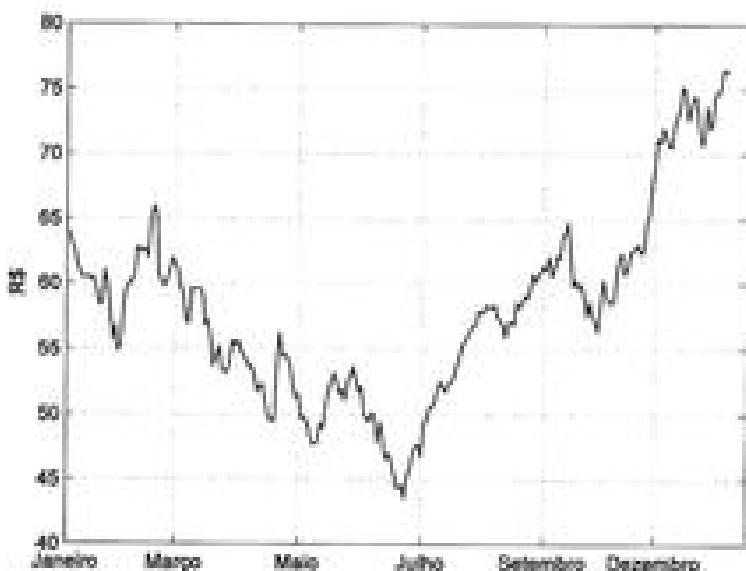


Figura 13.1 – Gráfico de evolução do preço das ações do grupo Pão de Açúcar no ano de 2005

A partir da figura 13.1 fica evidente o comportamento altamente não-linear da série, revelando-se a necessidade da utilização de ferramentas capazes de tratar este tipo de problema de maneira automática e eficiente.

Em consequência, como se deseja avaliar a cotação das ações para períodos futuros, tem-se então que a arquitetura neural mais apropriada seja aquela com configuração recorrente que, a partir de dados históricos, seja capaz de estimar vários passos (à frente) no horizonte futuro.

### 13.2 – Características da rede recorrente

A rede neural utilizada aqui para estimativa dos valores de ações é uma rede recorrente (simples), também denominada de Elman (subseção 5.4.3), semelhante àquela mostrada na figura 13.2.

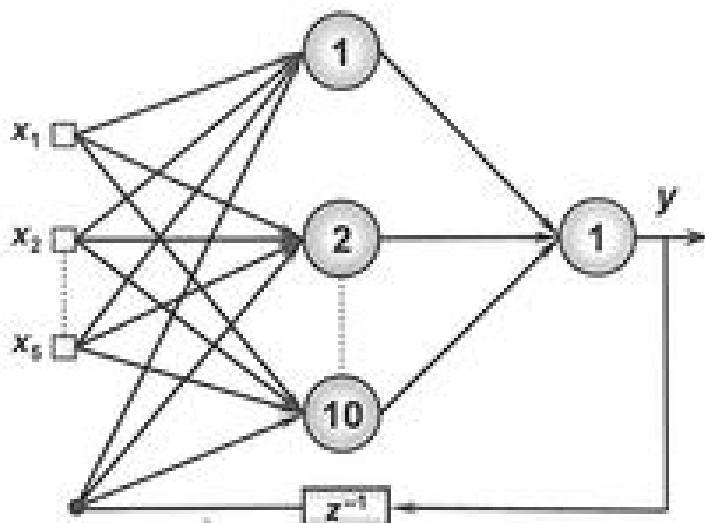


Figura 13.2 – Ilustração de rede neural recorrente

A topologia da rede neural utilizada é descrita como se segue:

- Camada de entrada → cinco entradas, sendo o preço de abertura ( $x_1$ ), o máximo preço ( $x_2$ ), o mínimo preço ( $x_3$ ), o preço de fechamento ( $x_4$ ) e o volume comercializado ( $x_5$ );
- Camada neural escondida → 10 neurônios;
- Camada neural de saída → 1 neurônio, representando o preço de fechamento das ações para dias futuros.

Também foi feita uma avaliação da previsão levando-se em conta não só os fatores históricos, mas também as variáveis econômicas que influenciam no preço das ações. Neste caso, testou-se uma segunda rede neural que tomasse em consideração tais variáveis, cuja topologia está configurada como segue:

Camada de entrada → oito entradas, sendo o preço de abertura, o pre-

o máximo, o preço mínimo, o preço de fechamento, o volume comercializado, a taxa de juros de curto prazo, o índice de produção industrial e o índice Bovespa;

Camada neural escondida → 10 neurônios;

Camada neural de saída → 1 neurônio, representando o preço de fechamento das ações para dias futuros.

### 13.3 – Resultados computacionais

Para validar a eficiência das redes neurais artificiais desenvolvidas foram realizados três estudos de casos, envolvendo o comportamento de ações das empresas do Grupo Pão de Açúcar, Grupo Itausa e Embraer.

Na figura 13.3 pode-se contemplar os resultados de estimativa das ações para o Grupo Pão de Açúcar, levando-se em consideração apenas os dados históricos de preço das ações.

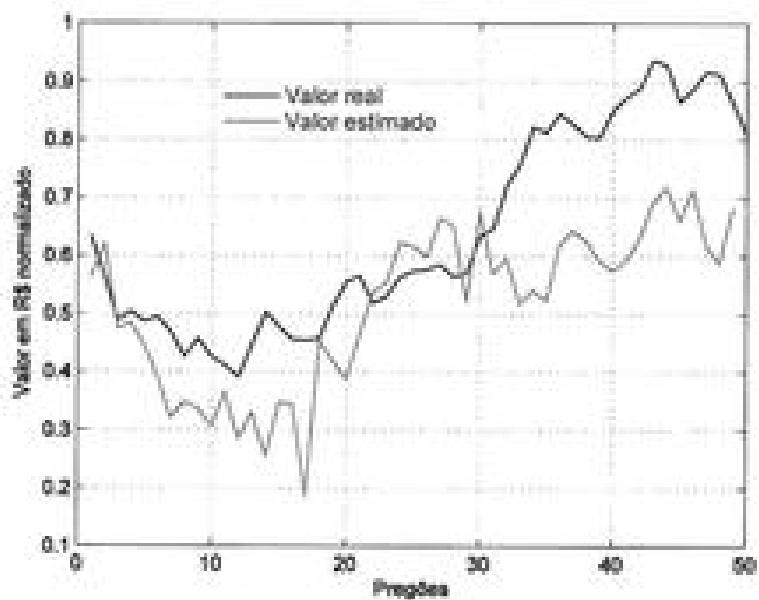


Figura 13.3 – Estimativa de preços do Grupo Pão de Açúcar utilizando-se dados históricos

Já na figura 13.4 encontram-se registrados os resultados para as ações do Grupo Pão de Açúcar, levando-se também em consideração as três variáveis macroeconómicas descritas anteriormente. A previsão apresentou uma melhoria de 85% em relação à utilização apenas das variáveis históricas.

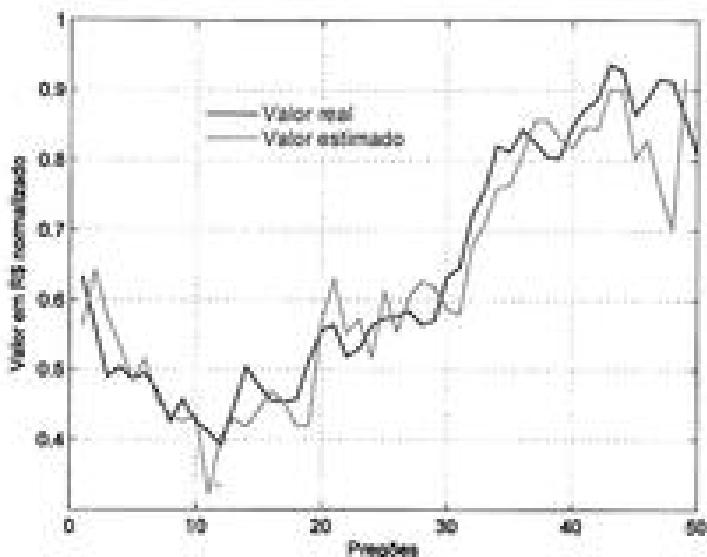


Figura 13.4 – Estimação de preços do Grupo Pão de Açúcar utilizando-se dados históricos e variáveis económicas

Os resultados utilizando-se as variáveis económicas foram significativamente melhores; no entanto, o uso da previsão de longo prazo tornou-se inapropriada para este caso, uma vez que os erros se autopropagam à medida em que se aumenta o horizonte de previsão.

Nas figuras 13.5 e 13.6 encontram-se os resultados para o Grupo Itau, contando com a análise da previsão do preço das ações com e sem a influência das variáveis económicas.

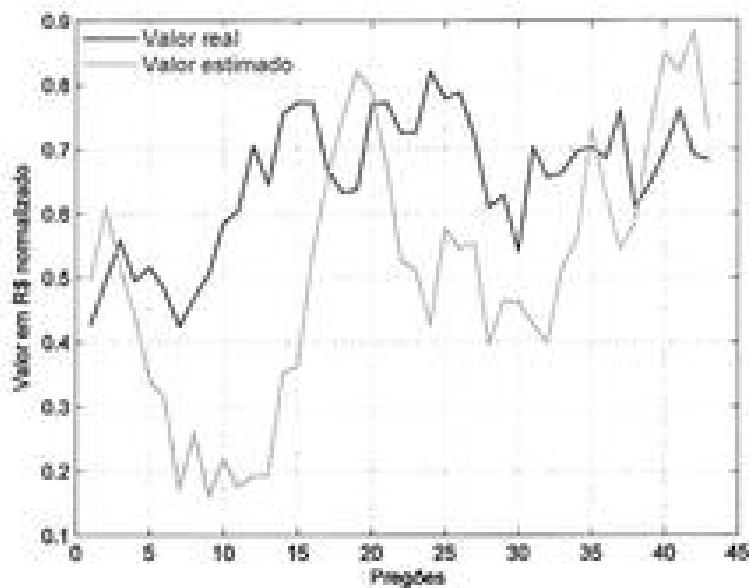


Figura 13.5 – Estimação de preços do Grupo Itaua utilizando-se dados históricos

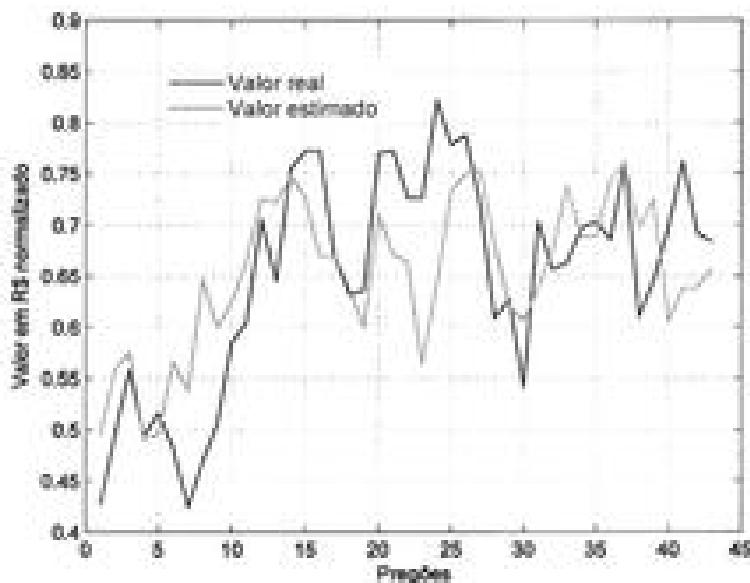


Figura 13.6 – Estimação de preços do Grupo Itaua utilizando-se dados históricos e variáveis econômicas

Mais uma vez constata-se um melhor desempenho na estimativa quando da presença das variáveis econômicas na rede neural recorrente.

Nas figuras 13.7 e 13.8 encontram-se os resultados de estimativa para a Embraer.

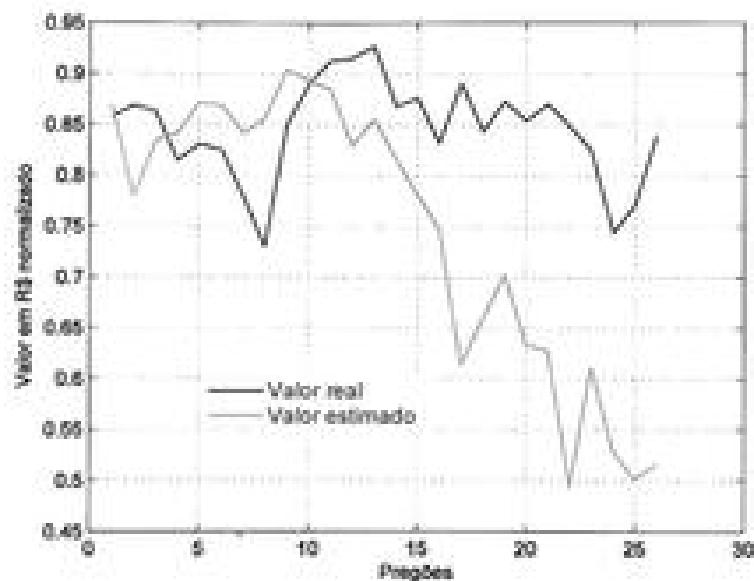


Figura 13.7 – Estimação de preços da Embraer utilizando-se dados históricos

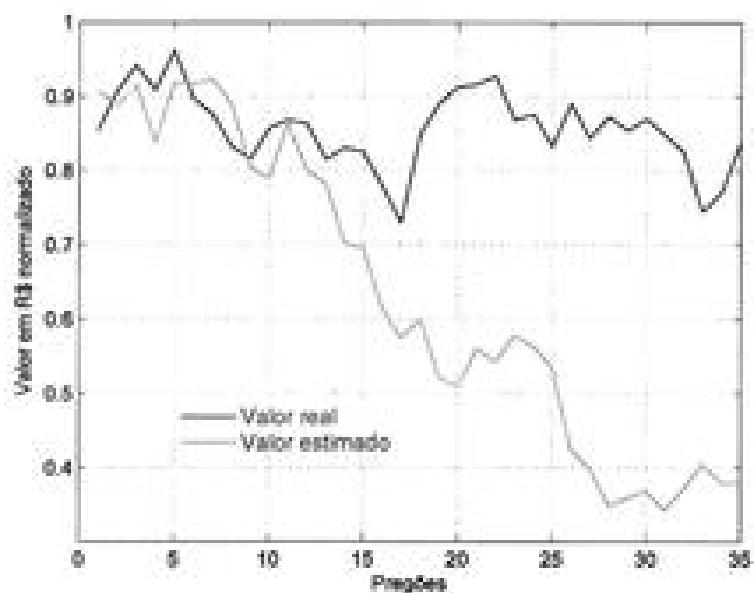


Figura 13.8 – Estimação de preços da Embraer utilizando-se dados históricos e variáveis econômicas

Como podem ser examinadas nas figuras 13.7 e 13.8, o emprego das variáveis econômicas adicionadas à rede recorrente pouco influenciou na melhora dos resultados para a estimativa de preços das ações da Embraer.

Tal comportamento decorre do fato de esta empresa atuar no ramo aeronáutico, destinado sobretudo ao mercado externo, sendo que as variáveis econômicas internas pouco afetam seu desempenho.

Assim, para a Embraer, utilizou-se uma base de dados maior para o treinamento, resultando em uma melhora significativa nos resultados de estimativa, conforme ilustra o gráfico da figura 13.9.

Os resultados promissores obtidos mostraram a eficiência das redes neurais recorrentes em prever séries temporais com comportamento não-linear, como aquelas do mercado de ações.

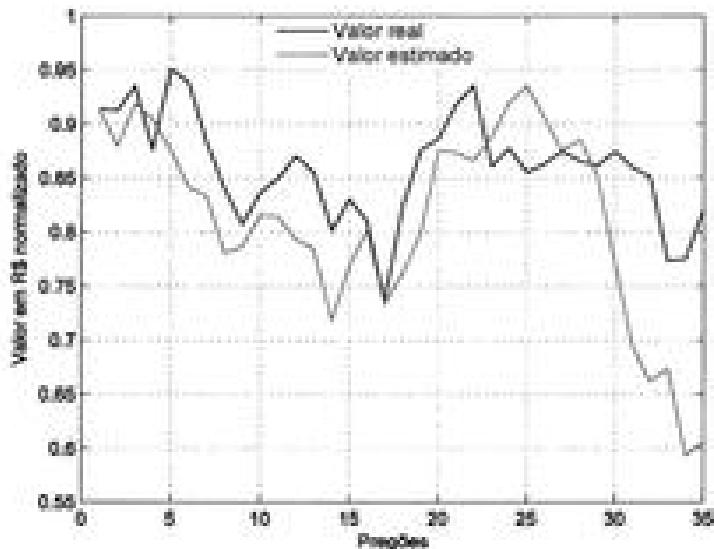


Figura 13.9 – Estimativa de preços das ações da Embraer com a base de dados incrementada.

A utilização das variáveis econômicas possibilitou uma melhora substancial nos resultados de estimativa, principalmente em ações cujas empresas dependem fortemente dos índices financeiros nacionais.

Já para empresas cujo mercado é regulado por influências internacionais (externas), como o mercado aeronáutico, tais variáveis econômicas pouco interferem na qualidade da estimativa, sendo necessário o estudo de outros índices capazes de sensibilizar melhor as redes neurais artificiais.

## Sistema para diagnóstico de doenças utilizando redes ART

### 14.1 – Introdução

A especificação de tratamento adequado para pessoas portadoras de distúrbios pode ser muito complexa devido à sua diversidade. Para tanto, apresenta-se neste capítulo uma aplicação de redes neurais artificiais com o objetivo de auxiliar na identificação de um diagnóstico mais preciso acerca de tais doenças.

Segundo a CID-10 (Classificação Estatística Internacional de Doenças e Problemas Relacionados à Saúde), os distúrbios estão divididos em 21 classes como se segue:

- 1) Doenças infecciosas e parasitárias;
- 2) Neoplasias (tumores);
- 3) Doenças do sangue e dos órgãos hematopoéticos e alguns transtornos imunitários;
- 4) Doenças endócrinas, nutricionais e metabólicas;
- 5) Transtornos mentais e comportamentais;
- 6) Doenças do sistema nervoso;
- 7) Doenças do olho e anexos;

- 8) Doenças do ouvido e da apófise mastoíde;
- 9) Doenças do aparelho circulatório;
- 10) Doenças do aparelho respiratório;
- 11) Doenças do aparelho digestivo;
- 12) Doenças da pele e do tecido subcutâneo;
- 13) Doenças do sistema osteomuscular e do tecido conjuntivo;
- 14) Doenças do aparelho geniturinário;
- 15) Gravidez, parto e puerpério;
- 16) Afecções originadas no período perinatal;
- 17) Malformações congénitas, deformidades e anomalias cromossómicas.
- 18) Sintomas, sinais e achados anormais de exames clínicos e de laboratório, não classificados em outra parte;
- 19) Lesões, envenenamento e algumas outras consequências de causas externas;
- 20) Causas externas de morbidade e de mortalidade;
- 21) Fatores que influenciam o estado de saúde e o contato com os serviços de saúde.

A presença de dois ou mais problemas apresentados nesta lista dificulta a escolha do(s) tratamento(s) a ser(em) indicado(s) ao paciente. Portanto, para  $n$  pacientes, dependendo das doenças presentes, pode-se necessitar de diversos tipos de tratamentos. Para tal propósito, projeta-se agora um sistema fundamentado em redes neurais artificiais do tipo *ART (adaptive resonance theory)*, conforme apresentada no capítulo 10, visando auxiliar na identificação de quantas classes de tratamento são necessárias para um determinado número de pacientes.

Conforme ilustrado na figura 14.1, por meio do conhecimento das deficiências presentes nos  $n$  pacientes, torna-se então possível indicar quantas classes de tratamento serão necessárias.

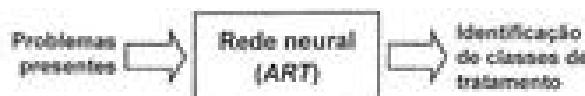


Figura 14.1 – Sistema de apoio ao diagnóstico do paciente

A rede neural ART será utilizada nesta aplicação, pois permitirá uma melhor definição das classes de tratamentos necessárias para um determinado número de pacientes.

## 14.2 – Características da rede ART

A tabela 14.1 representa o índice das classes de doenças, conforme a classificação pela CID, em que foi considerado o valor 1 (um) para presença e o valor 0 (zero) para ausência da doença em determinado paciente.

Tabela 14.1 – Classe representando presença ou ausência de doenças

Pacientes	Doenças							
	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	(...)	D <sub>m</sub>	
1	1	0	0	0	0	(...)	1	
2	1	0	0	1	0	(...)	0	
3	0	0	0	0	0	(...)	0	
4	1	0	1	0	0	(...)	0	
5	1	0	0	0	0	(...)	0	
6	0	0	1	0	0	(...)	0	
7	1	0	0	1	0	(...)	0	
(...)	1	0	0	1	0	(...)	0	
50	1	0	0	1	0	(...)	0	

Por conseguinte, a apresentação destas informações às entradas da rede ART permitirá que esta identifique a quantidade  $m$  de classes de tratamento que serão necessárias para o conjunto de pacientes avaliados. A figura 14.2 ilustra a configuração topológica da rede ART usada nesta aplicação.

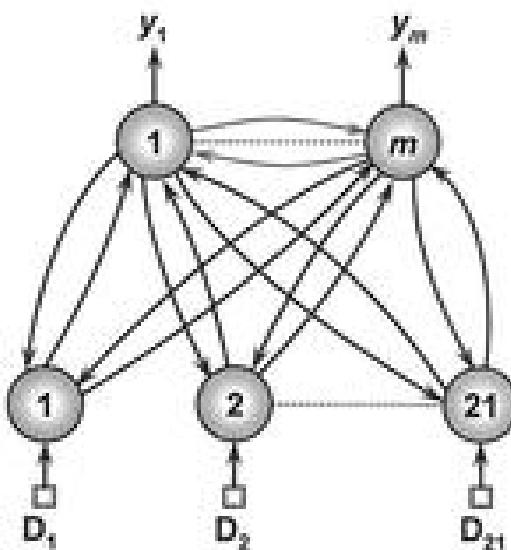


Figura 14.2 – Topologia da rede ART utilizada no diagnóstico de doenças

Consequentemente, após o ajuste de seus parâmetros internos e da respectiva identificação das  $m$  classes de tratamento, a resposta da rede ART indicará em que classe de tratamento o paciente poderá ser enquadrado, a fim de que possa ser encaminhado para o devido tratamento.

### 14.3 – Resultados computacionais

A apresentação das variáveis de entrada da tabela 14.1 à rede ART apresentou como resultado a necessidade de se ter uma quantidade de tipos de tratamentos.

Um conjunto de 50 pacientes classificados pela CID-10, conforme indicado na tabela 14.1, foi apresentado às entradas da rede ART para que esta identificasse a quantidade de classes de tratamento. Foram realizadas quatro simulações visando comparar os resultados quanto às variações nos parâmetros de vigilância da rede.

Após cada simulação, a rede ART indicou quantas classes estavam ativadas e quais situações estavam nos respectivos agrupamentos. Na figura 14.3 é possível conferir os resultados para diferentes parâmetros de vigilância.

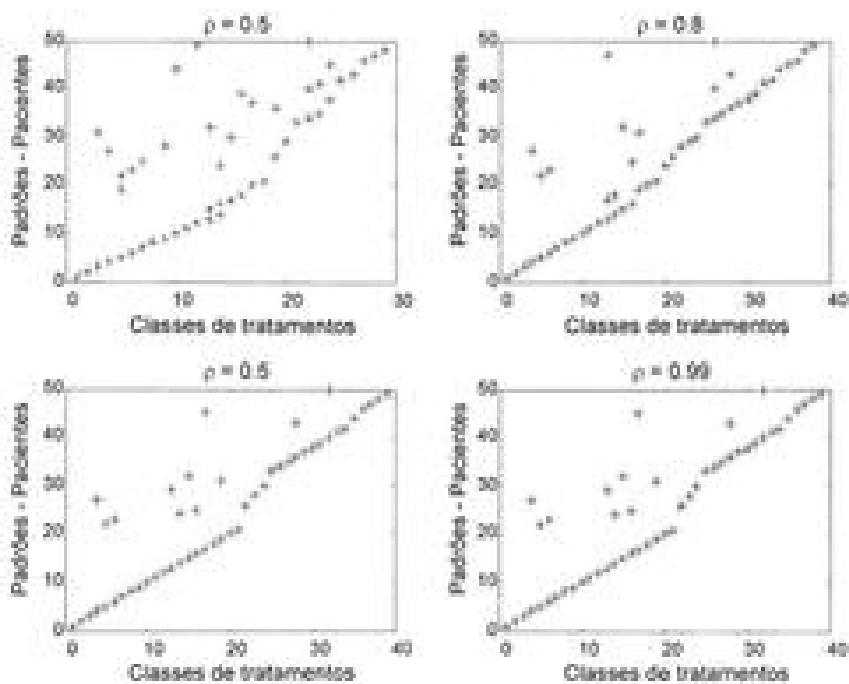


Figura 14.3 – Resposta da rede ART para diferentes parâmetros de vigilância

A figura 14.4 apresenta o número de classes requeridas para os pacientes avaliados pela rede ART, conforme a variação do parâmetro de vigilância.

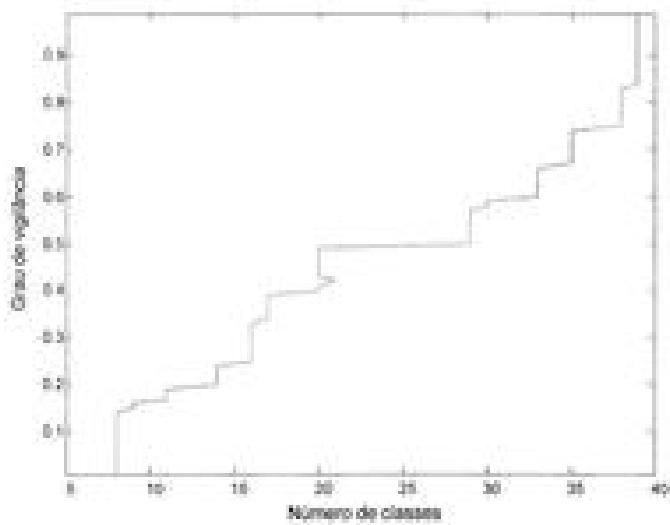


Figura 14.4 – Número de classes em função do grau de vigilância

É possível observar que a rede se apresenta adequada a essa aplicação, pois para um conjunto de valores de entrada tipicamente binários, verifica-se que esta tenta classificá-lo para uma categoria já definida.

Este tipo de aplicação auxilia na definição dos tratamentos a serem utilizados, assim como no encaminhamento do paciente para um provável diagnóstico.

Na figura 14.4 é possível ainda constatar o aumento da quantidade de classes encontradas quando se incrementa o parâmetro de vigilância. Para um parâmetro de vigilância em torno de 0,5, observa-se uma tendência de agrupamento maior das classes, ficando estas também estáveis entre 20 e 30.

O agrupamento dos pacientes em classes pode ser também mais bem examinado na figura 14.5, em que se ilustra uma região de semelhança entre os pacientes, podendo-se assim encaminhá-los para os mesmos grupos de tratamento.

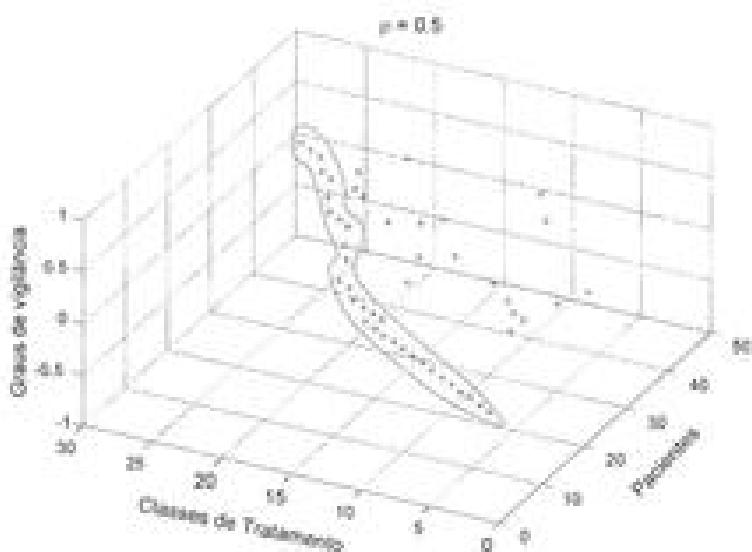


Figura 14.5 – Classes de tratamento para um parâmetro de vigilância de 0,5

A aplicação da rede ART apresentou resultados considerados bem satisfatórios.

Para todos os casos, a rede identificou a quantidade de classes de tratamento necessárias para os pacientes avaliados, podendo então contribuir também para diagnosticar mais precisamente tais doenças.

## Identificação de padrões de adulterantes em pó de café usando mapas de Kohonen

### 15.1 – Introdução

O objectivo desta aplicação é identificar padrões de adulterantes presentes em amostras de pó de café torrado e moído usando-se a rede de Kohonen. As respostas fornecidas pela rede serão confrontadas com os resultados experimentais emitidos pelo instrumento Ali-C (figura 15.1).



Figura 15.1 – Instrumento Ali-C para identificar padrões de adulterantes

O emprego de redes neurais artificiais pode facilitar o processo de detecção de fraudes no pó de café, que pode chegar ao valor de até 85%, sendo que a legislação permite apenas 1% de impurezas ("cascas e paus").

Visando a identificação dos padrões de adulterantes, implementaram-se aqui mapas auto-organizáveis (SOM) de Kohonen. Usou-se um conjunto de dados obtidos com o equipamento denominado de Analisador de Alimentos e Café (Ali-C), desenvolvido pela Embrapa Instrumentação Agropecuária.

Este dispositivo identifica a presença de impurezas pelo princípio fototérmico, em que se aplica uma luz branca com modulação em frequência ou pulso sobre a superfície de um suporte de amostra. Ocorrendo-se a geração de uma onda de calor que atravessa a amostra até atingir um detector pixelétrico, os sinais elétricos obtidos dependem da estrutura (compactação) e da composição do café. Baseado-se neste relacionamento, será possível classificar as amostras.

O processo tradicional é artesanal, isto é, as amostras são submetidas a um tratamento prévio com agentes químicos e, em seguida, a uma inspeção visual com o uso de lentes de aumento (lupa).

Portanto, um método automático para realizar esta identificação permite um maior controle de qualidade sem comprometer a amostra.

## 15.2 – Características da rede de Kohonen

Os dados usados na rede são grandezas físicas como tensão elétrica e grau de compactação. O mapa de Kohonen foi usado para identificar adulteração em amostras de café com impurezas, tais como palha e borna de café, cujo ajuste está como se segue:

- Camada de entrada → Os valores dos atributos de entrada foram normalizados ao intervalo [0;1] para propósitos de seus condicionamentos. Os descriptores considerados foram:
  - Tensão → Definido em um intervalo de variação de 3,0 mV a 5,0 mV, sendo que valores próximos a 3,0 mV são para altas taxas de adulteração do pó de café e 5,0 mV para o café puro;
  - Compactação → Intervalo de variação da compactação de 2,5 a 4,0 mm, conforme a granulometria das amostras;

- Teor de impureza → Considerado como puro o café de origem conhecida e que não apresenta qualquer vestígio de adulterantes. Medido com o equipamento Ali-C cujo limite inferior de detecção é de 5%. O intervalo de impureza pode variar de 0 a 100%. Este parâmetro é usado como referência para as medidas.
- Camada neural de saída → Representa o neurônio vencedor que estará associado a uma das seguintes classes:
  - Classe A → Produto sem adulteração;
  - Classe B → Produto adulterado com palha;
  - Classe C → Produto adulterado com borra de café.

A topologia da rede neural utilizada pode ser mais bem compreendida por meio da figura 15.2.

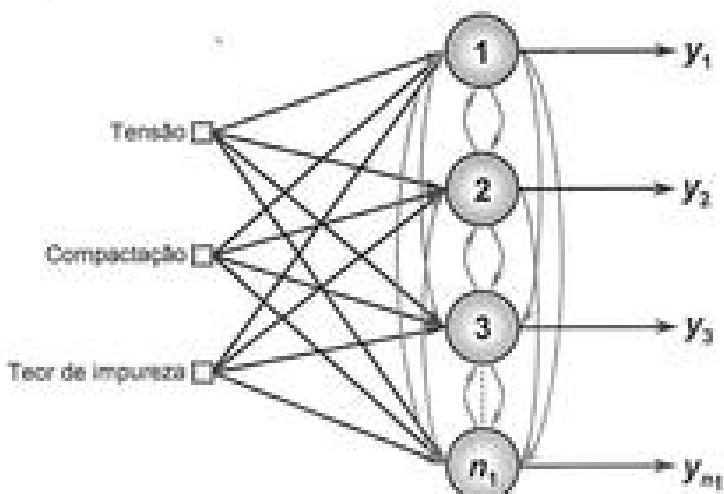


Figura 15.2 – Topologia da rede de Kohonen empregada na classificação

A métrica usada para a avaliação da similaridade entre os padrões, frente à fase de auto-organização do mapa de Kohonen, foi a norma euclidiana. Foram testadas topologias de mapas bidimensionais de 4 x 4, 10 x 10 e 30 x 30 neurônios. A figura 15.3 ilustra a configuração do mapa topológico bidimensional de 4 x 4 neurônios ( $n = 16$ ).

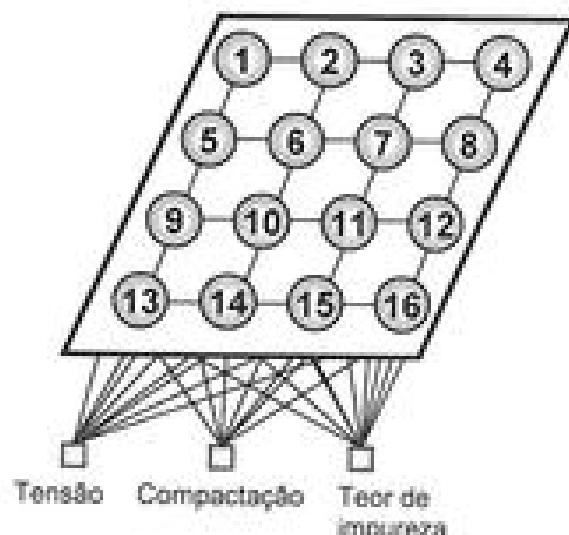


Figura 15.3 – Mapa topológico em grid constituído de 4 x 4 neurônios

A topologia que apresentou resultados mais promissores foi aquela composta de 30 x 30 neurônios, cujo mapa final ilustrado na figura 15.4 exibe a clara distinção dos três *clusters* (A, B e C), correspondentes às classes reais.

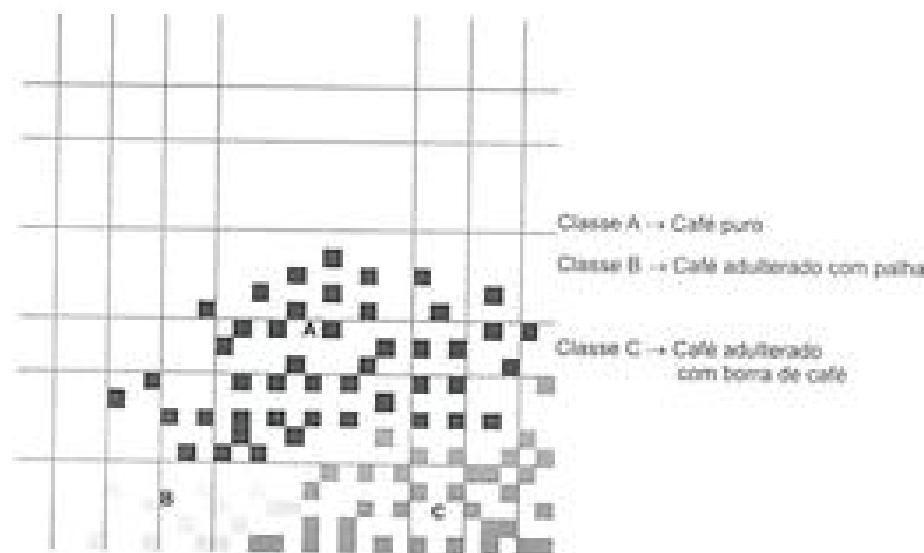


Figura 15.4 – Agrupamento de classes no mapa de Kohonen

Utilizou-se um conjunto de dados com 90 amostras para a fase de ajuste, com taxa de aprendizagem de 0,01 e raio de propagação igual a três vizinhos a fim de delimitar a influência do neurônio vencedor. Para a fase de teste, um total de 10 amostras foi usado, as quais não faziam parte do conjunto de treinamento.

### 15.3 – Resultados computacionais

Os resultados para os *clustres* das amostras são apresentados na tabela 15.1.

Tabela 15.1 – Comparação dos resultados

Amostra	Amplitude média (tensão)	Correção por espessura (compactação)	Teor de pureza	Classes (RNA)	Classes (real/Ali-C)
1	0,444	0,814	1,000	A	A
2	0,678	0,465	0,656	C	C
3	0,554	0,816	0,562	B	B
4	0,885	0,565	1,000	A	A
5	0,728	0,470	0,684	C	C
6	0,551	0,696	0,555	B	B
7	0,674	0,919	1,000	A	A
8	0,610	0,998	1,000	A	A
9	0,631	0,596	0,686	B	B
10	0,788	0,513	0,712	C	C

Da análise visual de posicionamento dos dados de validação, elucidados na tabela anterior, percebe-se que o mapa agrupa corretamente 100% dos exemplos. Ressalta-se que os três *clustres* identificados condizem com as classes verdadeiras (reais).

Em suma, os resultados obtidos pela rede de Kohonen foram considerados bem satisfatórios, pois corroboraram as informações provenientes do Ali-C no que tange à separação correta das amostras em classes.

Padrões referentes a cafés sem adulterantes concentraram-se nas pro-

ximidades de um único neurônio do *cluster A* (figura 15.4). Nota-se que a fronteira deste *cluster* é visivelmente bem delimitada em relação aos *clusters B* e *C*, os quais identificam o café adulterado. Esses agrupamentos dos padrões de cafés com impurezas foram distintos, mas muito próximos, formando-se quase que um único *cluster*.

## Reconhecimento de distúrbios relacionados à qualidade da energia elétrica utilizando redes PMC

### 16.1 – Introdução

A avaliação da Qualidade da Energia Elétrica (QEE) teve um grande salto técnico-científico após a utilização dos sistemas inteligentes. A investigação dos diversos distúrbios relacionados à QEE, por meio de tais sistemas, permite uma identificação mais ampla destes, podendo diferenciar transitórios, variações de curta duração, variações de longa duração, distorções de forma de onda, flutuações de tensão, desequilíbrio de tensão e variação da frequência do sistema.

A ausência de qualidade na energia elétrica fornecida pode resultar em falha ou má operação dos equipamentos dos consumidores, causando grandes prejuízos financeiros. Preocupados com essa questão, os órgãos competentes do setor elétrico brasileiro criaram algumas resoluções com o intuito de avaliar e melhorar a qualidade do fornecimento de energia elétrica. Assim, o correto reconhecimento dos distúrbios que degradam a qualidade da energia permite identificar suas causas. Tendo isso em vista, os indicadores de qualidade de energia elétrica podem ser melhorados quando bem identificados e, consequentemente, solucionados.

A classificação dos distúrbios requer a extração das peculiaridades do sinal elétrico, de forma a identificar a sua classe pertencente, o que tor-

na esse problema bem complexo. A extração por técnicas convencionais como transformada de Fourier, cálculo do valor RMS (*root mean square*) e outros, carecem de fornecerem bons resultados. Assim sendo, a utilização de redes neurais artificiais para esse fim torna-se justificável, pois uma das suas principais aplicações é em reconhecimento de padrões, além do fato de as redes neurais artificiais tratarrem eficientemente diversos tipos de problemas não-lineares.

Neste capítulo é apresentada uma aplicação de redes neurais artificiais para reconhecimento de quatro classes de distúrbios, relacionados à QEE sobre o sinal de tensão. Um distúrbio é caracterizado como qualquer alteração no sinal de tensão ou corrente, que o faz diferente de uma onda senoidal com frequência fundamental (geralmente 50 ou 60 Hz). Neste caso, os distúrbios que serão analisados pela abordagem neural são descritos a seguir.

#### A) Afundamento de tensão

Pode ser definido por qualquer decréscimo na magnitude de tensão que resulte em níveis entre 0,1 e 0,9 p.u. Geralmente, surge na ocorrência de uma falta (curto-círcuito) no sistema elétrico. A figura 16.1 ilustra um exemplo de afundamento de tensão.

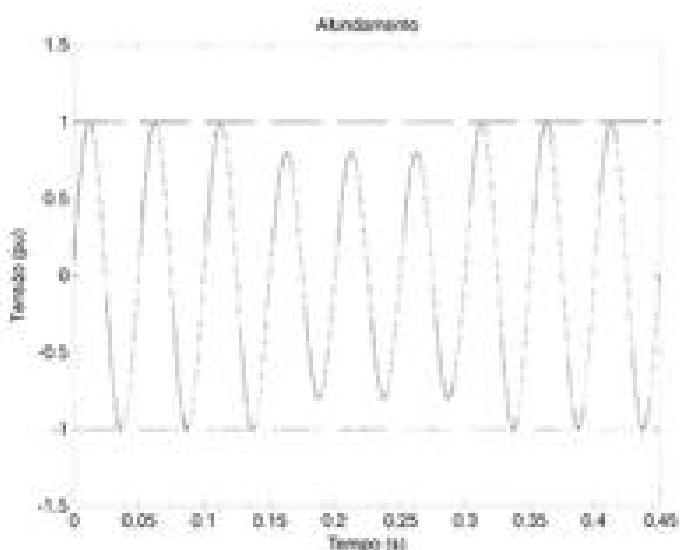


Figura 16.1 – Exemplo de afundamento de tensão

### B) Elevação de tensão

Ocorre com qualquer acréscimo na magnitude de tensão que resulte em níveis entre 1,1 e 1,8 p.u. É provocada também pela ocorrência de faltas. A figura 16.2 exibe um exemplo de elevação de tensão.

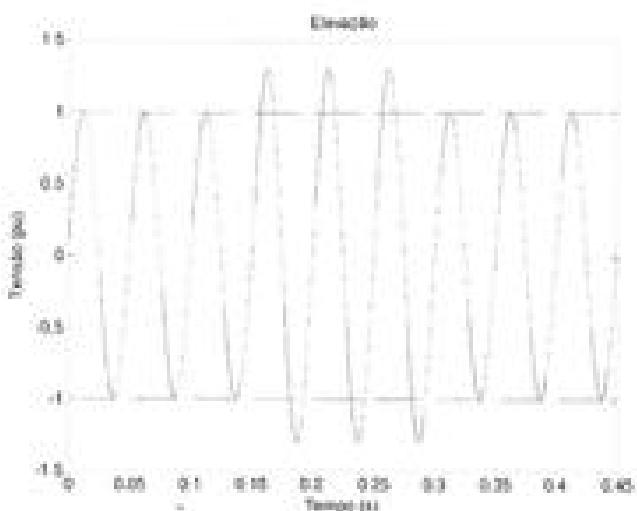


Figura 16.2 – Exemplo de elevação de tensão

### C) Interrupção de tensão

Esta situação é caracterizada por níveis de tensão abaixo de 0,1 p.u. e, geralmente, ocorre quando há falta permanente no sistema elétrico. A figura 16.3 mostra um exemplo de interrupção no sinal de tensão.

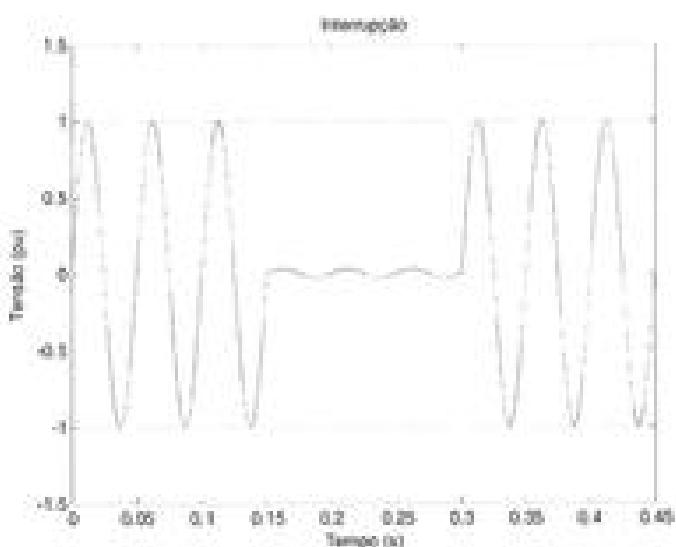


Figura 16.3 – Exemplo de interrupção de tensão

#### D) Distorção harmônica

A distorção harmônica ocorre quando o sinal de tensão é composto por outros espectros de frequência, além da componente fundamental. É provocada por transitórios ou cargas não-lineares da rede elétrica. A figura 16.4 exibe um exemplo de distorção harmônica.

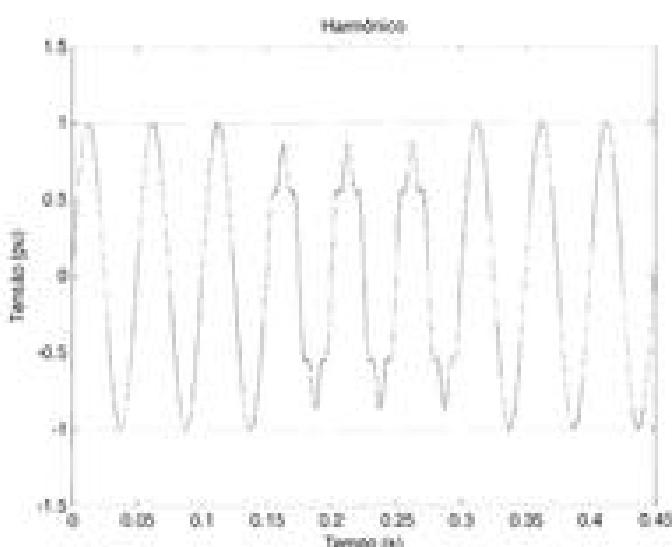


Figura 16.4 – Exemplo de distorção harmônica

A partir desta última figura, nota-se que os três primeiros ciclos representam um sinal de tensão em situação normal de operação, sendo que os três posteriores ilustram um exemplo do distúrbio de distorção harmônica.

## 16.2 – Características da rede PMC

A rede neural utilizada tem como objetivo reconhecer os quatro tipos de distúrbios relacionados à qualidade da energia elétrica, conforme apresentados na seção anterior. Para tanto, utilizou-se uma rede PMC visando classificá-los a partir de amostras de sinais de tensão. Assim sendo, tem-se a seguinte topologia para a rede PMC:

Camada de entrada → 128 entradas, correspondendo à amostragem de meio ciclo do sinal de tensão;

Camada neural escondida → 5, 10, 15 ou 20 neurônios;

Camada neural de saída → 4 neurônios, representando as quatro classes, que correspondem a cada um dos distúrbios analisados.

A topologia da rede neural utilizada pode ser mais bem compreendida por meio da figura 16.5.

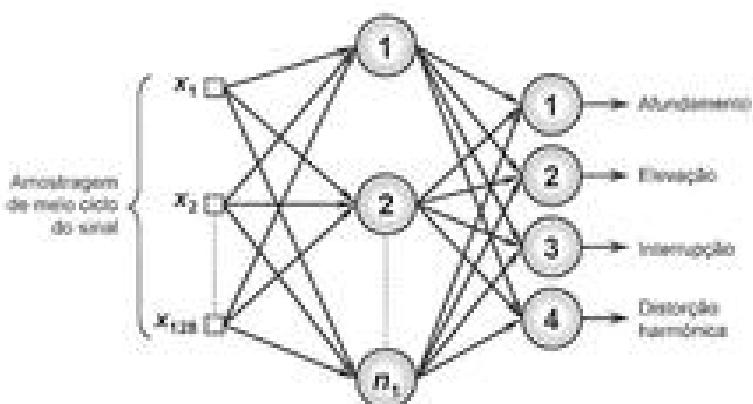


Figura 16.5 – Topologia da rede neural para investigação de distúrbios de QEE

O parâmetro  $n$ , refere-se ao número de neurônios na camada escondida e pode assumir os valores de 5, 10, 15 e 20.

### 16.3 – Resultados computacionais

Foi gerado por meio do programa ATP (*alternating transients program*) um total de 98 situações de distúrbios relacionados à QEE, que são referentes a um sistema de distribuição de energia elétrica. As situações são divididas em 23 casos de afundamento, 29 de elevação, 21 de interrupção e 25 de distorção harmônica. Em seguida, dividiu-se este total (98 situações) em 10 conjuntos com a finalidade de melhor avaliar as fases de treinamento e de teste do PMC.

Para formar cada conjunto dos 10 mencionados, utilizou-se de todos os fenômenos, contribuindo cada um com 10% do seu total para formar os conjuntos de testes e, do restante, com 90%, para os conjuntos de treinamentos.

Para fins de apresentação, escolheu-se o PMC, com o melhor desempenho na fase de treinamento, para empregá-lo nos testes de validação dos resultados que são apresentados a seguir. Quatro topologias de PMC foram investigadas, sendo realizados 10 treinamentos para cada uma.

Todas as topologias testadas apresentaram 100% de acertos, pois, ao se adotar quatro neurônios na camada de saída, conforme destacado na subseção 5.4.1, consegue-se uma maior capacidade de separação das regiões de fronteira. Nesta circunstância, tem-se um neurônio ativo para cada classe,

melhorando-se assim a eficiência da rede e, consequentemente, obtendo-se resultados bem promissores no que tange ao reconhecimento de fenômenos referentes à qualidade da energia elétrica.

Em suma, vale ressaltar que a utilização de 128 amostras (por meio ciclo do sinal de tensão) como entradas do PMC exige uma topologia complexa. Outra estratégia seria a utilização de pré-processamento para reduzir o número de entradas da rede. A figura 16.6 apresenta um esquema simplificado da aplicação do pré-processamento no reconhecimento de distúrbios de QEE por meio do PMC.

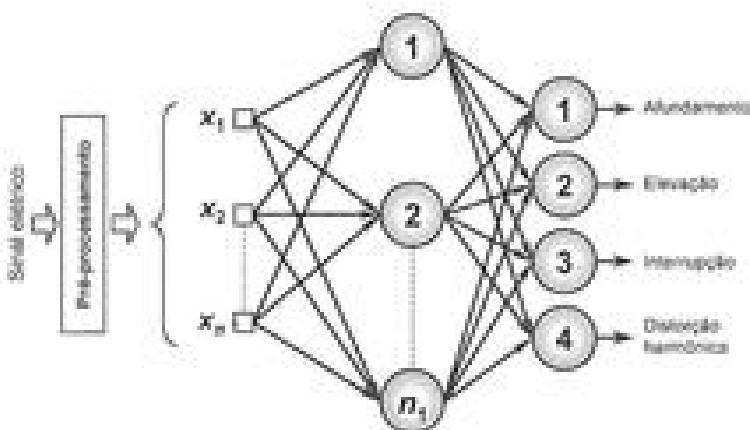


Figura 16.6 – Topologia alternativa para investigação de distúrbios de QEE.

Vale ressaltar que estas investigações envolvendo arquiteturas híbridas, associando técnicas de processamento de sinais e redes neurais artificiais, vêm ultimamente se destacando muito em aplicações de reconhecimento de fenômenos da qualidade de energia elétrica.

Adicionalmente, outras estratégias podem melhorar o desempenho e reduzir a complexidade da rede em aplicações de classificação de problemas relativos à QEE, por exemplo, o uso de redes especialistas para cada classe de distúrbio, conforme mostra a figura 16.7. Com tal esquema seria também possível reconhecer mais de um distúrbio incidente no sinal elétrico. Desta forma, cada uma das redes especialistas se responsabiliza por reconhecer as características particulares referentes a cada distúrbio, sendo que os resultados produzidos são bem promissores na análise de ocorrências simultâneas.

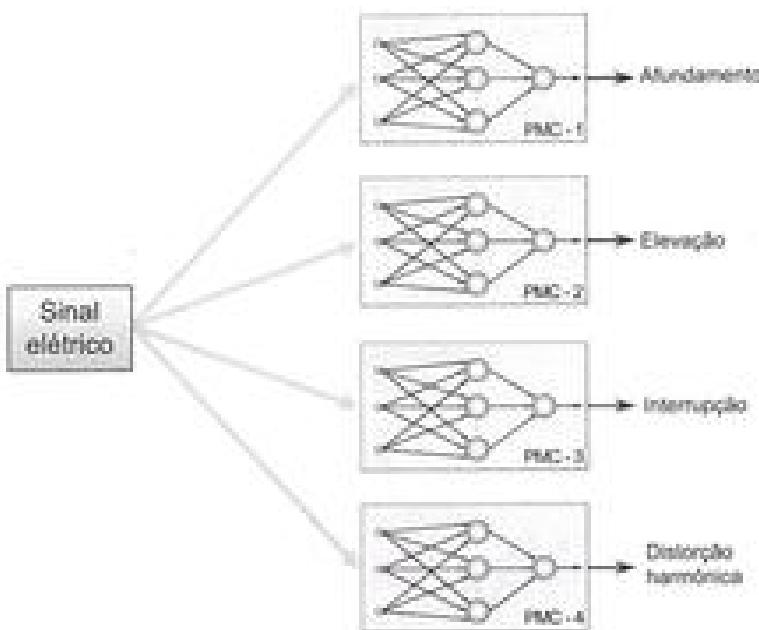


Figura 16.7 – Redes neurais artificiais especializadas para classificação de distúrbios

Em termos comparativos, esta configuração apresentada na figura 16.7 fornece melhores resultados que as outras estratégias, além de redução do esforço computacional utilizado nas fases de treinamento e ajustes das topologias neurais.

## Controle de trajetória de robôs móveis usando sistema *fuzzy* e redes PMC

### 17.1 – Introdução

Diversos métodos de controle são utilizados hoje em dia na automação; porém, nem todos respondem da forma esperada caso a planta de controle não apresente uma resposta linear.

A finalidade desta aplicação é investigar o modelo, analisar os resultados e verificar a eficiência de um controlador, fundamentado em rede neural do tipo PMC, para mapeamento e navegação autônoma. O treinamento da rede será efetuado a partir de uma grande quantidade de amostras que foram geradas por um sistema de inferência *fuzzy*, que foi especialmente projetado para essa aplicação. Os aspectos básicos relacionados aos sistemas de inferência *fuzzy*, com suas principais potencialidades, podem ser encontrados em Pedrycz & Gomide (2007), Buckley & Siler (2004) e Ross (2004).

O sistema de inferência *fuzzy* foi escolhido para gerar os padrões de treinamento da rede a fim de que esta pudesse absorver as características de tomadas de decisão, sendo que esse tipo de problema oferece uma elevada complexidade, e nem sempre é eficaz em uma modelagem matemática com implementação de controle PID analógico ou digital convencional. O robô que irá receber o controlador neural é o *Mesatrs II*, desenvolvido especificamente para ensaios experimentais que envolvem otimização e controle de trajetória, cuja ilustração é apresentada na figura 17.1.



Figura 17.1 – Ilustração do robô *Meautre II*

O *Meautre II* é dotado de sensores de luminosidade e distância. Os sensores de luminosidade foram posicionados estratégicamente para receber radiação luminosa proveniente de todas as direções, sendo essas definidas como *direita*, *esquerda*, *diamante* e *traseira*. O robô deverá seguir por um ambiente de tamanho limitado, que é constituído por objetos colocados em posições diversas e, também, por uma fonte de luz.

A fonte de luz será estrategicamente posicionada, de maneira que se possibilite a recepção da radiação pelos sensores, conforme ilustração da figura 17.2.

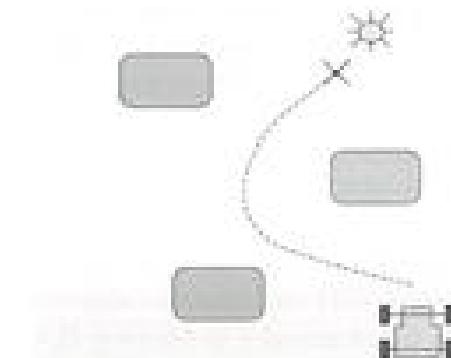


Figura 17.2 – Exemplo de ambiente para navegação do robô móvel

O robô deverá encontrar a fonte luminosa, independentemente de sua posição inicial. Como exemplo, se este estiver posicionado na direção contrária à fonte luminosa, o sensor responsável pela captação *luminos* deverá então informar isto ao controlador neural, que por sua vez fará com que o robô gire e siga em direção àquela fonte.

À medida que o robô esteja já posicionado bem próximo à fonte de luz, um sinal sonoro é emitido com o propósito de indicar o fim da navegação.

Para tanto, o controlador *fuzzy* possui as seguintes características de navegação:

- Controle de posição (direção de navegação);
- Controle de velocidade (ajuste da razão cíclica do controlador PWM (*pulse-width modulation*));
- Indicador de chegada (emissão de sinal sonoro).

Em virtude de limitações de hardware, o controlador neural é implementado em um microcomputador suporte, cujo propósito é receber e processar os dados provenientes do *Mecatron II*, sendo que (em seguida) são compilados os sinais de controle de trajetória, que serão retransmitidos ao robô por meio de comunicação via porta serial.

## 17.2 – Características da rede PMC

O controlador neural será alimentado com as informações provenientes do sistema de inferência *fuzzy*, pois tal medida foi adotada em função da ampla base de dados que tem sido gerada, a partir de situações de controle de trajetória que foram previamente tratadas.

Caso não seja detectado um obstáculo próximo ao robô, e a intensidade luminosa no sensor dianteiro seja também baixa, a velocidade de movimentação do robô será alta, sendo esta controlada de acordo com a relação de distância entre um obstáculo e o robô, ou ainda, de acordo com a intensidade luminosa recebida pelo sensor.

Quando mais próximo de um obstáculo, a decisão de escolher em qual

direção seguir, *direita* ou *esquerda*, será tomada de acordo com a intensidade de luz que atinge os seus respectivos sensores laterais. Por outro lado, quando a intensidade de luz no sensor dianteiro for alta, há a indicação de que o robô já está bem próximo da fonte de luz, sendo que (como consequência) a velocidade tenderá a zero e um sinal sonoro será disparado para indicar a conclusão do percurso.

Como todas as entradas e saídas são normalizadas, os resultados obtidos estão sempre compreendidos no intervalo [0;1]. Portanto, definiram-se, para a saída *Direção*, aqueles valores acima de 0,7 como *direita*, aqueles entre a faixa de 0,7 e 0,3 como *outro*, e outros abaixo de 0,3 como *esquerda*.

O sinal sonoro será emitido quando a saída *Buzina* possuir um valor acima de 0,7, ou quando a saída de controle de *Velocidade* for diretamente aplicado ao controle da razão cíclica do sinal PIPM, que é responsável por controlar o motor do robô. A topologia da rede neural empregada como controlador do *Macatoo II* é apresentada como se segue:

- Camada de entrada → cinco entradas, sendo quatro provenientes de dados recebidos pelos *Sensores de luminosidade* e um sinal do *Sensor de distância (sonar)*;
- Camada neural escondida → 45 neurônios;
- Camada neural de saída → 3 neurônios, representando o controle de *Direção*, *Velocidade* e *Buzina*.

A rede PMC empregada para o controle do *Macatoo II* pode ser mais bem compreendida por meio da figura 17.3.

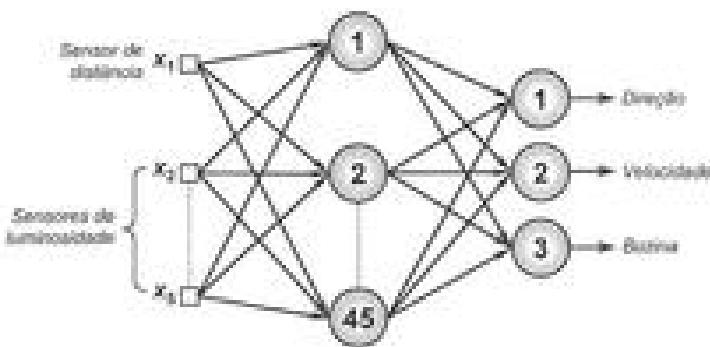


Figura 17.3 – Topologia da rede PMC para o controle do robô *Macatrus II*

Os dados advindos do sistema de inferência *fuzzy* são considerados ideais, portanto, em prática, devem ser também obtidos resultados muito próximos por meio das redes neurais artificiais. Para tanto, foram gerados 700 padrões, contendo as simulações de respostas dos sensores do *Macatrus II*.

Para averiguação do desempenho da aprendizagem neural, o comportamento do Erro Quadrático Médio (EQM) em função das épocas de treinamento pode ser contemplado na figura 17.4.

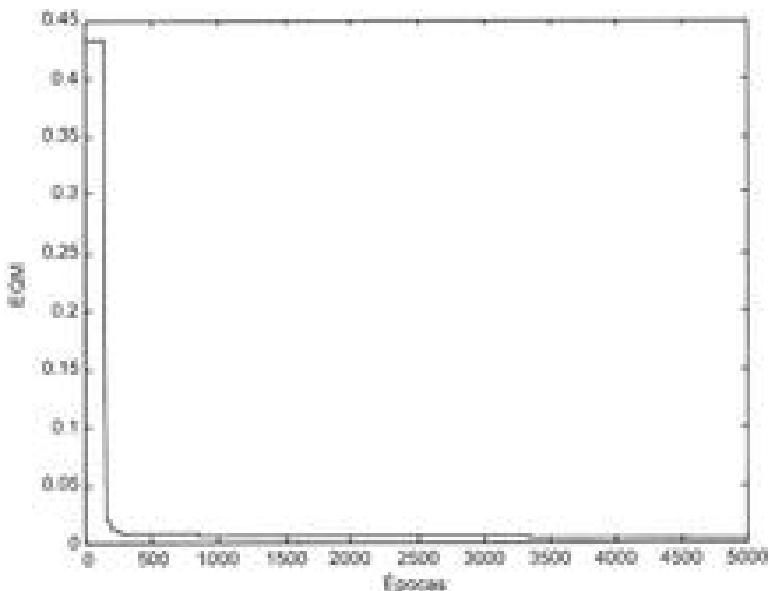


Figura 17.4 – Comportamento do erro quadrático médio em função do número de épocas de treinamento

Observa-se, por meio desta última figura, que o processo de aprendizado da rede PMC é bem rápido, pois este começa a se estabilizar em cerca de 250 épocas de treinamento.

### 17.3 - Resultados computacionais

Depois de treinado o PMC, um conjunto de 50 amostras de valores resultantes da aquisição dos sensores foi apresentado aos dois controladores (*fuzzy* e neural). As figuras 17.5 a 17.7 ilustram os gráficos representativos das respostas obtidas pelo controlador neural frente àquelas esperadas (obtidas pelo controlador *fuzzy*).

Por meio destes três gráficos, tornou-se possível verificar a qualidade do controle proporcionado pela rede PMC em relação ao controlador padrão *fuzzy*.

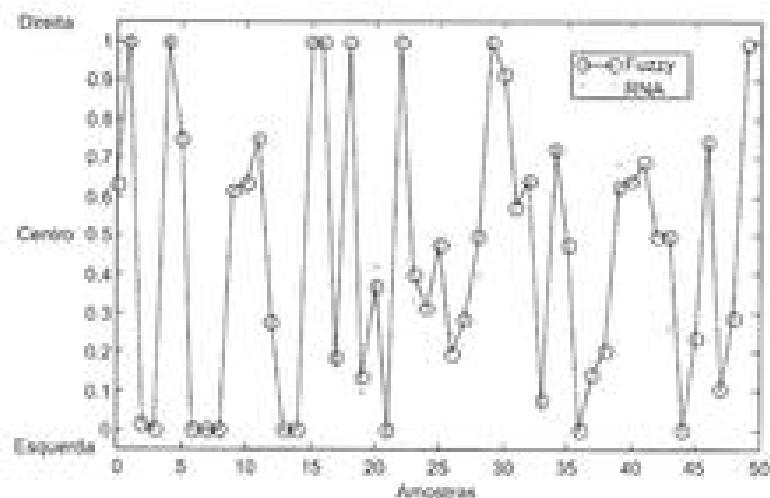


Figura 17.5 – Controles de direção proporcionados pelos controladores neural e *fuzzy*.

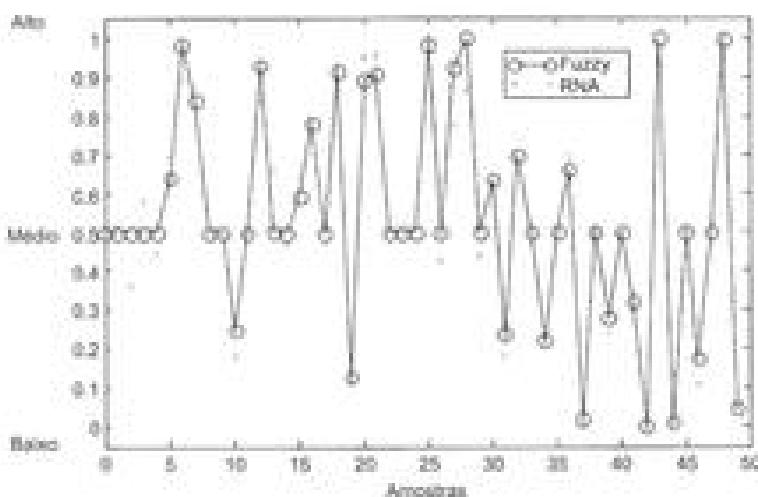


Figura 17.6 – Controles de velocidade proporcionados pelos controladores neural e fuzzy

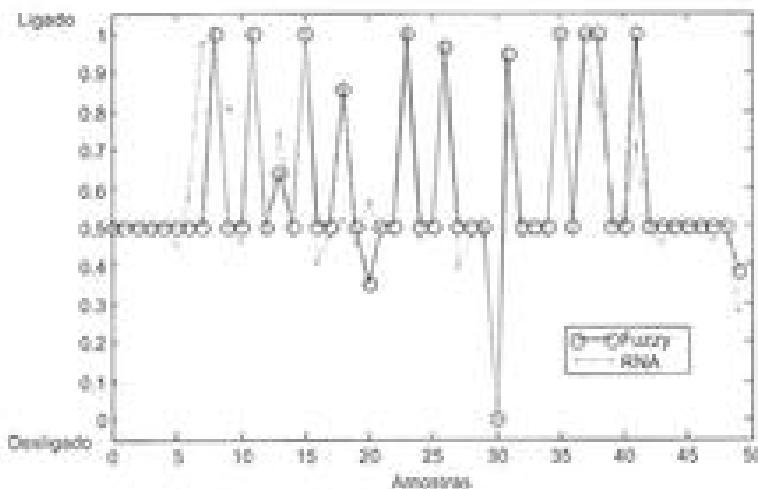


Figura 17.7 – Controles da batina obtidos pelos controladores neural e fuzzy

Para propósitos de acompanhamento do desempenho, uma interface gráfica, ilustrada na figura 17.8, foi também desenvolvida visando o rastreamento *on-line* de todos os parâmetros do controlador neural, inclusive de suas respostas gráficas obtidas em tempo real.

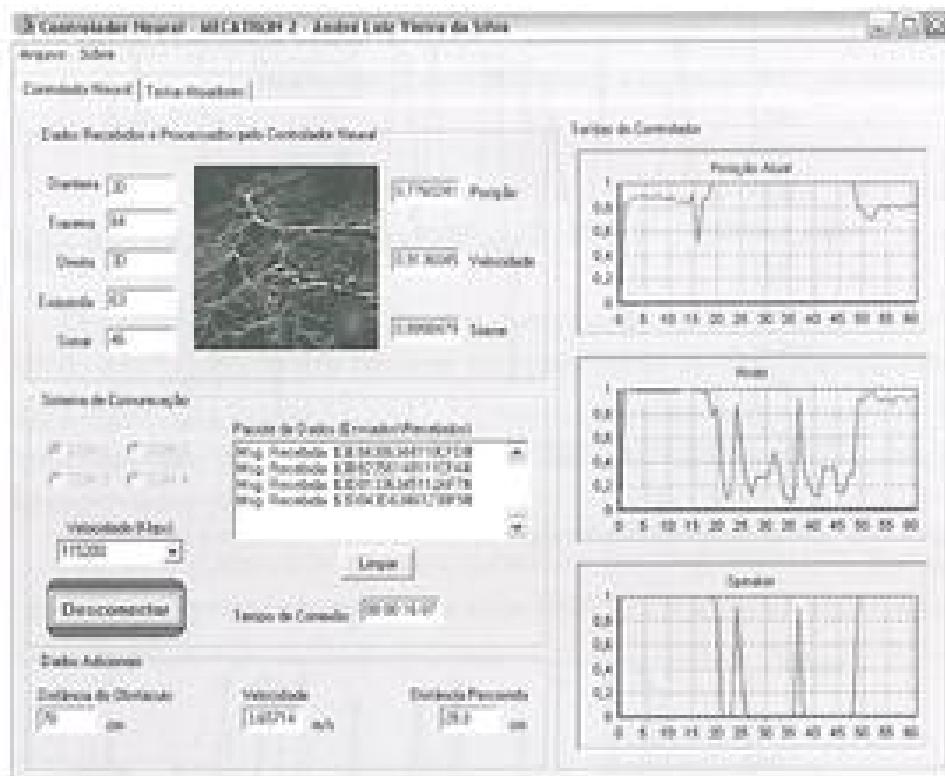


Figura 17.8 – Interface gráfica do controlador neural

Para deixar esta interface gráfica operacional, o hardware é conectado ao barramento serial (RS232), comunicando-se com o software a uma velocidade de 115.200 bps, sendo esta suficiente para a troca de informações de controle sem que haja perda de informações provenientes dos sensores.

Desta forma, torna-se assim possível acompanhar, em tempo real, a decisão proveniente do controlador neural, que recebe as informações oriundas dos sensores do *Mecatrus II*, e que em seguida envia os sinais de controle de volta para o robô.

Ressalta-se que o sistema de controle neural foi implementado em um microcomputador PC, ao invés de inseri-lo no microcontrolador do *Mecatrus II*. Destarte, o sistema de comunicação entre estas duas plataformas transporta os dados dos sensores para o controlador neural, ao passo que leva de volta, para o microcontrolador do robô, os sinais de controle.

Em suma, por meio da análise computacional, observou-se a promissora eficiência do controlador inteligente, baseado em rede PMC, o qual foi capaz de aprender e absorver características de um controlador *fuzzy*, produzindo-se assim respostas eficientes ao sistema de navegação, cujo comportamento entre as variáveis é bem não-linear.

Finalmente, o único pré-requisito para se utilizar o controlador neural é a disponibilização de um banco de dados que possa alimentar o processo de aprendizagem. Para esta aplicação, tal requisito foi suprido pela realização de simulações envolvendo o comportamento dos sensores, por meio de um sistema de inferência *fuzzy*.



## Método para classificação de tomates usando visão computacional e redes PMC

### 18.1 – Introdução

O controle de qualidade, quando realizado de maneira não-automatizada em uma linha de produção, está sujeito a erros cometidos pelos agentes de inspeção que, com o passar do tempo, e também devido ao cansaço, deixam passar (pelos esteiras) produtos impróprios para o consumo.

A maioria dos sistemas automatizados de inspeção é fundamentada na visão humana, simulando-se aqui uma visão artificial com o intuito de realizar a classificação dos produtos. A inspeção por meio da visão computacional privilegia aplicações principalmente na agricultura, em que há uma grande correlação da qualidade do alimento com a sua aparência.

A inspeção automatizada da qualidade de tomates tem como principal motivação a insatisfação dos consumidores, conforme reportado em diversos relatórios técnicos da área. Assim, o emprego de redes neurais artificiais para a classificação de tomates pode deixar o processo de inspeção mais eficiente, melhorando a qualidade dos produtos que chegam até os consumidores.

As amostras dos 102 tomates utilizados para esta aplicação foram adquiridas em supermercado da região de São Carlos, os quais pertencem ao grupo “saladete”, cujo formato padrão é aquele arredondado, com uma cor final avermelhada. A figura 18.1 ilustra alguns tomates deste grupo.



Figura 18.1 – Tomates do grupo “saladeir” em diferentes fases de maturação.

As amostras de tomates obtidas possuíam cores e tamanhos diferentes e foram agrupadas em quatro classes, conforme a tabela 18.1.

Tabela 18.1 – Distribuição de classes de tomates

Classes	Diametro equatorial (cm)	Cor
A	Acima de 7	Completamente vermelho
B	Menor que 7	Completamente vermelho
C	Acima de 7	Manchas esverdeadas
D	Menor que 7	Manchas esverdeadas

O processo de obtenção das imagens dos tomates foi feito por intermédio de uma câmera digital, utilizando-se de um módulo de aquisições, que também possui uma câmera de iluminação difusa, cujo propósito era reduzir ao máximo os reflexos indesejados. O módulo de aquisição pode ser contemplado na figura 18.2.



Figura 18.2 – Módulo de iluminação e de aquisição das imagens

Na figura 18.3 tem-se um exemplo de imagem digital de um tomate, que foi obtida pelo referido módulo.

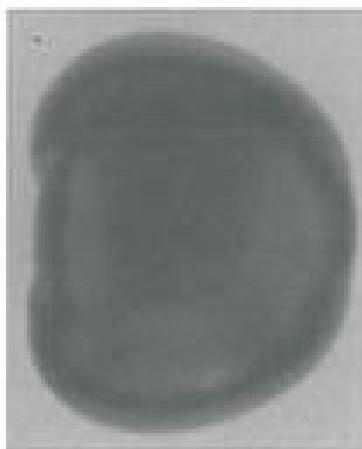


Figura 18.3 – Exemplo de imagem adquirida pelo módulo

## 18.2 – Características da rede neural

A partir das imagens digitais é possível se extrair as características que serão apresentadas à rede neural para o processo de aprendizado. Desse processo, resultam seis variáveis normalizadas, que representam os pixels médios referentes às cores vermelho ( $R$ ), verde ( $G$ ) e azul ( $B$ ), além das coordenadas

cromáticas de vermelho ( $r$ ), verde ( $g$ ) e azul ( $b$ ). As coordenadas cromáticas  $r$ ,  $g$ ,  $b$  são obtidas segundo a expressão (18.1), enquanto os pixels médios ( $R$ ,  $G$ ,  $B$ )<sub>m</sub> são calculados conforme (18.2).

$$(r, g, b) = \frac{\sum_{i=1}^N (R, G, B)_i}{\sum_{i=1}^N R_i + \sum_{i=1}^N G_i + \sum_{i=1}^N B_i} \quad (18.1)$$

$$(R, G, B)_m = \frac{\sum_{i=1}^N (R, G, B)_i}{N \cdot 255} \quad (18.2)$$

onde  $N$  é a quantidade de pixels.

Por meio da varredura dos índices das expressões (18.1) e (18.2), calculam-se todos os pixels médios e todas as coordenadas cromáticas. O processo de extração de características das imagens digitais, cujos aspectos básicos podem ser encontrados em Parker (1996), foi resumido segundo o algoritmo seguinte:

- 1) Aquisição da imagem colorida;
- 2) Aplicação de filtro mediano, com vizinhança de três pixels;
- 3) Conversão da imagem colorida para tons de cinza;
- 4) Obtenção do *threshold* (limiar) de segmentação, por meio do algoritmo de Otsu (1979);
- 5) Binarização da imagem;
- 6) Complementação da imagem, para ficar com fundo negro e objeto branco;
- 7) Multiplicação da imagem colorida original pela imagem binária, a fim de se obter a imagem colorida segmentada;
- 8) Separação de cada canal RGB em três vetores, com o intuito de contar apenas os pixels do objeto;

9) Computação dos pixels médios, seguidos de normalização, conforme a expressão (18.2);

10) Obtenção das coordenadas cromáticas, de acordo com a expressão (18.1).

A rede PMC utilizada para esta fase possui as seguintes características topológicas:

- Camada de entrada → sete entradas, sendo seis delas de atributos de cor (três de pixels médios e três de coordenadas cromáticas) e uma entrada referente ao diâmetro equatorial dos tomates;
- Camada neural escondida → 15 neurônios;
- Camada neural de saída → 2 neurônios, cujas saídas combinadas serão utilizadas para representar a classificação dos tomates a ser efetuada pela rede, conforme as quatro classes (*A*, *B*, *C* ou *D*) definidas na Tabela 18.1.

Esta topologia com uma única camada escondida pode ser mais bem compreendida por meio da figura 18.4, em que os atributos de cor serão fornecidos pelos seis componentes calculados pelas expressões (18.1) e (18.2), isto é, três componentes referentes às coordenadas cromáticas e três componentes advindas do cálculo do pixel médio, respectivamente.

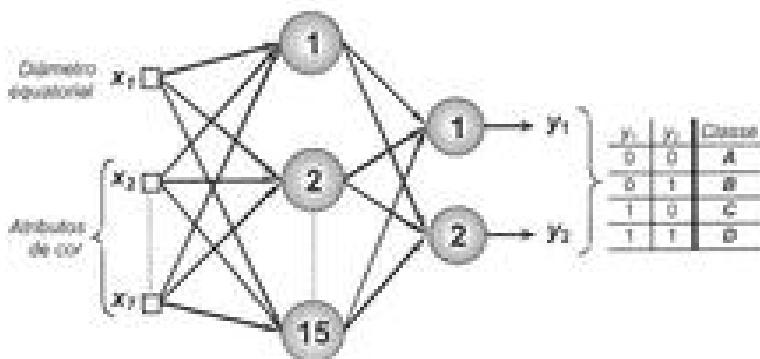


Figura 18.4 – Topologia da rede PMC para a classificação de tomates

Embora esta primeira configuração, apresentada na figura 18.4, possa ser utilizada para realizar o procedimento de classificação de tomates, optou-se também por testar uma segunda configuração, denominada “cascatada”, com o propósito de verificar se haveria melhoria substantiva na taxa de acertos.

Esta segunda configuração foi composta por duas redes PMC cascata-das, isto é, as respostas produzidas pela primeira rede PMC serão as entradas da segunda rede. A primeira tem a sua topologia representada como se segue:

- Camada de entrada → duas entradas, sendo as *Coordenadas cromáticas*  $\{r\}$  e  $\{g\}$ , pois a predominância de *vermelho* e *verde* nas classes *maduro* e *verde*, respectivamente, são os fatores de interesse;
- Camada neural escondida → 5 neurónios;
- Camada neural de saída → 1 neurónio, cuja saída produzirá o valor unitário  $\{1\}$  para identificar a classe *Verde*, ao passo que fornecerá o valor nulo  $\{0\}$  para indicar a classe *Maduro*.

Esta primeira rede PMC, que compõe a segunda configuração testada (cascatada), encontra-se representada na figura 18.5.

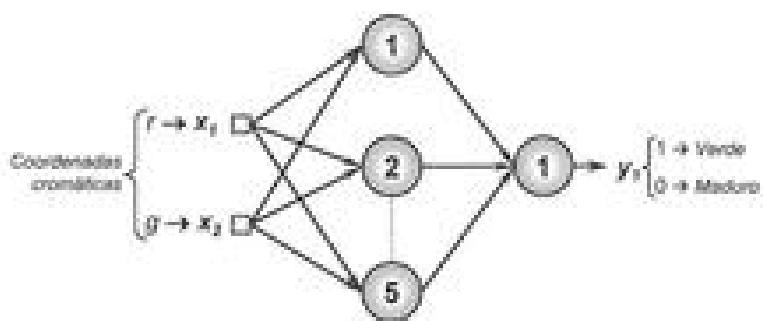


Figura 18.5 – Topologia da primeira rede PMC para classificar tomates entre verde e maduro (configuração cascata-das)

Uma vez que as amostras de tomates sejam classificadas em *verde* ou *maduro*, pode-se então levar em consideração a variável *diametro equatorial* a fim de realizar a classificação das 4 variedades de tomates descritas anteriormente. Esta segunda rede PMC que compõe a configuração cascata-das, conforme ilustração da figura 18.6, tem a seguinte topologia:

- Camada de entrada → duas entradas, sendo uma a resposta  $\{y_1\}$  produzida pela primeira rede PMC (figura 18.5). A outra entrada é a própria medida do *Diametro equatorial* das amostras de tomate;

- Camada neural escondida → 5 neurônios;
- Camada neural de saída → 4 neurônios, cujas saídas serão empregadas na classificação de tomates em uma das quatro classes definidas anteriormente.

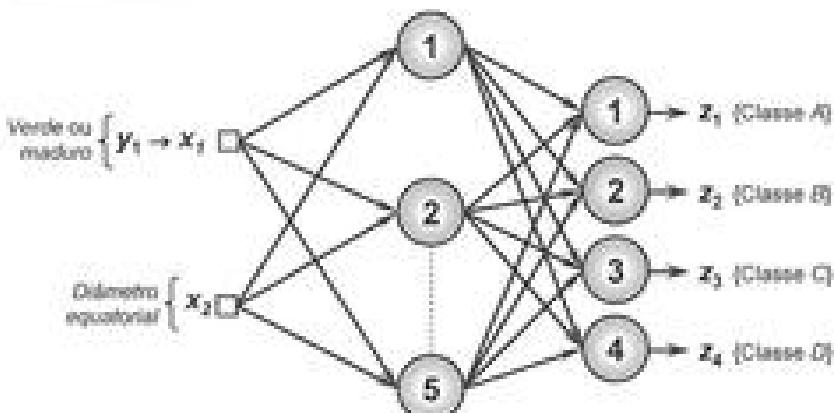


Figura 18.6 – Topologia da segunda rede PMC para classificação de tomates (configuração cascataada)

Em relação ao processo de aprendizagem, todas as redes neurais artificiais utilizaram 90% do total das amostras disponíveis para a fase de treinamento, enquanto que as amostras restantes (10%) foram usadas para a fase de teste.

A sistemática adotada para as respostas produzidas pelos neurônios de saída da segunda rede PMC (figura 18.6) foi a estratégia *one of c-classes*, conforme descrito na subseção 5.4.1.

### 18.3 – Resultados computacionais

A primeira configuração de rede PMC testada apresentou taxas de acerto de 80%. O número de neurônios foi também incrementado para se melhorar os resultados, chegando-se então a uma eficiência final de 90%. Entretanto, a partir deste ponto de desempenho, a sua estrutura topológica começou a se tornar complexa, sendo incapaz de incrementar ainda mais as taxas de acerto já alcançadas.

Já em relação à segunda configuração implementada (duas redes PMC cascataeadas), a classificação entre verde e maduro, realizada pela primeira rede PMC, apresentou taxas de acerto de 100% durante a fase de testes, sendo este

percentual também atingido pela segunda rede PMC, a qual foi responsável pela classificação das quatro variedades de tomates. Destarte, esta opção de se trabalhar com duas redes neurais artificiais em cascata foi a mais eficiente.

Em suma, uma das principais tarefas associadas à presente aplicação está justamente no processo de seleção das melhores variáveis a serem apresentadas às redes, a fim de que possam realizar a correta classificação dos tomates. A associação de duas redes em cascata, com o intuito de dividir as tarefas de classificação, mostrou-se a mais eficiente do ponto de vista computacional, obtendo-se aqui os melhores resultados quando comparado com a utilização de uma única rede neural mais complexa.

## Análise de desempenho de redes RBF e PMC em classificação de padrões

### 19.1 – Introdução

Tanto as redes RBF<sup>1</sup> como as redes PMC podem ser empregadas como classificadoras de padrões, frente às situações em que se tem espaços de grandes dimensões.

Análises sobre quais tipos de rede produziriam os melhores resultados, assim como orientações sobre a topologia mais apropriada para se chegar a tais desempenhos, são de grande valia para se escolher a rede mais indicada para cada tipo de problema.

Por outro lado, a capacidade de reconhecimento de padrões inerentes aos seres humanos é proveniente da correlação de diversas informações advindas dos órgãos sensitivos. Por certo, os seres humanos são exímios classificadores de padrões, realizando tais tarefas com um mínimo de esforço. De maneira semelhante, por meio do processo de aprendizado, as redes neurais artificiais também conseguem atribuir classes mediante o processamento de sinais de entrada.

Outrossim, quando uma nova amostra é apresentada, as redes extraem as informações necessárias para realizar a classificação, pois possuem certa habilidade de generalização, a qual foi adquirida do aprendizado por intermédio de exemplos.

As redes PMC, apresentadas em detalhes no capítulo 5, possuem uma ou mais camadas intermediárias, sendo que cada neurônio está sob a ação de uma função de ativação não-linear. O altíssimo grau de conectividade entre seus neurônios é também uma característica inerente a este tipo de arquitetura neural.

Já as redes RBF, conforme expostas minuciosamente no capítulo 6, possuem tipicamente apenas uma camada intermediária, sendo que as ativações utilizadas pelos neurônios são funções de base radial. A camada de saída, por sua vez, utiliza funções lineares para produzir a resposta da rede frente aos sinais de entrada.

Tipicamente, os neurônios das camadas ocultas de uma rede PMC obedecem a um mesmo modelo. Já os neurônios da camada oculta de uma RBF seguem modelos diferenciados, servindo também a propósitos diferentes.

## 19.2 – Características das redes RBF e PMC

Os sinais de entrada utilizados para realizar a comparação de desempenho entre as redes PMC e RBF são provenientes de duas bases de dados públicas e notórias, isto é, as bases *Wine* e *Breast Cancer Wisconsin*.

A base de dados *Wine* é identificada por 14 atributos numéricos, sendo 13 entradas e uma saída, e que representam três classes distintas de vinho. As frequências dessas classes são dadas na tabela 19.1.

Tabela 19.1 – Distribuição de amostras em classes (base de dados *Wine*)

Classes	Frequência %
1	33,14
2	39,88
3	26,27

A base de dados *Wisconsin* possui também atributos numéricos, sendo nove entradas e uma saída, provenientes de análises de biópsias realizadas em pacientes. As amostras são classificadas como de tipo maligno ou benigno, cujas frequências estão na tabela 19.2.

Tabela 19.2 – Distribuição de amostras em classes (base de dados *WDBC*)

Classes	Frequência %
Benigno	65,0
Maligno	35,0

É importante ressaltar que ambas as bases de dados têm sido muito utilizadas em investigações que envolvem análises comparativas entre métodos destinados à classificação de padrões.

### 19.3 – Resultados computacionais

A fim de comparar o desempenho de diversas topologias tanto de PMC como de RBF, implementou-se o mesmo número de neurônios em suas camadas intermediárias, sendo que para as redes PMC foram testadas topologias com apenas uma camada intermediária. Utilizou-se para a fase de treinamento um total de 90% das amostras disponíveis nas bases de dados, enquanto as 10% restantes foram empregadas na fase de teste.

A partir da análise da figura 19.1, torna-se então possível constatar que, considerando as amostras da base de dados *Wbc*, as redes PMC apresentaram os melhores resultados durante a fase de treinamento, em topologias com até dois neurônios na camada intermediária, pois os desvios no decorrer dos treinamentos foram menores para estas configurações.

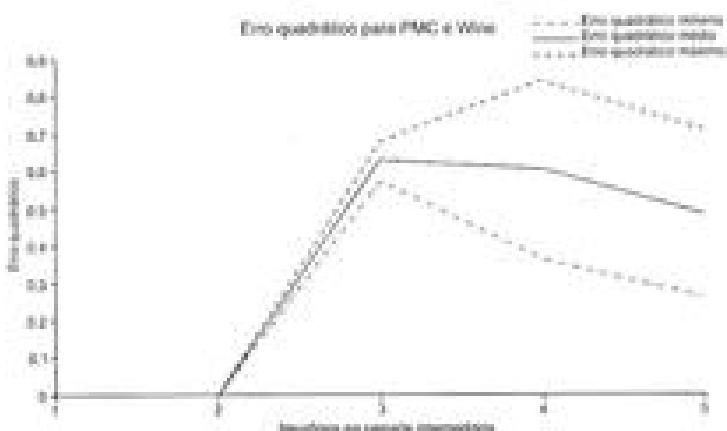


Figura 19.1 – Resultados de treinamento de redes PMC (base de dados *Wbc*)

Conforme ilustrado na figura 19.2, topologias de redes PMC, com a inclusão de termo de *momentum*, foram avaliadas para verificar o número de épocas necessárias durante a convergência.

Observa-se neste caso que os melhores resultados de treinamento, com valores de desvios reduzidos, foram observados para as topologias de PMC constituídas de 1, 2 e 4 neurônios em suas camadas intermediárias.

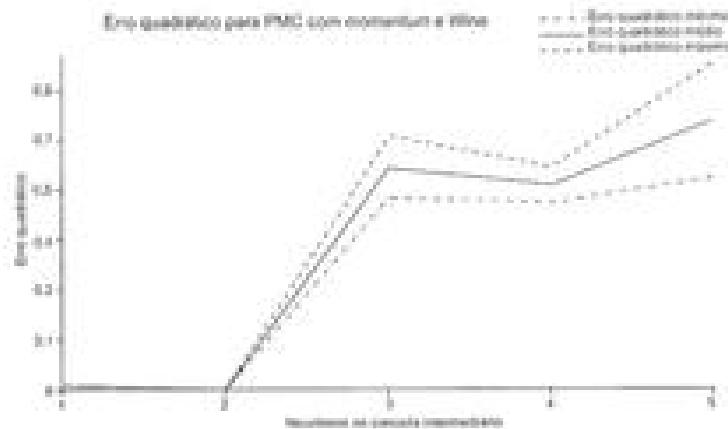


Figura 19.2 – Resultados de treinamento de redes PMC com termo de *momentum* (base de dados *W'air*)

Já na figura 19.3 são exibidos os resultados para o treinamento das redes PMC, assumindo-se agora a base de dados *W'rennings*.

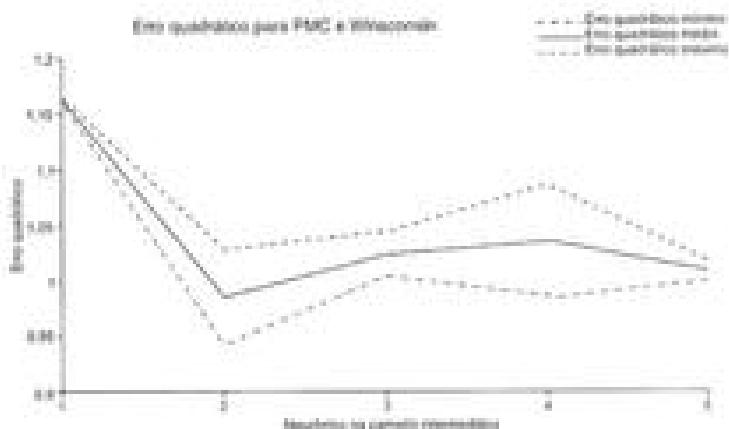


Figura 19.3 – Resultados de treinamento de redes PMC (base de dados *W'rennings*)

Para a base de dados *W'rennings*, implementou-se também redes PMC com termo de *momentum*, cujos resultados encontram-se na figura 19.4.

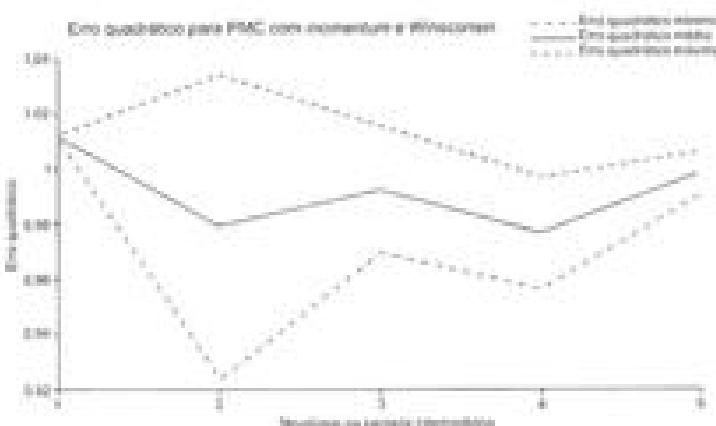


Figura 19.4 – Resultados de treinamento de redes PMC com termo de *momentum* (base de dados Wiscbrain)

Para as topologias apresentadas na figura 19.4, considerando ainda os valores dos desvios, observa-se que as configurações com 2 e 4 neurônios na camada intermediária forneceram os melhores resultados. Na figura 19.5 é possível verificar o número de épocas médio que foi necessário para realizar cada treinamento do PMC.



Figura 19.5 – Número de épocas necessárias para treinamento das diversas redes PMC

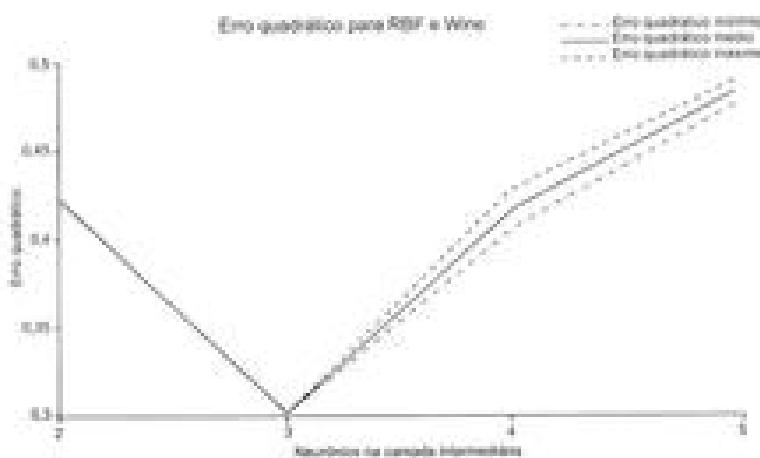
Para as duas bases de dados investigadas, percebe-se uma sensível redução no número de épocas de treinamento quando da inserção do termo de *momentum*. As taxas de acerto obtidas com os testes das redes PMC podem ser visualizadas na tabela 19.3.

Tabela 19.3 – Taxas de acertos obtidas nas fases de teste dos PMC

Neurônios da camada intermediária	Taxas de acertos (%) Wine		Taxas de acertos (%) Winsconsin	
	Sem Momentum	Com Momentum	Sem Momentum	Com Momentum
5	100,0	100,0	98,6	98,6
4	100,0	100,0	98,6	98,6
3	100,0	100,0	98,6	98,6
2	83,3	81,5	98,6	98,6
1	53,3	44,4	71,0	71,0

Com viés nos resultados apresentados, pode-se afirmar, considerando a base *Wine*, que a melhor topologia de PMC apresenta 3 neurônios na camada intermediária, fazendo-se uso de treinamento com termo de *momentum*. Já em relação à base de dados *Winsconsin*, também se deve empregar o termo de *momentum* para acelerar o processo de treinamento, obtendo-se os melhores resultados para uma topologia com 2 neurônios na camada intermediária. Essas topologias foram escolhidas levando-se em consideração a complexidade da rede e também a eficiência obtida nos testes realizados.

Nas figuras 19.6 a 19.8 são ilustrados os resultados obtidos pelas redes RBF, frente às mesmas bases *Wine* e *Winsconsin*.

Figura 19.6 – Resultados de treinamento de redes RBF (base de dados *Wine*)

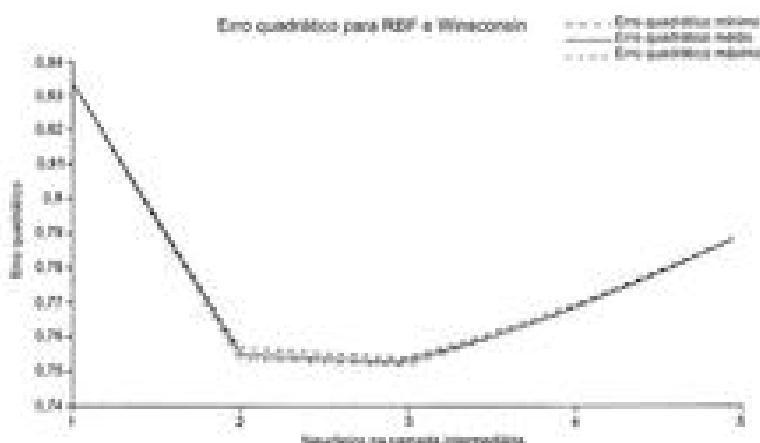


Figura 19.7 – Resultados de treinamento de redes RBF (base de dados Wineconin)

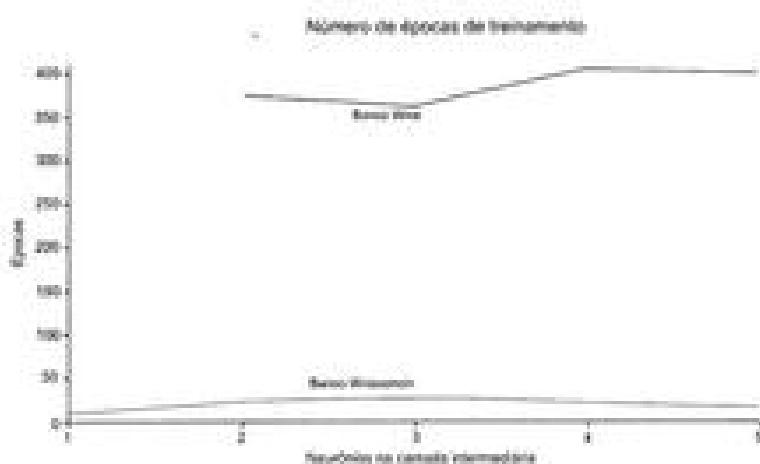


Figura 19.8 – Número de épocas necessárias para treinamento das diversas redes RBF

As taxas de acerto obtidas durante as fases de teste das redes RBF podem ser examinadas na tabela 19.4.

Tabela 19.4 – Taxas de acertos obtidas nas fases de teste das RBF

Neurônios da camada intermediária	Taxas de acertos (%) Wine	Taxas de acertos (%) Wisconsin
5	100,0	89,7
4	94,4	91,3
3	100,0	94,2
2	88,9	95,7
1		95,7

Analisando-se todos os resultados obtidos pelas redes RBF, é possível escolher a topologia, com 3 neurônios na camada intermediária, como a melhor classificadora para a base de dados *Wine*, ao passo que aquela constituída de apenas 1 neurônio seria a mais apropriada para a base *Wisconsin*.

Em suma, os resultados elencados, mediante esta análise comparativa resumida, mostraram-se muito similares entre ambas as redes testadas no que se refere à capacidade de acertos. Este fato já era esperado, pois tanto as redes PMC como as redes RBF são aproximadoras universais de funções, permitindo-se que sempre haja uma RBF capaz de operar de maneira semelhante a uma PMC específica.

No entanto, pôde-se observar um comportamento bem diferenciado entre as redes frente ao processo de treinamento, principalmente com a inserção do termo de *momentum* nas topologias PMC, implicando uma considerável aceleração de suas convergências.

Por outro lado, a rede RBF necessitou de menos neurônios que a rede PMC, quando aplicada na classificação da base de dados *Wisconsin*.

A análise comparativa permitiu ainda confirmar que, conforme investigado na subseção 5.5.3, nem sempre a topologia que apresenta melhor desempenho durante a fase de treinamento será aquela que produzirá também os melhores resultados frente à fase de teste. Tal ocorrência pode ser observada a partir da comparação entre os resultados fornecidos na Figura 19.1 em relação àqueles apresentados na Tabela 19.3.

# Resolução de problemas de otimização com restrições por redes de Hopfield

## 20.1 – Introdução

Problemas de otimização com restrições referem-se, geralmente, ao problema de minimizar ou maximizar uma função objetivo sujeita a um conjunto de restrições (lineares e/ou não-lineares) de igualdade e/ou desigualdade (Bazaraa & Shetty, 1979).

Embora existam na literatura diversos métodos de programação matemática que possam ser aplicados na resolução de problemas de otimização com restrições, há a necessidade crescente de investigar métodos alternativos que explorem arquiteturas de processamento inherentemente paralelas e adaptativas. Assim sendo, as redes neurais artificiais tornam-se uma abordagem promissora que pode ser aplicada eficientemente nestes tipos de problemas. Entre as principais vantagens de se utilizar a abordagem de redes neurais artificiais em otimização não-linear restrita, destacam-se: capacidade intrínseca de operação em paralelo; simplicidade de implementação em hardware; e alcance de altas taxas de computação por intermédio de elementos simples de processamento.

O problema a ser tratado neste capítulo consiste em aplicar uma rede de Hopfield a fim de minimizar uma função de energia  $E^*$ , a qual representa a função objetivo do problema, estando sujeita a diversas restrições de igualdade e/ou desigualdade. Essas restrições são todas agrupadas em um único

termo de energia, denominado  $E^{ext}$ , o qual possui a finalidade de confinar todas as restrições envolvidas com o processo de otimização. Assim sendo, os pontos de equilíbrio da rede de Hopfield estarão representando as possíveis soluções deste problema de otimização.

Conforme visto no capítulo 7, a expressão que rege o comportamento em tempo contínuo de cada neurônio da rede de Hopfield é dada por:

$$\dot{u}_j(t) = -\eta \cdot u_j(t) + \sum_{i=1}^n W_{ji} \cdot v_i(t) + i_j^b, \text{ onde } j = 1, \dots, n \quad (20.1)$$

$$v_j(t) = g(u_j(t)) \quad (20.2)$$

onde:

$u_j(t)$  é o estado interno do  $j$ -ésimo neurônio, sendo  $\dot{u}_j(t) = du/dt$ ;

$v_j(t)$  é a saída do  $j$ -ésimo neurônio;

$W_{ji}$  é o valor do peso sináptico conectando o  $j$ -ésimo neurônio ao  $i$ -ésimo neurônio;

$i_j^b$  é o limiar aplicado ao  $j$ -ésimo neurônio;

$g(\cdot)$  é uma função de ativação, monótona crescente, que limita a saída de cada neurônio em um intervalo predefinido;

$\eta \cdot u_j(t)$  é um termo de decaimento passivo.

Os pontos de equilíbrio da rede, conforme apresentado na seção 7.3, correspondem aos valores de  $v(t)$  que minimizam a seguinte função de energia associada à rede de Hopfield:

$$E(t) = -\frac{1}{2} v(t)^T \cdot W \cdot v(t) - v(t)^T \cdot i^b \quad (20.3)$$

Portanto, o mapeamento de problemas de otimização utilizando uma rede de Hopfield consiste em determinar, para cada tipo de problema, a matriz de pesos  $W$  e o vetor de limiares  $i^b$  associados à função de energia da rede (20.3).

Para o caso de problemas de otimização com restrições, utilizar-se-á aqui uma função de energia composta por dois termos, ou seja:

$$E(t) = E^{obj}(t) + E^{conf}(t) \quad (20.4)$$

Em que:

$$E^{obj}(t) = -\frac{1}{2} \mathbf{v}(t)^T \cdot \mathbf{W}^{obj} \cdot \mathbf{v}(t) - \mathbf{v}(t)^T \cdot \mathbf{f}^{obj} \quad (20.5)$$

$$E^{conf}(t) = -\frac{1}{2} \mathbf{v}(t)^T \cdot \mathbf{W}^{conf} \cdot \mathbf{v}(t) - \mathbf{v}(t)^T \cdot \mathbf{f}^{conf} \quad (20.6)$$

Desta forma, a rede de Hopfield atuará com o propósito de minimizar simultaneamente uma função de energia ( $E^o(t)$ ), correspondente à função objetivo, bem como terá o intuito de minimizar outra função de energia ( $E^{conf}(t)$ ) que estará mapeando todas as restrições estruturais envolvidas no problema.

## 20.2 – Características da rede de Hopfield

A topologia de rede de Hopfield, constituída de dois termos de energia, visando à resolução de problemas de otimização com restrições, é ilustrada na figura 20.1.

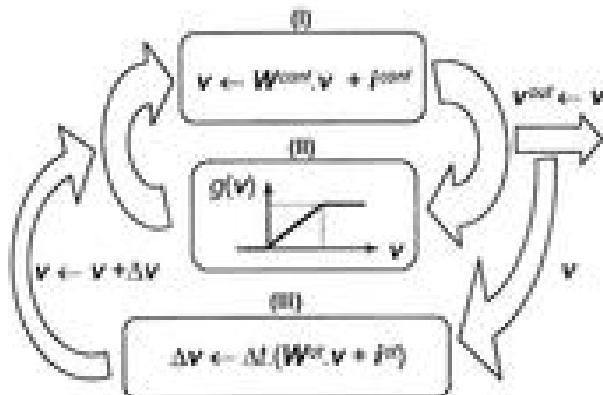


Figura 20.1 – Rede de Hopfield para resolução de problemas de otimização com restrições

Conforme se pode averiguar nesta figura, a dinâmica de operação da rede pode ser explicitada por intermédio de três etapas principais, isto é:

**I – Minimização de  $E^{\text{ext}}$**  → corresponde à projeção de  $\mathbf{v}$  sobre um subespaço-válido que confina todas as restrições impostas pelo problema:

$$\mathbf{v}(t+1) = \mathbf{W}^{\text{cont}} \cdot \mathbf{v}(t) + \mathbf{f}^{\text{cont}} \Rightarrow \mathbf{v} \leftarrow \mathbf{W}^{\text{cont}} \cdot \mathbf{v} + \mathbf{f}^{\text{cont}} \quad (20.7)$$

onde  $\mathbf{W}^{\text{ext}}$  é uma matriz projeção ( $\mathbf{W}^{\text{ext}} \cdot \mathbf{W}^{\text{ext}} = \mathbf{W}^{\text{ext}}$ ) e  $\mathbf{f}^{\text{ext}}$  é ortogonal ao subespaço-válido ( $\mathbf{W}^{\text{ext}} \cdot \mathbf{f}^{\text{ext}} = 0$ ). Uma análise detalhada desta técnica de subespaço-válido é apresentada em Silva (1997) e Aiyer *et al.* (1990).

**II – Aplicação de função de ativação** → emprego da função de ativação do tipo rampa simétrica, restringindo  $\mathbf{v}$  dentro de um hipercubo pré-definido:

$$g_i(v_i) = \begin{cases} \lim_i^{\text{inf}}, & \text{se } v_i < \lim_i^{\text{inf}} \\ v_i, & \text{se } \lim_i^{\text{inf}} \leq v_i \leq \lim_i^{\text{sup}} \\ \lim_i^{\text{sup}}, & \text{se } v_i > \lim_i^{\text{sup}} \end{cases} \quad (20.8)$$

onde  $v_i(t) \in [\lim_i^{\text{inf}}, \lim_i^{\text{sup}}]$ . Nesta circunstância, embora  $\mathbf{v}$  esteja dentro de um conjunto particularizado, a rede de Hopfield pode ser também estruturada para representar qualquer problema de otimização com restrições. Em tal situação, tem-se que  $\lim_i^{\text{inf}} = -\infty$  e  $\lim_i^{\text{sup}} = \infty$ .

**III – Minimização de  $E^{\text{c}}$**  → alteração de  $\mathbf{v}$  em direção a uma solução de custo ótimo (definida por  $\mathbf{W}^{\text{c}}$  e  $\mathbf{f}^{\text{c}}$ ), que é correspondente aos pontos de equilíbrio da rede, os quais são as soluções para o problema de otimização considerado. Utilizando a função de ativação rampa-simétrica, e dado que  $\eta = 0$ , pois as alterações nos neurônios são realizadas de forma síncrona, ou seja, todos os neurônios são alterados simultaneamente, então a equação (20.1) torna-se:

$$\frac{d\mathbf{v}(t)}{dt} = \dot{\mathbf{v}} = -\frac{\partial E^{\text{c}}(t)}{\partial \mathbf{v}} \quad (20.9)$$

$$\Delta \mathbf{v} = -\Delta t \cdot \nabla E^{\text{c}}(\mathbf{v}) = \Delta t \cdot (\mathbf{W}^{\text{c}} \cdot \mathbf{v} + \mathbf{f}^{\text{c}}) \quad (20.10)$$

Assim, a minimização de  $E^v$  consiste na alteração de  $v$  na direção oposta ao gradiente da função  $E^v$  em relação à  $v$ . Estes resultados são também válidos quando uma função de ativação do tipo tangente hiperbólica é utilizada.

Deve-se ressaltar que a aplicação das Etapas (I) a (III) é realizada de maneira discreta. Neste caso, após as considerações realizadas para a Etapa (III), a resolução da equação diferencial, representada em (20.9), será similar ao método de Euler [Franco, 2006] e, em termos de otimização, isto representa um processo de descida gradiente com passo fixo igual a  $\Delta t$ .

Em resumo, após cada etapa de otimização em (III), torna-se então necessário executar, iterativamente, as duas etapas subsequentes (I) e (II) até que haja a convergência do vetor de saída  $v$ , assegurando-se, por conseguinte, que todas as restrições envolvidas com o problema sejam satisfeitas.

Portanto, de acordo com a Figura 20.1, cada iteração tem dois estágios bem distintos. No primeiro estágio, conforme descrito pela Etapa (III), o vetor  $v$  é alterado usando apenas o gradiente do termo  $E^v$ . Já no segundo, após cada etapa proporcionada pela Etapa (III), o vetor  $v$  é então projetado ao subespaço-válido {Etapa (I)}, que está confinando todas aquelas restrições para o problema, sendo que, em seguida, é limitado pela função de ativação {Etapa (II)} a fim de que seus elementos estejam no domínio  $[l_{m,i}^{min}, l_{m,i}^{max}]$ .

Finalmente, o processo de convergência é concluído quando os valores de  $v^{mt}$ , durante duas iterações sucessivas, permanecem praticamente constantes, onde o valor de  $v^{mt}$  nesta condição será igual ao próprio valor de  $v$ .

### 20.3 – Mapeamento de problemas de otimização pela rede de Hopfield

Considera-se aqui o seguinte problema de otimização, composto de  $p$  restrições de desigualdade e  $N$  variáveis, ou seja:

$$\text{Minimizar } E^{\text{obj}}(v) = f(v) \quad (20.11)$$

$$\text{Sujeito a } E^{\text{conf}}(v) : h_i(v) \leq 0, \text{ onde } i \in \{1, \dots, p\} \quad (20.12)$$

$$x^{\min} \leq v \leq x^{\max} \quad (20.13)$$

onde  $\mathbf{v}$ ,  $\mathbf{x}^{\min}$ ,  $\mathbf{x}^{\max} \in \mathbb{R}^N$ ;  $f(\mathbf{v})$  e  $h(\mathbf{v})$  são funções contínuas e diferenciáveis. Os vetores  $\mathbf{x}^{\min}$  e  $\mathbf{x}^{\max}$  definem, respectivamente, os limites inferiores e superiores para cada uma das componentes do vetor  $\mathbf{v}$ . As condições (20.12) e (20.13) definem um conjunto fechado convexo em  $\mathbb{R}^N$ , no qual o vetor  $\mathbf{v}$  deve permanecer a fim de representar uma solução válida para o problema de otimização regido por (20.11).

Uma resposta para o problema pode ser obtida por uma rede de Hopfield, cuja implementação do subespaço-válido garante a satisfação das condições definidas em (20.12). Além disso, o hipercubo inicial representado pelas restrições de desigualdade (variáveis canalizadas), em (20.13), é diretamente mapeado pela função rampa-simétrica dada por (20.8), a qual é usada como função de ativação neural, isto é,

$$\mathbf{v} \in [\mathbf{x}^{\min}, \mathbf{x}^{\max}]$$

Os parâmetros  $\mathbf{W}^{\text{ext}}$  e  $\mathbf{P}^{\text{ext}}$ , pertencentes ao subespaço-válido, são calculados transformando-se as restrições de desigualdade em (20.12) num conjunto de restrições de igualdade, mediante a utilização de uma variável auxiliar para cada restrição de desigualdade:

$$h_i(\mathbf{v}) + \sum_{j=1}^p \delta_{ij} \cdot q_j = 0 \quad (20.14)$$

onde  $q_j \geq 0$  são as variáveis auxiliares, que podem ser tratadas como variáveis  $v_j$ , e  $\delta_{ij}$  é definida pela função impulso de Kronecker [Graham, 1986], sendo dada por:

$$\delta_{ij} = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{se } i \neq j \end{cases} \quad (20.15)$$

Após esta transformação, as equações (20.11), (20.12) e (20.13), as quais definem o problema de otimização, podem ser rescritas como:

$$\text{Minimizar } E^{\text{OL}}(\mathbf{v}^+) - f(\mathbf{v}^+) \quad (20.16)$$

$$\text{Sujeito a } E^{\text{conf}}(\mathbf{v}^*) : h_i(\mathbf{v}^*) \leq 0, \text{ onde } i \in \{1, \dots, p\} \quad (20.17)$$

$$z^{\min} \leq v^* \leq z^{\max}, \text{ onde } i \in \{1, \dots, N\} \quad (20.18)$$

$$0 \leq v^* \leq z^{\max}, \text{ onde } i \in \{N+1, \dots, N'\} \quad (20.19)$$

onde  $N' = N + p$ , e  $\mathbf{v}'^T = [\mathbf{v}^T \ \mathbf{q}^T] \in \mathbb{R}^{N'}$  é um vetor de variáveis estendidas.

Deve-se notar que  $E^*$  não dependerá das variáveis auxiliares  $\mathbf{q}$ .

A matriz projeção  $W^{\text{ext}}$ , referente à equação do subespaço-válido, é calculada por meio da projeção de  $\mathbf{v}^*$ , obtida a partir da minimização de  $E^*(\mathbf{v}') = f(\mathbf{v}')$ , para o subespaço tangente à superfície delimitada pelas restrições dadas por (20.17). Desse modo, uma equação para  $W^{\text{ext}}$  pode ser definida por [Luenberger, 2003]:

$$W^{\text{ext}} = I - \nabla h(\mathbf{v}^*)^T \cdot (\nabla h(\mathbf{v}^*) \cdot \nabla h(\mathbf{v}^*)^T)^{-1} \cdot \nabla h(\mathbf{v}^*) \quad (20.20)$$

onde:

$$\nabla h(\mathbf{v}^*) = \begin{bmatrix} \frac{\partial h_1(\mathbf{v}^*)}{\partial v_1^*} & \frac{\partial h_1(\mathbf{v}^*)}{\partial v_2^*} & \cdots & \frac{\partial h_1(\mathbf{v}^*)}{\partial v_{N'}^*} \\ \frac{\partial h_2(\mathbf{v}^*)}{\partial v_1^*} & \frac{\partial h_2(\mathbf{v}^*)}{\partial v_2^*} & \cdots & \frac{\partial h_2(\mathbf{v}^*)}{\partial v_{N'}^*} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_p(\mathbf{v}^*)}{\partial v_1^*} & \frac{\partial h_p(\mathbf{v}^*)}{\partial v_2^*} & \cdots & \frac{\partial h_p(\mathbf{v}^*)}{\partial v_{N'}^*} \end{bmatrix} = \begin{bmatrix} \nabla h_1(\mathbf{v}^*)^T \\ \nabla h_2(\mathbf{v}^*)^T \\ \vdots \\ \nabla h_p(\mathbf{v}^*)^T \end{bmatrix} \quad (20.21)$$

Substituindo-se a matriz  $W^{\text{ext}}$ , dada por (20.20), na equação do subespaço-válido em (20.7), obtém-se:

$$\mathbf{v}'^* \leftarrow (I - \nabla h(\mathbf{v}^*)^T \cdot (\nabla h(\mathbf{v}^*) \cdot \nabla h(\mathbf{v}^*)^T)^{-1} \cdot \nabla h(\mathbf{v}^*)) \cdot \mathbf{v}^* + l^{\text{conf}} \quad (20.22)$$

Por fim, resultados da teoria de estabilidade de Lyapunov [Vidyasagar, 2002] devem ser introduzidas na equação (20.22), a fim de se garantir a es-

tabilidade do sistema não-linear e, consequentemente, forçar a convergência da rede para os pontos de equilíbrio que representam uma solução para o problema. Pela definição de Jacobiano, quando  $v$  tende ao ponto de equilíbrio implica-se que  $v^* = 0$ . Neste caso, o valor de  $J^{ext}$  deve ser também nulo para satisfazer a condição de equilíbrio, isto é:

$$v^* = v(t) = v(t + \Delta t) = 0 \quad (20.23)$$

Assim, a equação (20.17) pode ser aproximada por:

$$h(v^*) \approx h(v^*) + J \cdot (v^* - v^*) \quad (20.24)$$

onde  $J = \nabla h(v^*)$  e  $h(v^*) = [h_1(v^*) \ h_2(v^*) \ \dots \ h_p(v^*)]^T$ .

Na vizinhança do ponto de equilíbrio, onde  $v^*$  pode ser considerado nulo ( $v^* = 0$ ), há a seguinte relação:

$$\lim_{v^* \rightarrow v^*} \frac{\|h(v^*)\|}{\|v^*\|} = 0 \quad (20.25)$$

Introduzindo-se os resultados de (20.24) e (20.25) em (20.22), obtém-se:

$$v^* \leftarrow v^* - \nabla h(v^*)^T \cdot (\nabla h(v^*) \cdot \nabla h(v^*)^T)^{-1} \cdot h(v^*) \quad (20.26)$$

Portanto, em (20.26), há a sintetização da expressão do subespaço-válido para tratamento de sistemas de equações não-lineares. Diante do exposto, para problemas de otimização com restrições, a expressão original do subespaço-válido em (20.7), o qual está representado pela etapa (I) na figura 20.1, deve ser substituída por (20.26). Assim, de acordo ainda com a figura 20.1, a aplicação sucessiva da Etapa (I), seguida da etapa (II), faz com que  $v$  converja para um ponto de equilíbrio que satisfaça todas as restrições impostas pelo problema de otimização não-linear.

Já os parâmetros  $W^*$  e  $P^*$ , associados ao termo de energia  $E^*$ , dado por (20.5) e representado em (20.16), são definidos com o objetivo de que a solução ótima corresponda à minimização de  $E^*$ . Este procedimento pode ser implementado alterando-se o vetor  $v^*$  na direção oposta ao gradiente da função de energia  $E^*$ . Desde que (20.17) e (20.18) definem um poliedro limitado, a função objetivo (20.16) tem sempre um ponto de mínimo. Logo, os pontos de equilíbrio da rede podem ser calculados assumindo-se os seguintes valores para  $T^*$  e  $P^*$ :

$$\rho^* = - \left[ \frac{\partial f(v^*)}{\partial v_1} \quad \frac{\partial f(v^*)}{\partial v_2} \quad \dots \quad \frac{\partial f(v^*)}{\partial v_N} \right]^T \quad (20.27)$$

$$T^* = 0 \quad (20.28)$$

Conforme mencionado anteriormente, visto que  $v^* = [v^* \; w^*]$ , o vetor  $P^*$  dado em (20.27) seria então representado por:

$$\rho^* = - \left[ \frac{\partial f(v^*)}{\partial v_1} \quad \frac{\partial f(v^*)}{\partial v_2} \quad \dots \quad \frac{\partial f(v^*)}{\partial v_N} \quad \frac{\partial f(v^*)}{\partial q_1} \quad \frac{\partial f(v^*)}{\partial q_2} \quad \dots \quad \frac{\partial f(v^*)}{\partial q_p} \right]^T \quad (20.29)$$

Como o processo de otimização de  $E^*$  não depende dos valores das variáveis auxiliares  $q$ , a expressão dada em (20.29) pode ser então substituída por:

$$\rho^* = - \underbrace{\left[ \frac{\partial f(v^*)}{\partial v_1} \quad \frac{\partial f(v^*)}{\partial v_2} \quad \dots \quad \frac{\partial f(v^*)}{\partial v_N} \right]}_{N \text{ componentes}} \quad \underbrace{\left[ 0 \quad 0 \quad 0 \quad \dots \quad 0 \right]^T}_{p \text{ componentes}} \quad (20.30)$$

Para ilustrar o desempenho da rede neural proposta, alguns resultados de simulação são apresentados na próxima seção.

## 20.4 – Resultados computacionais

No primeiro problema, conforme proposto em Bazaraa & Shetty (1979), a função objetivo está sujeita somente às restrições de desigualdade, ou seja:

$$\text{Minimizar } f(v) = v^2 + v_1^2 + 4v_1 + 2v_2^2 - 6v_2 + 2v_3$$

$$\begin{array}{ll} \text{Sujeito a:} & \left. \begin{array}{l} v_1^2 + v_2^2 + 6v_3 \leq 15 \quad (\text{R1}) \\ v_1^2 - v_2 + 5v_3 \leq 25 \quad (\text{R2}) \\ v_1^2 + v_2^2 - v_3 \leq 10 \quad (\text{R3}) \\ 0 \leq v_1 \leq 4 \quad (\text{R4}) \\ 0 \leq v_2 \leq 2 \quad (\text{R5}) \\ v_3 \geq 0 \quad (\text{R6}) \end{array} \right. \end{array}$$

Para a otimização, com três restrições de desigualdade e com variáveis limitadas, o vetor solução ( ponto de equilíbrio) obtido, após a convergência da rede de Hopfield, é dado por  $v = [0,00001 \ 1,50002 \ 0,00000]^T$ , com função objetivo  $f(v) = -3,49999$ . Este resultado é bem próximos ao valor da solução ótima, fornecida por  $v^* = [0,0 \ 1,5 \ 0,0]^T$  e  $f(v^*) = -3,5$ . A figura 20.2 ilustra a evolução dos valores de  $v_1$ ,  $v_2$  e  $v_3$  obtidos pela rede de Hopfield em relação ao número de iterações.

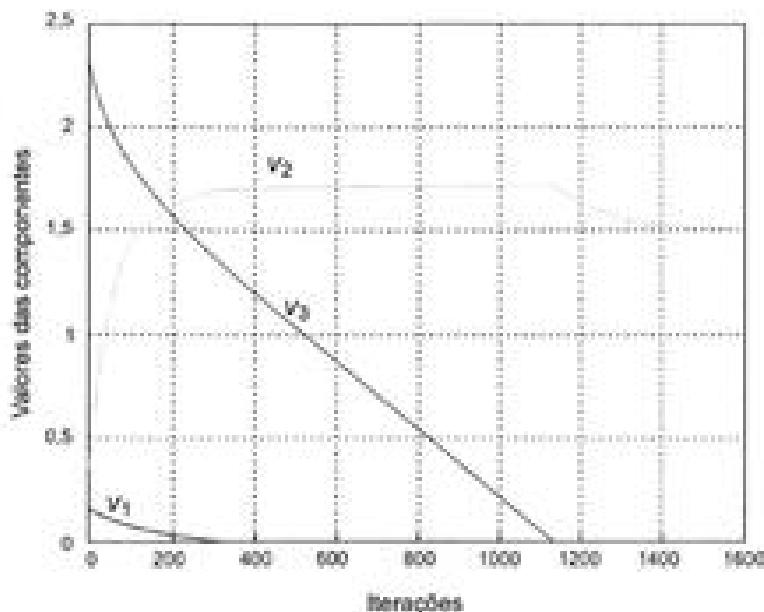


Figura 20.2 – Evolução do vetor de saída da rede (1º problema)

O esboço do comportamento da sua função objetivo é apresentado na figura 20.3. Cabe aqui também ressaltar que as restrições dadas em (R4), (R5) e (R6) são diretamente tratadas pela função de ativação da rede definida em (20.8).

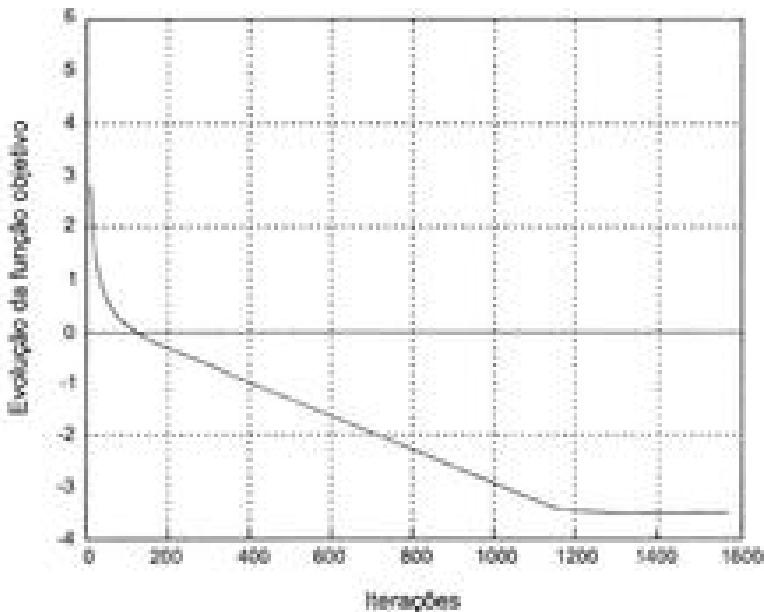


Figura 20.3 – Evolução da função objetivo (1º problema)

O segundo problema tratado pela rede de Hopfield é constituído por restrições de igualdade e de desigualdade, sendo o mesmo definido pelas seguintes expressões:

$$\text{Minimizar } f(\mathbf{v}) = v_1^2 + 2v_2^2 + v_3 + 2v_3$$

$$\begin{array}{l} \text{Sujeito a:} \\ \left\{ \begin{array}{l} v_1^2 + v_2^2 + v_3^2 = 4 \\ v_1^2 - v_2 + 2v_3 \leq 2 \\ v_1, v_2, v_3 \geq 0 \end{array} \right. \end{array}$$

Em relação a este problema, a figura 20.4 mostra a evolução dos valores de  $v_1$ ,  $v_2$  e  $v_3$  em função do número de iterações. O vetor solução obtido, após a convergência da rede, é dado por  $\mathbf{v} = [0,00000 \ 4,00000 \ 0,00001]^T$ , sendo o valor final da função objetivo  $f(\mathbf{v}) = E^0(\mathbf{v}) = 0,00034$ . A solução ótima para o problema é fornecida por  $\mathbf{v}^* = [0,0 \ 4,0 \ 0,0]^T$ , com  $E^0(\mathbf{v}^*) = 0,0$ .

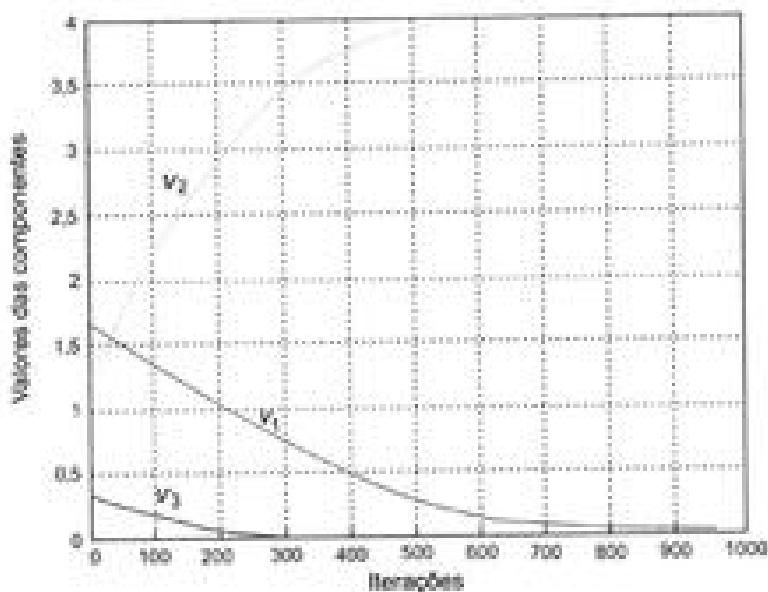


Figura 20.4 – Evolução do vetor de saída da rede (2º problema)

A figura 20.5, por sua vez, ilustra o comportamento da função objetivo do problema em relação ao número de iterações. Os valores iniciais atribuídos ao vetor  $v$  foram gerados aleatoriamente entre zero e um.

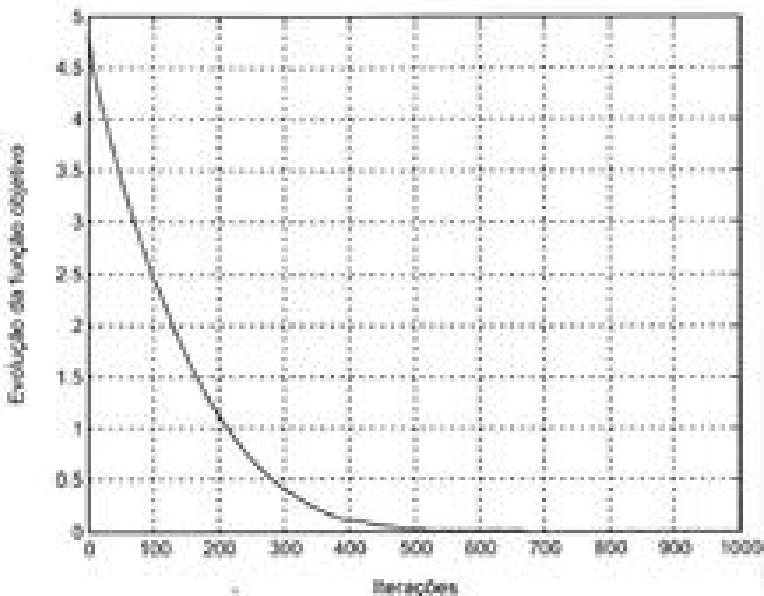


Figura 20.5 – Evolução da função objetivo (2º problema)

O terceiro e último problema visa confirmar a aplicabilidade da rede proposta quando todas as restrições associadas ao mesmo são do tipo linear, ou seja:

$$\text{Minimizar } f(v) = 0.4v_2 + v_1^2 + v_2^2 - v_1 \cdot v_2 + \frac{1}{30}v_1^3$$

$$\text{Sujeito a: } \begin{cases} v_1 + 0.5v_2 \geq 0.4 \\ 0.5v_1 + v_2 \geq 0.5 \\ v_1, v_2 \geq 0 \end{cases}$$

A figura 20.6 apresenta a evolução dos valores dos elementos do vetor de saída da rede em relação ao número de iterações. Após a convergência da rede, o vetor solução obtido é dado por  $v = [0.3398 \ 0.3301]^T$ , com  $E^*(v) = 0.2456$ . Estes resultados são bem próximos aos valores da solução exata, fornecida por  $v^* = [0.3395 \ 0.3302]^T$ , com função objetivo  $E^*(v^*) = 0.2455$ .

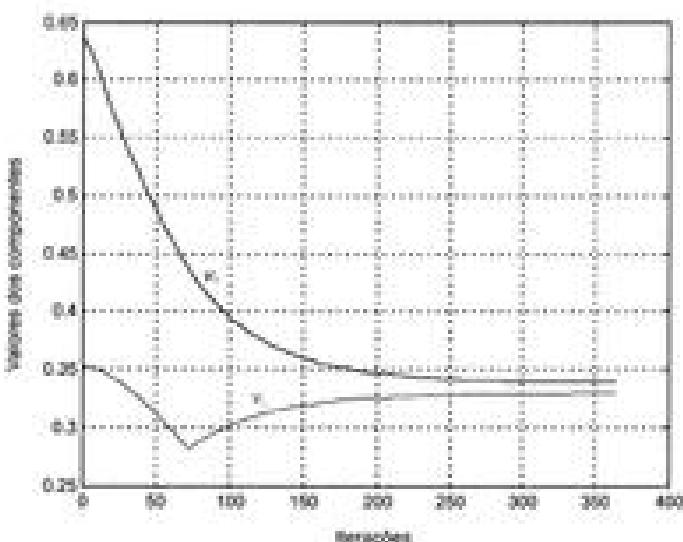


Figura 20.6 – Evolução do vetor de saída da rede (3º problema)

A figura 20.7 apresenta o comportamento da função objetivo do problema frente ao número de iterações. Neste caso, a rede foi iniciada com valores aleatórios, uniformemente distribuídos, entre zero e um.

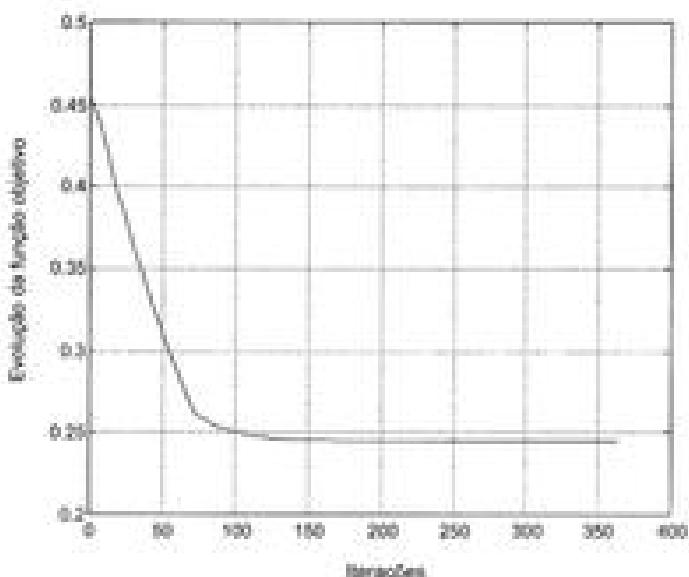


Figura 20.7 – Evolução da função objetivo (3º problema)

Estes resultados mostram a aplicação efetiva da rede de Hopfield para resolver problemas de otimização não-linear com restrições. A rede foi também

simulada considerando diversos valores iniciais atribuídos ao vetor de saída  $v$ . Para todas as simulações, a rede sempre convergiu para as mesmas soluções.

Embora o número de iterações seja relativamente elevado, o tempo gasto para a obtenção das soluções tem sido bastante curto. Em relação a este aspecto, vale ressaltar que, embora se tenha utilizado computadores sequenciais nessas simulações, o desempenho da rede alcançaria eficiência máxima quando for executada em computadores com processadores operando em paralelo, pois, nesta circunstância, cada unidade de processamento corresponderia a uma unidade neural.

Outro aspecto relevante diz respeito à qualidade das soluções obtidas pela rede. Apesar de os valores das soluções apresentarem alguns desvios em relação aos valores ótimos, observa-se que estes são menores quando comparados com outras abordagens neurais utilizadas nesses tipos de problemas.

É também motivo de registro que alguns dos testes realizados com problemas de dimensões maiores, tendo cerca de 800 restrições de desigualdade, mostraram que a rede pode tornar-se lenta em sua procura pelos pontos de equilíbrio que representam a solução do sistema. Este comportamento deve-se ao fato de se estar realizando toda a simulação em computadores sequenciais. A utilização de máquinas com processadores em paralelo, conforme mencionado anteriormente, pode potencialmente contornar esta situação.

Finalmente, o tratamento de problemas de otimização não-linear com restrições, por meio da rede de Hopfield, apresenta algumas particularidades em relação aos métodos primais [Bazaraa *et al.*, 2006] que são também utilizados na solução desses problemas, das quais se destacam: não há necessidade do cálculo do conjunto ativo de restrições em cada iteração; o vetor de inicialização da rede (solução inicial) não precisa pertencer ao conjunto factível definido pelas restrições; não há necessidade de obtenção, em cada iteração, de uma direção admissível de busca; e o mecanismo de busca independe da utilização de multiplicadores de Lagrange.



Leopold Kronecker



## Bibliografia

- Aguirre, L. A. (2000); *Introdução à identificação de sistemas – técnicas descent e não-descent aplicadas a sistemas reais*, Editora UFMG, Belo Horizonte.
- Aihara, K., Takabe, T., Toyoda, M. (1990); *Chaotic neural networks*. Physics Letters A, vol. 144, pp. 333-340.
- Aizer, S. V. B., Nitinjan, M., Faloutsos, F. (1990); *A theoretical investigation into the performance of the Hopfield model*. IEEE Transactions on Neural Networks, vol. 1, n°. 2, pp. 204-215.
- Alligood, K. T., Sauer, T. D., Yorke, J. A. (2000); *Chaos - an introduction to dynamical systems*. Springer-Verlag, New York, USA.
- Amit, D. J. (1992); *Modelling brain function - the world of attractor neural networks*. Cambridge University Press, Cambridge, UK.
- Amit, D. J., Gutfreund, H., Sompolinsky, H. (1985); *Spin-glass models of neural networks*. Physical Review A, vol. 32, pp. 1007-1018.
- Atencia, M., Joya, G., Sandoval, F. (2005); *Dynamical analysis of nonlinear higher-order Hopfield networks for combinatorial optimization*. Neural Computation, vol. 17, n°. 1802-1819.
- Aurenhammer, F. (1991); *Voronoi diagrams - a survey of a fundamental geometric data structure*. ACM Computing Surveys, vol. 23, n°. 3, pp. 345-405.
- Boraldi, A., Alpaydin, E. (2002); *Contractive feedforward ART clustering networks*. IEEE Transactions on Neural Networks, vol. 13, n°. 3, pp. 645-661.
- Bottini, R. (1992); *First and second order methods for learning between steepest descent and Newton's method*. Neural Computation, vol. 4, n°. 2, pp. 141-166.
- Bazaraa, M. S., Sherali, H. D., Shetty, C. M. (2006); *Nonlinear programming - theory and algorithms*. John Wiley & Sons, 3<sup>rd</sup> edition, Hoboken, New Jersey, USA.
- Bazaraa, M.S., Shetty, C.M. (1979); *Nonlinear programming*. John Wiley & Sons, New York, USA.

- Belu, V., Quintana, J. M., Avedillo, M. J. (2005); VTSL implementations of threshold logic – a non-invasive way. *IEEE Transactions on Neural Networks*, vol. 14, n°. 5, pp. 1217-1243.
- Bertsekas, D. P., Tsitsiklis, J. N. (1996). *Near-dynamic programming*. Athena Scientific, Belmont, Massachusetts, USA.
- Bouyer, A. (1981); Computing discrete solutions. *The Computer Journal*, vol. 24, n°. 2, pp. 162-166.
- Boydstad, R., Nishchany, L. (2004); *Dispositivos eletrônicos e teoria de circuitos*. Prentice-Hall, 8<sup>a</sup> edição, São Paulo.
- Buckley, J. J., Siler, W. (2004); *Fuzzy expert systems and fuzzy reasoning*. John Wiley, New York, USA.
- Carpenter, G. A., Grossberg, S. (1987a); A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, vol. 37, n°. 1, pp. 54-115.
- Carpenter, G. A., Grossberg, S. (1987b); ART2: self-organization of stable category recognition rules for analog input patterns. *Applied Optics*, vol. 26, n°. 23, pp. 4919-4930.
- Carpenter, G. A., Grossberg, S. (1988); The ART of adaptive pattern recognition by a self-organizing neural network. *Computer IEEE*, vol. 21, n°. 3, pp. 77-88.
- Carpenter, G. A., Grossberg, S. (1990); ART3: hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, vol. 3, n°. 2, pp. 129-152.
- Carpenter, G. A., Grossberg, S., Reynolds, J. H. (1991a); Adaptive real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, vol. 4, n°. 5, pp. 565-588.
- Carpenter, G. A., Grossberg, S., Rosen, D. B. (1991b); Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, vol. 4, n°. 6, pp. 759-771.
- Castro, I. N., Timmis, J. (2002); *Artificial immune systems – a new computational intelligent approach*. Springer-Verlag, London, UK.
- Chang, C.-Y., Su, S.-J. (2005); The application of a full counter-propagation neural network to image watermarking. *Proceedings of the IEEE International Conference on Networking, Sensing and Control*, Tucson, Arizona, pp. 993-998.
- Chen, S. (1995); Nonlinear time series modeling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning. *Electronics Letters*, vol. 31, n°. 2, pp. 117-118.
- Chen, S., Cowan, C. F. N., Grant, P. M. (1991); Orthogonal least square learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, vol. 2, n°. 2, pp. 302-309.
- Cho, J., Principe, J. C., Erdogmus, D., Mitter, M. A. (2006); Modeling and inverse controller design for an automated aerial vehicle based on the self-organizing map. *IEEE Transactions on Neural Networks*, vol. 17, n°. 2, pp. 445-460.
- Cho, K.-C., Ma, Y.-B., Lee, J.S. (2006); Design of computational grid-based intelligent ART-1 classification system for misinformative applications. *Proceedings of the 5<sup>a</sup> International Conference on Grid and Cooperative Computing Workshops*, Changsha, China, pp. 367-370.
- Coakley, J. R., Brown, C. E. (2000); *Artificial neural networks in accounting and finance: modeling issues*. *International Journal of Intelligent Systems in Accounting, Finance & Management*, vol. 9, n°. 2, pp. 119-144.
- Curry, B., Morgan, P. H. (2006); Model selection in neural networks: some difficulties. *European Journal of Operational Research*, vol. 170, n°. 2, pp. 567-577.
- Cybenko, G. V. (1989); Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, vol. 2, n°. 4, pp. 303-314.

- D'Inverno, M., Luck, M. (2004); *Understanding agent systems*. Springer-Verlag, Berlin, Germany.
- Daley, M. N., Cornell, G. W. (2002); *Empathic a neural network that categorizes facial expressions*. Journal of Cognitive Neuroscience, vol. 14, pp. 1158-1173.
- Dorigo, D. (2006); *Advances in artificial immune systems*. IEEE Computational Intelligence Magazine, vol. 1, n°. 4, pp. 40-49.
- Dorigo, D., Michalewicz, Z. (1997); *Evolutionary algorithms in engineering applications*. Springer-Verlag, Berlin, Germany.
- Dong, Y., Shao, M., Tai, X. (2008); *An adaptive counter propagation network based on soft competition*. Pattern Recognition Letters, vol. 29, n°. 7, pp. 938-949.
- Duda, R. O., Hart, P. E., Stork, D. G. (2001); *Pattern classification*. John Wiley, New York, USA.
- Eliasm, J. L. (1990); *Finding structure in time*. Cognitive Science, vol. 14, pp. 179-211.
- Faggin, F. (1991); *Tutorial note: VLSI implementation of neural networks*. Proceedings of the International Joint Conference on Neural Networks, Seattle, Washington, USA.
- Fernandes, R. A. S. (2009); *Identificação de fontes de correntes harmônicas por rede neural artificial*. Dissertação de mestrado (engenharia elétrica), Universidade de São Paulo, São Carlos.
- Finnoff, W., Hergert, F., Zimmermann, H. G. (1993); *Improving model selection by nonconvergent methods*. Neural Networks, vol. 6, n°. 6, pp. 771-783.
- Forrester, F. D., Hagan, M. T. (1997); *Generalization approximation to Bayesian regularization*. Proceedings of the International Joint Conference on Neural Networks, Hong Kong, China, pp. 1930-1935.
- Franco, N. M. B. (2006); *Cálculo essencial*. Prentice-Hall, Rio de Janeiro.
- Garcia, C., Moreno, J. A. (2004); *The Hopfield associative memory network: improving performance with the forced "bid"*. Lecture notes in Computer Science (iberamis'2004), vol. 3315, pp. 871-880.
- Glover, F., Laguna, M. (1998); *Tabu search*. Springer, London, UK.
- Goldberg, D. E. (1989); *Genetic algorithms in search, optimization and machine learning*. Addison Wesley, Reading, Massachusetts, USA.
- Goldberg, D. E. (2002); *The design of innovation (genetic algorithms and evolutionary computation)*. Kluwer Academic Publishers, Dordrecht, The Netherlands.
- Graham, A. (1986); *Kronecker products and matrix calculus with applications*. Ellis Horwood Ltd, reprint edition, Chichester, UK.
- Grossberg, S. (1974); *Classical and instrumental learning by neural networks*. In: Progress in Theoretical Biology, vol. 3, pp. 51-141. Academic Press, New York, USA.
- Grossberg, S. (1976a); *Adaptive pattern classification and universal reading: Part I (Parallel development and coding of neural feature detectors)*. Biological Cybernetics, vol. 23, n°. 3, pp. 121-134.
- Grossberg, S. (1976b); *Adaptive pattern classification and universal reading: Part II (Feedback, expectation, affection, and illusions)*. Biological Cybernetics, vol. 23, n°. 4, pp. 187-202.
- Grossberg, S. (1980); *How does a brain build a cognitive code?*. Psychological Review, vol. 87, pp. 1-51.
- Hagan, M. T., Menhaj, M. B. (1994); *Training feedforward networks with the Marquardt algorithm*. IEEE Transactions on Neural Networks, vol. 5, n°. 6, pp. 989-993.
- Hagan, M. T., Demuth, H. B., Beale, M. H. (1996); *Neural network design*. PWS Publishing, Boston, Massachusetts, USA.
- Hampshire II, J. B., Pearlmanter, B. (1991); *Equivalent proofs for multi-layer perceptron classifiers and the Bayesian discriminant function*. In: Connectionist Models: Proceedings of the 1990 Summer School, Morgan Kaufmann, San Mateo, California, USA.

- Hartigan, J. A. (1975); *Clustering algorithms*. John Wiley & Sons, New York, USA.
- Haykin, S. (1999); *Neural networks: a comprehensive foundation*. Prentice Hall, 2<sup>nd</sup> edition, Upper Saddle River, New Jersey, USA.
- He, Y., Li, X., Deng, X. (2007); *Discriminative features of tea using near infrared spectroscopy by principal component analysis and BP model*. Journal of Food Engineering, vol. 79, n°. 4, pp. 1238-1242.
- Hebb, D. O. (1949); *The organization of behavior: a neurophysiological theory*. Wiley, New York, USA.
- Hecht-Nielsen, R. (1987a); *Counterpropagation networks*. Applied Optics, vol. 26, n°. 23, pp. 4979-4984.
- Hecht-Nielsen, R. (1987b); *Counterpropagation networks*. Proceedings of the IEEE First International Conference on Neural Networks, San Diego, California, vol. 2, pp. 19-32.
- Hodgkin, A. L., Huxley, A. F. (1952); *A quantitative description of membrane current and its application to conduction and excitation in nerve*. Journal of Physiology, vol. 117, pp. 500-544.
- Hopfield, J. J. (1982); *Neural network and physical systems with emergent collective computational abilities*. Proceedings of the National Academy of Sciences of the USA, vol. 79, n°. 8, pp. 2554-2558.
- Hopfield, J. J. (1984); Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Sciences of the USA, vol. 81, no. 10, pp. 3085-3092.
- Hopfield, J. J., Tank, D. W. (1985); *Neural computation of decisions in optimization problems*. Biological Cybernetics, vol. 52, n°. 3, pp. 141-152.
- Huang, Y., Yang, X.-S. (2006); *Chaos and bifurcation in a new class of simple Hopfield neural network*. Lecture Notes in Computer Science (ISBN 32006), vol. 3971, pp. 316-321.
- Jang, J.S.R., Sun, C.T., Mizutani, E. (1997); *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*. Prentice-Hall, Upper Saddle River, New Jersey, USA.
- Jordan, M. I. (1986); *Attractor dynamics and parallelism in a recurrentist sequential machine*. Proceedings of the 8<sup>th</sup> Annual Conference on Cognitive Science, pp. 331-346.
- Kandel, E. R., Schwartz, J. H., Jessell, T. M. (2000); *Principles of neural science*. McGraw-Hill, 4<sup>th</sup> Edition, New York, USA.
- Kennedy, J., Eberhart, R. C. (2001); *Swarm intelligence*. Morgan Kaufmann Publishers, San Diego, California, USA.
- Khan, J., Wei, J. S., Ringnér, M., Saal, L. H., Ladanyi, M., Westermann, P., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C., Meltzer, P.S. (2001); *Classification and diagnostic prediction of cancer using gene expression profiling and artificial neural networks*. Nature Medicine, vol. 7, n°. 6, pp. 673-679.
- Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. (1983); *Optimization by simulated annealing*. Science, vol. 220, n°. 4598, pp. 671-680.
- Kohavi, R. (1995); *A study of cross-validation and bootstrap for accuracy estimation and model selection*. Proceedings of the 14<sup>th</sup> International Joint Conference on Artificial Intelligence, Montreal, Canada, pp. 1137-1145.
- Kohonen, T. (1982); *Self-organized formation of topologically correct feature maps*. Biological Cybernetics, vol. 43, pp. 59-69.
- Kohonen, T. (1984); *Self-organization and associative memory*. Springer-Verlag, London, UK.
- Kohonen, T. (1990); *Improved version of learning vector quantization*. Proceedings of the International Joint Conference on Neural Networks, San Diego, California, pp. 545-550.
- Kohonen, T. (1997); *Self-organizing maps*. Springer-Verlag, 2<sup>nd</sup> edition, Berlin, Germany.

- Kolmogorov, A. N. (1957); *On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition*. Doklady Akademii Nauk USSR, vol. 114, pp. 679-681.
- Kunze, J. P., Ross, K. W. (2006); *Rodar de computadores e a internet*. Addison-Wesley, 3<sup>rd</sup> edition, São Paulo.
- Lang, K.J., Hinton, G.E. (1988); *The development of the time-delay neural network architecture for speech recognition*. Technical report CMU-CS-88-152, Carnegie-Mellon University, Pittsburgh, Pennsylvania, USA.
- LeCun, Y. (1989); *Generalization and network design strategies*. Technical Report CRG-TR-89-4, Department of Computer Science, University of Toronto, Canada.
- Lee, D.-L. (2006); *Improvement of complex-valued Hopfield association memory by using generalized projection rule*. IEEE Transactions on Neural Networks, vol. 17, n°. 5, pp. 1341-1347.
- Lee, S.-W., Kim, J.-S. (1995); *Multi-channel, multi-font and multi-size large-set character recognition using self-organizing neural network*. Proceedings of the Third International Conference on Document Analysis and Recognition, Montreal, Canada, pp. 28-33.
- Leondes, T. L. (2006); *Control and dynamic systems - neural network systems techniques and applications*. Academic Press, San Diego, California, USA.
- Li, J.-H., Michel, A. N., Porod, W. (1989); *Analysis and synthesis of a class of neural networks: linear gates operating on a closed hypercube*. IEEE Transactions on Circuits and Systems, vol. 36, n°. 11, pp. 1405-1422.
- Lin, Z., Khorsandi, K., Patel, R. V. (1990); *A counter-propagation neural network for function approximation*. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Los Angeles, California, USA, pp. 362-364.
- Lippmann, R.P. (1987); *An introduction to computing with neural nets*. IEEE ASSP Magazine, vol. 4, pp. 4-22.
- Ljung, L. (1998); *System identification: theory for the user*, Prentice Hall, 2<sup>nd</sup> edition, Upper Saddle River, New Jersey, USA.
- Lucenberger, D. G. (2003); *Linear and nonlinear programming*. Kluwer Academic Publishers, 2<sup>nd</sup> edition, Norwell, Massachusetts, USA.
- Lui, H. C. (1990); *Analysis of decision surface of neural network with sigmoidal nonlinearity*. Proceedings of the International Joint Conference on Neural Networks, Washington/DC, USA, pp. 655-658.
- Makhoul, J., El-Jaroudi, A., Schwartz, R. (1989); *Formation of disseminated decision regions with a single hidden layer*. Proceedings of the International Joint Conference on Neural Networks, Cambridge, Massachusetts, USA, pp. 455-460.
- Malek, A., Yari, A. (2005); *Primal-dual solution for the linear programming problems using neural networks*. Applied Mathematics and Computation, vol. 167, pp. 198-211.
- McCulloch, W. S., Pitts, W. (1943); *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133.
- Mendyk, A., Jachowicz, R. (2007); *Unified methodology of neural analysis in decision support systems built for pharmaceutical technology*. Expert Systems with Applications, vol. 32, n. 4, p. 1124-1131.
- Michalewicz, Z. (1999); *Genetic algorithms + data structures = evolution programs*. Springer Verlag, 3<sup>rd</sup> edition, Berlin, Germany.
- Minsky, M. L., Papert, S. A. (1969); *Perceptrons: an introduction to computational geometry*. The MIT Press, Cambridge, Massachusetts, USA.

- Mishra, D., Chandra Bose, N. S., Tolatibya, A., Dwivedi, A., Kandala, P., Kumar, A., Kalra, P. K. (2006); *Color image compression with modified forward-only multipropagative neural network: improvement of the quality using different distance measure*. Proceedings of the 9<sup>th</sup> International Conference on Information Technology, Bhubanerwar, India, pp. 139-140.
- Moon, T. K. (2005); *Error corrective coding – mathematical methods and algorithms*. John Wiley, New Jersey, USA.
- Morre, J. P., Diaz, S. S. (2003); *The DAPPN: a new neural network and circuit simulation for optical character recognition*. IEEE Transactions on Signal Processing, vol. 51, n°. 12, pp. 3198-3209.
- Muezzinoglu, M. K., Guzelis, C., Zurada, J. M. (2005); *An energy function-based design method for discrete Hopfield associative memory with attractive fixed points*. IEEE Transactions on Neural Networks, vol. 16, n°. 1, pp. 370-378.
- Narendra, K. S., Parthasarathy, K. (1990); *Identification and control of dynamical systems using neural networks*. IEEE Transactions on Neural Networks, vol. 1, n°. 1, pp. 4-27.
- Nazario, S. L. S. (2007); *Construção de árvores utilizando técnicas de ultra-som e redes neurais*. Dissertação de mestrado (engenharia elétrica), Universidade Estadual Paulista, Ilha Solteira.
- Nelles, O. (2005); *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer-Verlag, Berlin, Germany.
- Norgaard, M., Ravn, O., Poulsen, N. K., Hansen, L. K. (2006); *Neural networks for modelling and control of dynamic systems*. Springer-Verlag, Berlin, Germany.
- Nowlan, S. J. (1989); *Maximally likelihood competitive learning*. In: Advances in Neural Information Processing Systems 2, pp. 574-582, Morgan Kaufmann, San Mateo, California, USA.
- Ortega, A. V., Silva, I. N. (2008); *Neural network model for designing automotive drives using SMID* IJETD International Journal of Automotive Technology, vol. 9, n°. 2, pp. 203-210.
- Otsu, N. (1979); *A threshold selection method from gray-level histograms*. IEEE Transactions on Systems, Man and Cybernetics, vol. 9, n°. 1, pp. 62-66.
- Park, J., Sandberg, I. W. (1991); *Universal approximation using radial-basis-function networks*. Neural Computation, vol. 3, pp. 246-257.
- Parker, J. R. (1996); *Algorithms for image processing and computer vision*. John Wiley & Sons, New York, USA.
- Parsons, S., Jones, G. (2000); *Arrestive identification of twelve species of teleosts by discriminant function analysis and artificial neural networks*. Journal of Experimental Biology, vol. 203, n°. 17, pp. 2641-2656.
- Pedrycz, W., Gorzal, E. (2007); *Fuzzy systems engineering – toward human-centric computing*. John Wiley & Sons, Hoboken, New Jersey, USA.
- Pinca, M. (1970); *A Monte Carlo method for the approximate solution of certain types of constrained optimization problems*. Operation Research, vol. 18, pp. 1225-1228.
- Powell, M. J. D. (1987); *Radial basis functions for multivariable interpolation: a review*. In: Algorithms for Approximation, pp. 143-167, Oxford University Press, Oxford, UK.
- Rajasekaran, S., Raj, R. A. (2004); *Image recognition using analog-ART1 architecture augmented with moment-based feature extractors*. Neurocomputing, vol. 56, pp. 61-77.
- Reed, R. D., Marks II, R. J. (1999); *Neural smoothers: supervised learning in feedforward artificial neural networks*. MIT Press, Cambridge, Massachusetts, USA.
- Richter, T. (2009); *Arquitetura de sistemas inteligentes para treinamento virtual de exames em medicina biomédica*. Dissertação de mestrado (engenharia elétrica), Universidade de São Paulo, São Carlos.

- Riedmiller, M., Braun, H. (1993); *A direct adaptive method for faster backpropagation learning: the RPROP-algorithm*. Proceedings of the International Joint Conference on Neural Networks, San Francisco, California, USA, pp. 586-591.
- Ripley, B. D. (1996); *Pattern recognition and neural networks*. Cambridge University Press, Cambridge, UK.
- Ritter, H., Martinez, T., Schulten, K. (1997); *Neural computation and self-organizing maps - an introduction*. Addison-Wesley, New York, USA.
- Rosenthal, F. (1958); *The perceptron: a probabilistic model for information storage and organization in the brain*. Psychological Review, vol. 65, pp. 386-408.
- Ross, T.J. (2004); *Fuzzy logic with engineering applications*. John Wiley, New York, USA.
- Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986); *Learning internal representations by error propagation*. In: Parallel Distributed Processing, vol. 1, chapter 8. MIT Press, Cambridge, Massachusetts, USA.
- Schacter, G. M. (1990); *The cognitive organization of the brain*. Oxford University Press, 3<sup>rd</sup> edition, New York, USA.
- Silva, C. B. S. (2007); *Pracmamento de sinal de resistencia magnetica nuclear usando classificador neural para reconhecimento de corne biliaria*. Dissertação de mestrado (engenharia elétrica), Universidade de São Paulo, São Carlos.
- Silva, I. N. (1997); *Linha alternativa avançada para estimativa de sistemas e identificação robusta*. Tese de doutorado (engenharia elétrica), Universidade Estadual de Campinas, Campinas.
- Silva, I. N., Amaro, W. C., Arruda, L. V. R. (2007); *A novel approach based on recurrent neural networks applied to nonlinear systems optimization*. Applied Mathematical Modelling, vol. 31, pp. 78-92.
- Silva, I. N., Arruda, L. V. R., Amaro, W. C. (2001); *An efficient model of neural networks for dynamic programming*. International Journal of Systems Science, vol. 32, n°. 6, pp. 715-722.
- Sontag, E. D. (1992); *Feedback stabilization using two-hidden-layer nets*. IEEE Transactions on Neural Networks, vol. 3, n°. 6, pp. 981-990.
- Srinivasan, A., Batur, C. (1994); *Hopfield/ART-1 neural network-based fault detection and isolation*. IEEE Transactions on Neural Networks, vol. 5, n°. 6, pp. 890-899.
- Szczepanski, H. D. (1956); *Caractérisation des corps matériels en partie*. Bulletin L'Academie Polonaise des Sciences, vol. IV, n°. C1-III, pp. 801-804.
- Sutton, R. S., Barto, A. G. (1998); *Reinforcement learning: an introduction*. MIT Press, Cambridge, Massachusetts, USA.
- Suykens, J., Vandewalle, J. P. L., De Moor, B. L. (2001); *Artificial neural networks for modelling and control of non-linear systems*. Springer-Verlag, Berlin, Germany.
- Szygnowski, W., Macukow, B. (1996); *Backpropagation neural network for image compression*. Optical Engineering, vol. 35, n°. 8, pp. 2214-2217.
- Tibach, E. E., Lancelot, L., Shahrouz, I., Najjar, V. (2007); *Use of artificial neural network simulation methodology to assess groundwater contamination in a real project*. Mathematical and Computer Modelling, vol. 45, n°. 7-8, pp. 766-776.
- Tank, D. W., Hopfield, J. J. (1986); *Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit*. IEEE Transactions on Circuits and Systems, vol. 33, n°. 5, pp. 533-541.
- Taniguchi, T., Kawata, S. (2004); *Plant layout planning using ART neural networks*. Proceedings of the SICE 2004 Annual Conference, Sapporo, Japan, pp. 1863-1868.

- Tian, H., Shang, Z. (2006); *Artificial neural network as a classification method of mice by their calls*. Ultrasonics, vol. 44, pp. e273-e278, 2006.
- Tso, C.-P. J., Chen, S. R., Hwang, S. (2007); *Inversion for acoustic impedance of a wall by using artificial neural network*. Applied Acoustics, vol. 68, n°. 4, pp. 377-389.
- Tsoukalas, L. H., Uhrig, R. E. (1997); *Fuzzy and neural approaches in engineering*. John Wiley, New York, USA.
- Vakil-Baghmisheh, M.-T., Paveiic, N. (2003); *Probability clustering phenomena and new training algorithms for L1/Q*. Pattern Recognition, vol. 36, n°. 8, pp. 1901-1912.
- Vapnik, V. N. (1998); *Statistical learning theory*. John Wiley, New York, USA.
- Vicente, B. G. L. Z., Cesare, M. J., Silva, I. N. (2007); *Controlador neural de marcha lenta para motores de combustão interna*. Anais do VIII Simpósio Brasileiro de Automação Inteligente, Florianoópolis, CD-ROM (paper Nº. 30909, 6 páginas).
- Vidyasagar, M. (2002); *Nonlinear systems analysis*. SIAM Publications, 2<sup>nd</sup> edition, Philadelphia, Pennsylvania, USA.
- Walibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K. (1987); *Speaker recognition using time-delay neural networks*. IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 37, n°. 3, pp. 328-339.
- Wang, J. (2004); *A recurrent neural network for solving the shortest path problem*. IEEE Transactions on Circuits and Systems - Part I, vol. 43, n°. 6, pp. 482-486.
- Wasserman, P. D. (1989); *Neural computing - theory and practice*. Van Nostrand Reinhold, New York, USA.
- Watkins, C. J. C. H. (1989); *Learning from delayed rewards*. Ph.D. Thesis, Psychology Department, Cambridge University, UK.
- Werbos, P. J. (1974); *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. Ph.D. thesis, Harvard University, Cambridge, Massachusetts, USA.
- Widrow, B., Hoff, M. E. (1960); *Adaptive switching circuits*. Proceedings of the IRE Wexcon Convention Record, pp. 96-104.
- Widrow, B., Winter, R. (1968); *Neural nets for adaptive filtering and adaptive pattern recognition*. IEEE Computer Magazine, vol. 21, n°. 3, pp. 25-39.
- Wirth, N. (1989); *Algoritmos e estruturas de dados*. Prentice-Hall, Rio de Janeiro.
- Witten, I. H., Frank, E. (2005); *Data mining - practical machine learning tools and techniques*. Morgan Kaufmann, 2<sup>nd</sup> edition, San Francisco, California, USA.
- Xia, Y., Wang, J. (2004); *A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints*. IEEE Transactions on Circuits and Systems - Part I, vol. 51, n°. 7, pp. 1385-1394.
- Xiang, C., Ding, S. Q., Lee, T. H. (2005); *Geometrical interpretation and architecture selection of ANN*. IEEE Transactions on Neural Networks, vol. 16, no. 1, pp. 84-96.
- Yan, H., Jiang, Y., Zheng, J., Peng, C., Li, Q. (2006); *A multilayer perceptron-based medical decision support system for breast disease diagnosis*. Expert Systems with Applications, vol. 30, n°. 2, pp. 272-281.
- Yang, C.-H., Luo, C.-H., Yang, C.-H., Chuang, L.-Y. (2004); *Counterpropagation network with variable degree neurons step size LMS for single switch typing recognition*. Bio-medical Materials and Engineering, vol. 14, n°. 1, pp. 23-32.
- Young, R. J., Rothaler, M., Zimmer, P., McGraw, J., Healy, M. J., Caudell, T. P. (2007); *Comparative*

- of adaptive resonance theory neural networks for astronomical regions of interest detection and noise denoising. Proceedings of the International Joint Conference on Neural Networks, Albuquerque, New Mexico, USA, pp. 2123-2128.
- Zadeh, L. (1992); *Fuzzy logic, neural networks and soft computing*. Proceedings of the 2<sup>nd</sup> International Conference on Fuzzy Logic and Neural Networks, Iizuka, Japan, pp. 13-14.
- Zhang, Q., Stanley, S. J. (1999); *Real-time water treatment process control with artificial neural networks*. Journal of Environmental Engineering, vol. 125, n°. 2, pp. 153-160.
- Zhang, Q., Xu, X. W. J. (2007); *Delay-dependent global stability results for delayed Hopfield neural networks*. Chaos, Solitons & Fractals, vol. 34, n°. 2, pp. 662-668.
- Zhang, Q.-B., Hebsa, R. J., Zhang, Q. J., Alfaro, R. I. (2003); *Modeling tree-ring growth responses to climatic variables using artificial neural networks*. Journal of Forest Science, vol. 49, n°. 2, pp. 229-239.
- Zhang, Z., Friedrich, K. (2003); *Artificial neural networks applied to polymer composites: a review*. Elsevier Science, Composites Science and Technology, vol. 63, n°. 14, pp. 2029-2044.



## Apêndice I

Conjunto de treinamento referente à seção 3.6<sup>(1)</sup>

Amostra	$x_1$	$x_2$	$x_3$	$d$
01	-0,6508	0,1097	4,0009	-1,0000
02	-1,4492	0,8896	4,4005	-1,0000
03	2,0850	0,6876	12,0710	-1,0000
04	0,2626	1,1476	7,7985	1,0000
05	0,6418	1,0234	7,0427	1,0000
06	0,2569	0,6730	8,3265	-1,0000
07	1,1155	0,8043	7,4446	1,0000
08	0,0914	0,3389	7,0677	-1,0000
09	0,0121	0,5256	4,6316	1,0000
10	-0,0429	0,4660	5,4323	1,0000
11	0,4340	0,6870	8,2287	-1,0000
12	0,2735	1,0287	7,1934	1,0000
13	0,4839	0,4851	7,4850	-1,0000
14	0,4089	-0,1267	5,5019	-1,0000
15	1,4391	0,1614	8,5843	-1,0000
16	-0,9115	-0,1973	2,1962	-1,0000

<sup>(1)</sup> Tabelas disponíveis em <http://lages.scl.esc.usp.br>

Amostra	$x_1$	$x_2$	$x_3$	$d$
17	0,3654	1,0475	7,4858	1,0000
18	0,2144	0,7515	7,1699	1,0000
19	0,2013	1,0014	6,5489	1,0000
20	0,6483	0,2183	5,8991	1,0000
21	-0,1147	0,2242	7,2435	-1,0000
22	-0,7970	0,8795	3,8762	1,0000
23	-1,0625	0,6366	2,4707	1,0000
24	0,5307	0,1285	5,6883	1,0000
25	-1,2200	0,7777	1,7252	1,0000
26	0,3957	0,1076	5,6623	-1,0000
27	-0,1013	0,5989	7,1812	-1,0000
28	2,4482	0,9455	11,2095	1,0000
29	2,0149	0,6192	10,9263	-1,0000
30	0,2012	0,2611	5,4631	1,0000

## Apêndice II

Conjunto de treinamento referente à seção 4.6<sup>(\*)</sup>

Amostra	$x_1$	$x_2$	$x_3$	$x_4$	$d$
01	0,4329	-1,3719	0,7022	-0,8535	1,0000
02	0,3024	0,2286	0,8630	2,7909	-1,0000
03	0,1349	-0,6445	1,0530	0,5687	-1,0000
04	0,3374	-1,7163	0,3670	-0,6283	-1,0000
05	1,1434	-0,0485	0,6637	1,2606	1,0000
06	1,3749	-0,5071	0,4464	1,3009	1,0000
07	0,7221	-0,7587	0,7661	-0,5592	1,0000
08	0,4403	-0,8072	0,5154	-0,3129	1,0000
09	-0,5231	0,3548	0,2538	1,5776	-1,0000
10	0,3255	-2,0000	0,7112	-1,1209	1,0000
11	0,5824	1,3915	-0,2291	4,1735	-1,0000
12	0,1340	0,6081	0,4450	3,2230	-1,0000
13	0,1480	-0,2988	0,4778	0,8649	1,0000
14	0,7359	0,1869	-0,0872	2,3584	1,0000
15	0,7115	-1,1469	0,3394	0,9573	-1,0000
16	0,6251	-1,2840	0,8452	1,2382	-1,0000

<sup>(\*)</sup> Tabelas disponíveis em <http://lipsel.oeic.ufsc.br>

Amostra	$x_1$	$x_2$	$x_3$	$x_4$	$d$
17	0,1569	0,3712	0,8825	1,7633	1,0000
18	0,0033	0,6835	0,5389	2,8249	-1,0000
19	0,4243	0,8313	0,2634	3,5855	-1,0000
20	1,0490	0,1326	0,9138	1,9792	1,0000
21	1,4276	0,5331	-0,0145	3,7286	1,0000
22	0,5971	1,4865	0,2904	4,6069	-1,0000
23	0,8475	2,1479	0,3179	5,6235	-1,0000
24	1,3967	-0,4171	0,6443	1,3927	1,0000
25	0,0044	1,5378	0,6099	4,7755	-1,0000
26	0,2201	-0,5668	0,0515	0,7829	1,0000
27	0,6300	-1,2480	0,8591	0,8093	-1,0000
28	-0,2479	0,8960	0,0547	1,7381	1,0000
29	-0,3088	-0,0929	0,8859	1,5483	-1,0000
30	-0,5180	1,4974	0,5453	2,3993	1,0000
31	0,6833	0,8266	0,0829	2,8864	1,0000
32	0,4353	-1,4066	0,4207	-0,4879	1,0000
33	-0,1069	-3,2329	0,1856	-2,4572	-1,0000
34	0,4662	0,6261	0,7304	3,4370	-1,0000
35	0,8298	-1,4089	0,3119	1,3235	-1,0000

## Apêndice III

### Conjunto de treinamento referente à seção 5.8<sup>(1)</sup>

Amostra	$x_1$	$x_2$	$x_3$	$d'$	Amostra	$x_1$	$x_2$	$x_3$	$d'$
1	0,8799	0,7998	0,3972	0,8399	101	0,8293	0,6682	0,4260	0,5843
2	0,5700	0,5111	0,2418	0,6258	102	0,6226	0,2148	0,1021	0,4452
3	0,6796	0,4117	0,3370	0,6622	103	0,4889	0,3147	0,2236	0,4962
4	0,3567	0,2967	0,6037	0,5969	104	0,3471	0,8889	0,1564	0,5875
5	0,3866	0,8990	0,6292	0,5355	105	0,5782	0,8292	0,4116	0,7850
6	0,6271	0,7788	0,7445	0,6335	106	0,9053	0,6245	0,5264	0,8506
7	0,8174	0,8422	0,3229	0,8068	107	0,2960	0,0793	0,0549	0,2224
8	0,6027	0,1468	0,3759	0,5342	108	0,9667	0,3634	0,4425	0,6990
9	0,1203	0,3260	0,5419	0,4768	109	0,5170	0,8266	0,1565	0,6594
10	0,1305	0,2082	0,4934	0,4105	110	0,8149	0,0398	0,6227	0,6165
11	0,6960	1,0000	0,4321	0,6404	111	0,3710	0,3534	0,5633	0,6171
12	0,6036	0,1940	0,3274	0,3697	112	0,8702	0,3185	0,2782	0,6287
13	0,2650	0,0161	0,5947	0,4125	113	0,1018	0,6382	0,3173	0,4957
14	0,5849	0,6019	0,4378	0,7464	114	0,3890	0,2369	0,0082	0,3235
15	0,6108	0,3638	0,1810	0,2860	115	0,2702	0,8817	0,1218	0,5319
16	0,9008	0,7264	0,9154	0,9602	116	0,7473	0,6907	0,5682	0,8484
17	0,0023	0,9659	0,3182	0,4986	117	0,9128	0,2139	0,4641	0,6625
18	0,1366	0,6357	0,5967	0,6459	118	0,4343	0,6028	0,1344	0,5546

<sup>(1)</sup> Tabelas disponíveis em <http://biometria.uol.com.br>.

Amostra	$x_1$	$x_2$	$x_3$	$\sigma$	Amostra	$x_1$	$x_2$	$x_3$	$\sigma$
19	0,6621	0,7363	0,2742	0,7718	119	0,6547	0,4062	0,6318	0,8204
20	0,6682	0,6624	0,4211	0,5764	120	0,6697	0,6443	0,6900	0,6904
21	0,6113	0,6014	0,5254	0,7668	121	0,4911	0,4138	0,6715	0,7222
22	0,6630	0,7365	0,6928	0,6368	122	0,5949	0,2600	0,6810	0,4480
23	0,7644	0,5984	0,6407	0,2053	123	0,1845	0,7908	0,8925	0,7425
24	0,6441	0,2697	0,5847	0,6546	124	0,3438	0,6725	0,9621	0,7305
25	0,6603	0,3799	0,6929	0,4691	125	0,8088	0,1362	0,8119	0,7222
26	0,1908	0,6046	0,5402	0,6865	126	0,2345	0,0071	0,6156	0,4452
27	0,6637	0,3967	0,6055	0,7595	127	0,3742	0,6666	0,8194	0,8371
28	0,2591	0,0562	0,3978	0,3804	128	0,9572	0,9836	0,3793	0,8566
29	0,4341	0,1889	0,6686	0,6298	129	0,7496	0,0410	0,1360	0,4089
30	0,5332	0,9303	0,2475	0,6287	130	0,9129	0,3510	0,0680	0,9495
31	0,3625	0,1992	0,9981	0,5948	131	0,6654	0,5900	0,6601	0,8388
32	0,6259	0,6960	0,1645	0,4716	132	0,5393	0,6529	0,5729	0,7880
33	0,8806	0,6779	0,0033	0,6242	133	0,3156	0,3851	0,5963	0,6161
34	0,6838	0,5472	0,3758	0,4835	134	0,1460	0,1637	0,0249	0,1813
35	0,6303	0,5151	0,7253	0,6491	135	0,7789	0,4491	0,4614	0,7496
36	0,9293	0,8319	0,9664	0,9640	136	0,5929	0,6647	0,6601	0,9176
37	0,7268	0,1440	0,9793	0,7096	137	0,2204	0,1780	0,4627	0,4276
38	0,2888	0,6693	0,4078	0,6326	138	0,7355	0,6264	0,7015	0,9214
39	0,5915	0,1384	0,2694	0,4745	139	0,9891	0,6727	0,3139	0,7829
40	0,7683	0,0067	0,5846	0,5708	140	0,9123	0,0000	0,1106	0,3044
41	0,6462	0,6761	0,6340	0,6930	141	0,2868	0,6688	0,3282	0,5998
42	0,3684	0,2212	0,1233	0,3658	142	0,7031	0,6660	0,6038	0,8728
43	0,2706	0,3232	0,9996	0,6310	143	0,7641	0,6778	0,6012	0,6892
44	0,6282	0,1404	0,8474	0,6733	144	0,1360	0,5861	0,2867	0,4622
45	0,5861	0,6693	0,3818	0,7433	145	0,6345	0,5165	0,7139	0,8591
46	0,6057	0,6901	0,5141	0,6486	146	0,2453	0,5888	0,1059	0,4765
47	0,5915	0,5888	0,3068	0,6787	147	0,1134	0,5436	0,3687	0,4983
48	0,6359	0,4145	0,6016	0,7567	148	0,3687	0,3228	0,6952	0,8378
49	0,5497	0,6319	0,8382	0,6521	149	0,6537	0,6948	0,4451	0,8426
50	0,7672	0,1721	0,3812	0,5772	150	0,7904	0,6348	0,6449	0,8876
51	0,1185	0,5984	0,8378	0,6211	151	0,1427	0,6480	0,6267	0,3780
52	0,6365	0,6562	0,4965	0,7693	152	0,1516	0,9524	0,6827	0,6127
53	0,4145	0,5797	0,8999	0,7878	153	0,4963	0,6223	0,7462	0,8116
54	0,3675	0,5358	0,4028	0,5777	154	0,3408	0,6115	0,0783	0,4559
55	0,2026	0,3300	0,3054	0,4061	155	0,8146	0,6378	0,5837	0,8628
56	0,3085	0,6476	0,5941	0,4625	156	0,2820	0,5469	0,7256	0,6939
57	0,4694	0,1726	0,7863	0,6015	157	0,8716	0,2958	0,6477	0,8619
58	0,1261	0,6181	0,4937	0,5739	158	0,9323	0,6229	0,4797	0,5731

Amostra	$x_1$	$x_2$	$x_3$	$d$	Amostra	$x_1$	$x_2$	$x_3$	$d$
59	0,5224	0,4662	0,3146	0,4007	150	0,2907	0,7245	0,5165	0,6911
60	0,6793	0,6774	1,0000	0,9341	151	0,0068	0,0049	0,0061	0,0051
61	0,9179	0,0358	0,2506	0,4707	152	0,2638	0,9885	0,2175	0,5847
62	0,6907	0,6985	0,9975	0,8929	153	0,0360	0,3693	0,7901	0,5117
63	0,2404	0,5411	0,8754	0,8860	154	0,9670	0,3031	0,7127	0,7836
64	0,6553	0,2609	0,1188	0,4851	155	0,0000	0,7763	0,8735	0,6368
65	0,8888	0,0288	0,2604	0,4802	156	0,4395	0,0001	0,9761	0,5712
66	0,3974	0,5275	0,6457	0,7215	157	0,9359	0,0366	0,9514	0,6826
67	0,2109	0,4919	0,5432	0,5913	158	0,0173	0,9548	0,4269	0,5527
68	0,8873	0,5571	0,1849	0,6805	159	0,8112	0,9070	0,6286	0,8863
69	0,6693	0,0242	0,9293	0,6033	160	0,2010	0,9573	0,6791	0,7283
70	0,8439	0,4531	0,6345	0,8226	161	0,8914	0,9144	0,2641	0,7966
71	0,3844	0,2948	0,3937	0,5340	162	0,0001	0,0802	0,8621	0,3711
72	0,2014	0,6326	0,9782	0,7143	163	0,2212	0,4864	0,3621	0,6260
73	0,4039	0,0845	0,4629	0,4547	164	0,2401	0,8954	0,0751	0,4537
74	0,7137	0,0670	0,2359	0,4622	165	0,7881	0,9833	0,3038	0,8049
75	0,4277	0,9555	0,0000	0,5477	166	0,2435	0,0794	0,9551	0,4223
76	0,6269	0,7634	0,2859	0,4738	167	0,2752	0,8454	0,2797	0,6079
77	0,1571	0,7682	0,9697	0,7397	168	0,7616	0,4698	0,5337	0,7809
78	0,3218	0,5420	0,0677	0,4526	169	0,3086	0,0022	0,0087	0,1536
79	0,2524	0,7688	0,9523	0,7711	170	0,7849	0,9981	0,4449	0,8841
80	0,3621	0,5295	0,3521	0,5871	171	0,8312	0,0961	0,2129	0,4857
81	0,2942	0,1625	0,2745	0,3759	172	0,8763	0,1102	0,6237	0,6667
82	0,8183	0,0023	0,1438	0,4018	173	0,8587	0,3284	0,6932	0,7829
83	0,8429	0,1704	0,5291	0,6063	174	0,9295	0,3275	0,7536	0,8018
84	0,9812	0,6898	0,6636	0,9128	175	0,2435	0,2160	0,7625	0,5449
85	0,1009	0,4193	0,0626	0,3055	176	0,8281	0,8358	0,5285	0,8911
86	0,7071	0,7704	0,8328	0,8298	177	0,8313	0,7566	0,6102	0,9047
87	0,3371	0,7819	0,0659	0,5377	178	0,1712	0,0545	0,5033	0,3561
88	0,1568	0,9599	0,9221	0,6663	179	0,9609	0,1702	0,4306	0,3340
89	0,7358	0,1877	0,3311	0,5869	180	0,5899	0,9406	0,0369	0,6245
90	0,1665	0,7449	0,0697	0,4508	181	0,7858	0,5115	0,0916	0,6066
91	0,8762	0,2498	0,9167	0,7829	182	1,0000	0,1853	0,7103	0,7172
92	0,9885	0,6229	0,2686	0,7300	183	0,2997	0,1163	0,3431	0,3385
93	0,0461	0,7745	0,5632	0,5949	184	0,2306	0,0330	0,6293	0,1560
94	0,3209	0,6229	0,9233	0,6810	185	0,8477	0,8378	0,4623	0,8254
95	0,9189	0,5930	0,7288	0,8989	186	0,9677	0,7895	0,9467	0,9782
96	0,6362	0,5515	0,8818	0,5999	187	0,0080	0,4668	0,1528	0,3250
97	0,3726	0,9588	0,3814	0,7066	188	0,9885	0,8897	0,6175	0,9366
98	0,4219	0,2668	0,3307	0,5880	189	0,9609	0,1702	0,4306	0,3340

Amostra	$x_1$	$x_2$	$x_3$	$d$	Amostra	$x_1$	$x_2$	$x_3$	$d$
99	0,2378	0,0617	0,3574	0,3452	199	0,7408	0,5361	0,2732	0,6949
100	0,9893	0,7627	0,2526	0,7795	200	0,6643	0,3737	0,1962	0,5625

### Conjunto de treinamento referente à seção 5.9<sup>11</sup>

Amostra	$x_1$	$x_2$	$x_3$	$x_4$	$d_1$	$d_2$	$d_3$	Amostra	$x_1$	$x_2$	$x_3$	$x_4$	$d_1$	$d_2$	$d_3$
1	0,3841	0,2221	0,2000	0,3438	1	0	0	66	0,3367	0,4333	0,2338	0,1678	1	0	0
2	0,1763	0,1613	0,3401	0,2643	1	0	0	67	0,4744	0,4654	0,1807	0,4873	1	0	0
3	0,3179	0,5768	0,3087	0,4192	0	1	0	68	0,7810	0,4350	0,5453	0,4831	0	1	0
4	0,2467	0,0307	0,2099	0,3454	1	0	0	69	0,4049	0,5638	0,2534	0,5373	0	1	0
5	0,8102	0,6192	0,4679	0,4762	0	1	0	70	0,1449	0,1539	0,3446	0,5659	1	0	0
6	0,7050	0,7764	0,7482	0,6962	0	0	1	71	0,3460	0,2722	0,1466	0,5049	1	0	0
7	0,4797	0,4348	0,4852	0,3643	0	1	0	72	0,2341	0,2046	0,2675	0,2891	1	0	0
8	0,7589	0,8256	0,8914	0,6143	0	0	1	73	0,1412	0,2364	0,4625	0,3861	1	0	0
9	0,1549	0,3641	0,2551	0,2919	1	0	0	74	0,5762	0,6418	0,7212	0,6366	0	0	1
10	0,5561	0,5602	0,5805	0,2109	0	1	0	75	0,8163	0,6971	0,6229	0,6689	0	0	1
11	0,3267	0,2974	0,0343	0,1466	1	0	0	76	0,6914	0,7684	0,5385	0,5243	0	0	1
12	0,2363	0,0442	0,3889	0,1713	1	0	0	77	0,7228	0,5708	0,5812	0,7560	0	0	1
13	0,2993	0,2963	0,2600	0,3039	1	0	0	78	0,4270	0,5302	0,6811	0,3884	0	1	0
14	0,5797	0,4719	0,5280	0,3648	0	1	0	79	0,6189	0,1952	0,4918	0,3042	1	0	0
15	0,5560	0,5250	0,4762	0,4621	0	1	0	80	0,2143	0,3868	0,1198	0,0000	1	0	0
16	0,7049	0,6933	0,6449	0,6623	0	0	1	81	0,5896	0,7238	0,7199	0,6677	0	0	1
17	0,8134	0,8412	0,6078	0,5894	0	0	1	82	0,8656	0,6793	0,6870	0,6885	0	0	1
18	0,2033	0,4943	0,2525	0,2947	1	0	0	83	0,6962	0,6868	0,7409	0,7047	0	0	1
19	0,2628	0,4172	0,2775	0,2721	1	0	0	84	0,4167	0,6259	0,5906	0,4003	0	1	0
20	0,4850	0,5506	0,5289	0,6006	0	1	0	85	0,6025	0,4804	0,7900	0,7471	0	0	1
21	0,2434	0,2587	0,2312	0,2624	1	0	0	86	0,4124	0,1191	0,6720	0,3184	1	0	0
22	0,1260	0,3023	0,1826	0,2188	1	0	0	87	1,0000	1,0000	0,7104	0,7074	0	0	1
23	0,5598	0,4253	0,4258	0,3192	0	1	0	88	0,5685	0,6934	0,6180	0,5782	0	1	0
24	0,5798	0,7674	0,6154	0,4447	0	0	1	89	0,6505	0,4864	0,2972	0,4599	0	1	0
25	0,5662	0,5858	0,5832	0,4595	0	0	1	90	0,8124	0,7680	0,6720	1,0000	0	0	1
26	0,4650	0,7082	0,3221	0,2949	0	1	0	91	0,8613	0,7160	1,0000	0,8048	0	0	1
27	0,5568	0,7582	0,6293	0,5453	0	1	0	92	0,5872	0,7568	0,9307	0,6791	0	0	1
28	0,6642	0,7509	0,5723	0,5814	0	0	1	93	0,3798	0,2139	0,2136	0,4295	1	0	0
29	0,7909	0,9049	0,7339	0,7324	0	0	1	94	0,5150	0,4546	0,3715	0,4088	0	1	0
30	0,7124	0,7728	0,6005	0,6656	0	0	1	95	0,5768	0,6204	0,2044	0,4888	0	0	1
31	0,6749	0,8767	0,6543	0,7461	0	0	1	96	0,1864	0,2864	0,2090	0,3425	1	0	0
32	0,3674	0,4389	0,4230	0,2985	1	0	0	97	0,2495	0,2607	0,4879	0,2290	1	0	0
33	0,3473	0,3784	0,2183	0,7925	1	0	0	98	0,2487	0,2048	0,0913	0,1281	1	0	0
34	0,6931	0,5168	0,5286	0,5794	0	1	0	99	0,5748	0,6652	0,6473	0,7317	0	0	1
35	0,6439	0,4959	0,4322	0,4562	0	1	0	100	0,3858	0,7585	0,3279	0,3588	0	1	0
36	0,5627	0,4993	0,6831	0,5120	0	1	0	101	0,3329	0,4946	0,5814	0,3152	0	1	0
37	0,5162	0,7659	0,6368	0,4536	0	1	0	102	0,3891	0,4809	0,7598	0,4231	0	1	0

<sup>11</sup> Tabelas disponíveis em <http://luisclaudio.usp.br>.

Amostra	$x_1$	$x_2$	$x_3$	$x_4$	$d_1$	$d_2$	$d_3$	Amostra	$x_1$	$x_2$	$x_3$	$x_4$	$d_1$	$d_2$	$d_3$
38	0,0046	0,2479	0,6642	0,4378	0	1	0	103	0,2688	0,4686	0,1930	0,0177	1	0	0
39	0,6038	0,6786	0,7751	0,6183	0	0	1	104	0,3657	0,4900	0,2372	0,3689	0	1	0
40	0,2429	0,4694	0,2855	0,2977	1	0	0	105	0,6047	0,6234	0,6274	0,3609	0	1	0
41	0,6371	0,5669	0,5316	0,4920	0	1	0	106	0,8440	0,7031	0,6469	0,4791	0	0	1
42	0,4288	0,5970	0,6467	0,7677	0	0	1	107	0,6554	0,6785	0,5279	0,7723	0	0	1
43	0,3629	0,5504	0,3796	0,4826	0	1	0	108	0,6468	0,3688	0,6649	0,3762	1	0	0
44	0,4302	0,3207	0,6397	0,4319	0	1	0	109	0,6154	0,6196	0,6309	0,7729	0	0	1
45	0,7078	0,8604	0,7470	0,6399	0	0	1	110	0,8402	0,8887	0,8383	0,6961	0	0	1
46	0,7350	0,8176	0,7327	0,6278	0	0	1	111	0,6927	0,7870	0,7689	0,7213	0	0	1
47	0,3611	0,2948	0,6625	0,4312	0	1	0	112	0,4032	0,6188	0,4930	0,5380	0	1	0
48	0,5661	0,3817	0,6363	0,3683	0	1	0	113	0,4606	0,3694	0,3868	0,2611	1	0	0
49	0,6066	0,2563	0,2603	0,3627	1	0	0	114	0,7416	0,7138	0,6803	0,6957	0	0	1
50	0,5996	0,5704	0,6985	0,6548	0	0	1	115	0,7454	0,6764	0,6293	0,4684	0	0	1
51	0,4269	0,3709	0,3994	0,3658	0	1	0	116	0,7735	0,7087	0,6626	0,6142	0	0	1
52	0,2060	0,3658	0,3204	0,1802	1	0	0	117	0,5803	0,9835	0,3708	0,5838	0	1	0
53	0,2335	0,2656	0,3912	0,1601	1	0	0	118	0,2081	0,3738	0,3119	0,3532	1	0	0
54	0,3266	0,7751	0,4356	0,3448	0	1	0	119	0,5676	0,8972	0,5168	0,6850	0	0	1
55	0,3437	0,1020	0,1238	0,2206	1	0	0	120	0,6584	0,8967	0,6060	0,7157	0	0	1
56	0,4656	0,4815	0,4211	0,4862	0	1	0	121	0,3879	0,3670	0,3637	0,1220	1	0	0
57	0,7511	0,8868	0,5408	0,6253	0	0	1	122	0,2644	0,6960	0,3872	0,1931	1	0	0
58	0,7625	0,9086	0,6610	0,6996	0	0	1	123	0,4895	0,4791	0,4213	0,5859	0	1	0
59	0,3483	0,4118	0,2807	0,0454	1	0	0	124	0,0848	0,0249	0,4349	0,3328	1	0	0
60	0,5172	0,1482	0,3172	0,2323	1	0	0	125	0,4668	0,8778	0,3533	0,3616	0	1	0
61	0,6942	0,4578	0,5387	0,5983	0	1	0	126	0,4155	0,6569	0,5310	0,5404	0	1	0
62	0,7588	0,7017	0,7129	0,7949	0	0	1	127	0,3894	0,5244	0,4817	0,4324	0	1	0
63	0,6689	0,6664	0,6602	0,4829	0	1	0	128	0,5843	0,8517	0,8576	0,7133	0	0	1
64	0,4742	0,5979	0,4135	0,4161	0	1	0	129	0,1995	0,3690	0,3537	0,3462	1	0	0
65	0,4459	0,5791	0,4515	0,4487	0	1	0	130	0,3632	0,3221	0,0341	0,2450	1	0	0

### Conjunto de treinamento referente à seção 5.10<sup>(1)</sup>

Amostra	$\tilde{f}(t)$	Amostra	$\tilde{f}(t)$	Amostra	$\tilde{f}(t)$	Amostra	$\tilde{f}(t)$
$t = 1$	0,1701	$t = 26$	0,2398	$t = 51$	0,3087	$t = 76$	0,3701
$t = 2$	0,1023	$t = 27$	0,0508	$t = 52$	0,0159	$t = 77$	0,0008
$t = 3$	0,4405	$t = 28$	0,4497	$t = 53$	0,4330	$t = 78$	0,3943
$t = 4$	0,3609	$t = 29$	0,2178	$t = 54$	0,0733	$t = 79$	0,0646
$t = 5$	0,7182	$t = 30$	0,7762	$t = 55$	0,7895	$t = 80$	0,7878
$t = 6$	0,2258	$t = 31$	0,1078	$t = 56$	0,0262	$t = 81$	0,1694
$t = 7$	0,3175	$t = 32$	0,3773	$t = 57$	0,4223	$t = 82$	0,4468
$t = 8$	0,0127	$t = 33$	0,0001	$t = 58$	0,0385	$t = 83$	0,0372
$t = 9$	0,4290	$t = 34$	0,3877	$t = 59$	0,3303	$t = 84$	0,2632

<sup>(1)</sup> Tabelas disponíveis em <http://lajpsseleccao.ufsc.br>

Amostra	$f(t)$	Amostra	$f(t)$	Amostra	$f(t)$	Amostra	$f(t)$
$t = 10$	0,0844	$t = 35$	0,0821	$t = 60$	0,2037	$t = 85$	0,3048
$t = 11$	0,8000	$t = 36$	0,7836	$t = 61$	0,7332	$t = 86$	0,6616
$t = 12$	0,0450	$t = 37$	0,1887	$t = 62$	0,3328	$t = 87$	0,4690
$t = 13$	0,4268	$t = 38$	0,4483	$t = 63$	0,4445	$t = 88$	0,4132
$t = 14$	0,0112	$t = 39$	0,0424	$t = 64$	0,0909	$t = 89$	0,1523
$t = 15$	0,3218	$t = 40$	0,2539	$t = 65$	0,1838	$t = 90$	0,1162
$t = 16$	0,2185	$t = 41$	0,3164	$t = 66$	0,3888	$t = 91$	0,4334
$t = 17$	0,7240	$t = 42$	0,6386	$t = 67$	0,5277	$t = 92$	0,3978
$t = 18$	0,3516	$t = 43$	0,4862	$t = 68$	0,6042	$t = 93$	0,6987
$t = 19$	0,4420	$t = 44$	0,4068	$t = 69$	0,3435	$t = 94$	0,2538
$t = 20$	0,0984	$t = 45$	0,1811	$t = 70$	0,2304	$t = 95$	0,2998
$t = 21$	0,1747	$t = 46$	0,1101	$t = 71$	0,0568	$t = 96$	0,0195
$t = 22$	0,3964	$t = 47$	0,4372	$t = 72$	0,4500	$t = 97$	0,4366
$t = 23$	0,5114	$t = 48$	0,3795	$t = 73$	0,2371	$t = 98$	0,0924
$t = 24$	0,6183	$t = 49$	0,7092	$t = 74$	0,7705	$t = 99$	0,7984
$t = 25$	0,3330	$t = 50$	0,2400	$t = 75$	0,1246	$t = 100$	0,0077

## Apêndice IV

### Conjunto de treinamento referente à seção 6.5<sup>(1)</sup>

Amostra	$x_1$	$x_2$	$x_3$	$d$	Amostra	$x_1$	$x_2$	$x_3$	$d$
1	0.9532	0.6949	0.4451	0.8426	76	0.6441	0.2997	0.5847	0.6545
2	0.7954	0.6346	0.0449	0.6676	77	0.0803	0.3799	0.6020	0.4991
3	0.1427	0.048	0.6267	0.3780	78	0.1908	0.6046	0.5402	0.6665
4	0.1516	0.9824	0.0627	0.4627	79	0.6907	0.3967	0.6055	0.7505
5	0.4868	0.6223	0.7462	0.8116	80	0.2581	0.0582	0.3978	0.3604
6	0.3498	0.5115	0.0783	0.4559	81	0.4241	0.1860	0.9066	0.6298
7	0.8146	0.6378	0.5837	0.8628	82	0.3382	0.9303	0.2475	0.6287
8	0.2820	0.5409	0.7256	0.6939	83	0.3629	0.1502	0.9981	0.5948
9	0.5710	0.2968	0.5477	0.6619	84	0.9259	0.0960	0.1645	0.4716
10	0.9323	0.0229	0.4797	0.5731	85	0.8608	0.6779	0.0033	0.6342
11	0.2907	0.7245	0.5165	0.6911	86	0.0838	0.5472	0.3758	0.4635
12	0.0068	0.0545	0.0861	0.0861	87	0.0303	0.9191	0.7233	0.6491
13	0.2630	0.9885	0.2175	0.5847	88	0.9293	0.6319	0.9654	0.9840
14	0.035	0.3683	0.7801	0.5117	89	0.7268	0.1449	0.9753	0.7098
15	0.667	0.3031	0.7127	0.7806	90	0.2868	0.6593	0.4076	0.6328
16	0.0000	0.7763	0.8735	0.6368	91	0.5515	0.1364	0.2804	0.4745
17	0.4395	0.0601	0.9791	0.5712	92	0.7663	0.0067	0.5546	0.5708
18	0.9359	0.0366	0.9514	0.6826	93	0.6462	0.6761	0.6340	0.8933
19	0.0173	0.9548	0.4289	0.5527	94	0.3694	0.3212	0.1233	0.3658
20	0.6112	0.907	0.6296	0.8803	95	0.2708	0.3222	0.9946	0.6310
21	0.2010	0.9673	0.6791	0.7263	96	0.6282	0.1404	0.8474	0.6733
22	0.6914	0.9144	0.2641	0.7966	97	0.6861	0.6693	0.3810	0.7433

<sup>(1)</sup> Tabelas disponíveis em <http://lajps.sciencedirect.com>

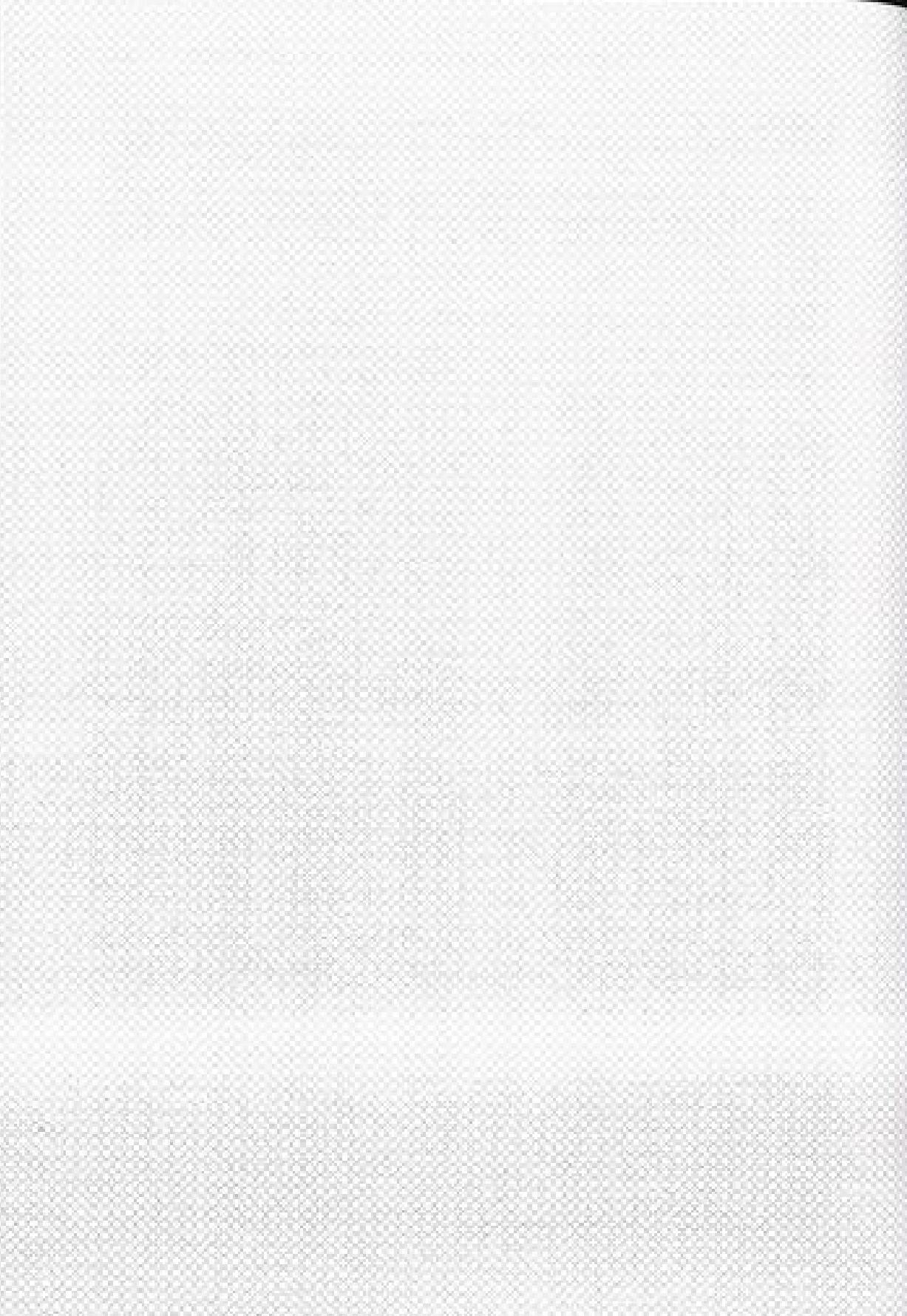
Amostra	$x_0$	$x_1$	$x_2$	$d$	Amostra	$x_0$	$x_1$	$x_2$	$d$
23	0,0061	0,6802	0,8621	0,3711	98	0,6857	0,9901	0,5141	0,8466
24	0,2312	0,4664	0,3821	0,5280	99	0,5915	0,5588	0,3053	0,6787
25	0,2451	0,6964	0,0751	0,4637	100	0,6359	0,4145	0,5016	0,7597
26	0,7681	0,9833	0,3038	0,8049	101	0,5497	0,6319	0,6382	0,8521
27	0,2433	0,0794	0,5851	0,4223	102	0,7072	0,1721	0,3812	0,5772
28	0,2782	0,8414	0,2797	0,6079	103	0,1185	0,5084	0,8376	0,6211
29	0,7616	0,4698	0,5337	0,7809	104	0,6365	0,5562	0,4965	0,7693
30	0,3395	0,0022	0,0087	0,1838	105	0,4145	0,5797	0,6569	0,7878
31	0,7849	0,9981	0,4449	0,8841	106	0,2575	0,5258	0,4038	0,5777
32	0,8012	0,0981	0,2129	0,4857	107	0,2026	0,3800	0,3054	0,4261
33	0,9783	0,1102	0,6227	0,6667	108	0,3885	0,0470	0,5941	0,4625
34	0,6597	0,3264	0,6802	0,7829	109	0,4094	0,1726	0,7803	0,6015
35	0,9295	0,3275	0,7536	0,8018	110	0,1261	0,5151	0,4927	0,5739
36	0,2435	0,2163	0,7625	0,5449	111	0,1234	0,4662	0,2146	0,4007
37	0,9281	0,8356	0,5285	0,8991	112	0,6793	0,6774	1,0000	0,9141
38	0,8313	0,7568	0,6192	0,9047	113	0,8176	0,0558	0,2506	0,4707
39	0,1712	0,0545	0,5033	0,3561	114	0,6937	0,6683	0,5075	0,6220
40	0,0609	0,1702	0,4306	0,3310	115	0,2404	0,5411	0,8754	0,6960
41	0,5899	0,8408	0,0369	0,8245	116	0,6563	0,2609	0,1168	0,4851
42	0,7868	0,5115	0,0916	0,6066	117	0,8886	0,0288	0,2804	0,4802
43	1,0000	0,1653	0,7103	0,7172	118	0,3874	0,5275	0,8457	0,7215
44	0,2007	0,1163	0,3431	0,3385	119	0,2108	0,4910	0,5432	0,5913
45	0,2306	0,033	0,0293	0,1590	120	0,8875	0,6571	0,1849	0,6805
46	0,8477	0,6373	0,4823	0,8254	121	0,5693	0,0242	0,8293	0,6003
47	0,9677	0,7895	0,9467	0,9762	122	0,8439	0,4631	0,6345	0,6236
48	0,6339	0,4659	0,1526	0,3250	123	0,3844	0,2948	0,3937	0,5240
49	0,008	0,8958	0,4201	0,5404	124	0,2014	0,6328	0,9782	0,7143
50	0,9965	0,6897	0,6175	0,8360	125	0,4039	0,0645	0,4629	0,4547
51	0,7408	0,6361	0,2732	0,6949	126	0,7137	0,0670	0,2359	0,4602
52	0,6843	0,3737	0,1562	0,5625	127	0,4277	0,9555	0,0000	0,5477
53	0,8769	0,7998	0,3972	0,8399	128	0,0259	0,7634	0,3689	0,4738
54	0,5700	0,5111	0,2418	0,6258	129	0,1871	0,7662	0,9697	0,7397
55	0,6798	0,4117	0,3370	0,6822	130	0,3216	0,6420	0,0677	0,4526
56	0,3587	0,2987	0,6037	0,5969	131	0,2524	0,7658	0,9523	0,7711
57	0,3866	0,8390	0,0332	0,5318	132	0,3621	0,5295	0,2521	0,5571
58	0,0371	0,7788	0,7445	0,6305	133	0,2942	0,1625	0,2745	0,3759
59	0,8174	0,8422	0,3229	0,8068	134	0,8180	0,0023	0,1439	0,4018
60	0,6027	0,1468	0,3759	0,6342	135	0,8429	0,1704	0,5251	0,6563
61	0,1203	0,3260	0,5419	0,4768	136	0,9612	0,6898	0,6630	0,8128
62	0,1325	0,2082	0,4934	0,4105	137	0,1009	0,4191	0,0828	0,3055
63	0,6650	1,0000	0,4321	0,8404	138	0,7071	0,7704	0,6328	0,9298
64	0,0036	0,1940	0,3274	0,2697	139	0,3371	0,7619	0,0959	0,5377
65	0,2650	0,0161	0,5947	0,4125	140	0,9931	0,6727	0,3130	0,7829
66	0,5849	0,6019	0,4376	0,7464	141	0,9123	0,0000	0,1106	0,3944
67	0,0108	0,3538	0,1810	0,2800	142	0,2858	0,9688	0,2262	0,5888
68	0,9008	0,7264	0,9184	0,9802	143	0,7931	0,8863	0,9028	0,9728
69	0,0023	0,9659	0,3182	0,4986	144	0,7641	0,0778	0,9013	0,6832
70	0,1366	0,6057	0,6967	0,6459	145	0,1389	0,5881	0,2367	0,4622

Amostra	$x_1$	$x_2$	$x_3$	$d$	Amostra	$x_1$	$x_2$	$x_3$	$d$
71	0,8621	0,7353	0,2742	0,7718	146	0,8345	0,5165	0,7139	0,8191
72	0,8682	0,9624	0,4211	0,5764	147	0,2453	0,5888	0,1559	0,4765
73	0,6112	0,6014	0,5254	0,7668	148	0,1174	0,5438	0,3857	0,4953
74	0,0030	0,7585	0,8928	0,6348	149	0,3867	0,3228	0,6952	0,6376
75	0,7644	0,8984	0,0407	0,6055	150	0,2204	0,1785	0,4807	0,4276

### Conjunto de treinamento referente à seção 6.6<sup>(\*)</sup>

Amostra	$x_1$	$x_2$	$d$	Amostra	$x_1$	$x_2$	$d$
1	0,2663	0,0503	-1	21	0,456	0,1871	1
2	0,2405	0,9018	-1	22	0,1715	0,7713	1
3	0,1157	0,3676	1	23	0,5571	0,5485	-1
4	0,5147	0,0167	1	24	0,3344	0,0259	1
5	0,4127	0,3275	1	25	0,4803	0,7635	-1
6	0,2809	0,583	1	26	0,9721	0,485	-1
7	0,8263	0,9301	-1	27	0,8318	0,7844	-1
8	0,9359	0,6724	-1	28	0,1373	0,0292	1
9	0,1096	0,9165	-1	29	0,366	0,8581	-1
10	0,5158	0,8545	-1	30	0,3626	0,7302	-1
11	0,1334	0,1362	1	31	0,6474	0,3324	1
12	0,6371	0,1439	1	32	0,3461	0,2398	1
13	0,7052	0,6277	-1	33	0,1353	0,812	1
14	0,8703	0,8666	-1	34	0,3463	0,1017	1
15	0,2612	0,6109	1	35	0,9056	0,1947	-1
16	0,0244	0,5279	1	36	0,5227	0,2321	1
17	0,9588	0,3672	-1	37	0,5153	0,2041	1
18	0,9332	0,5499	-1	38	0,1832	0,0661	1
19	0,9623	0,2961	-1	39	0,6015	0,9812	-1
20	0,7297	0,5776	-1	40	0,5024	0,5274	-1

<sup>(\*)</sup> Tabelas disponíveis em <http://lapisd.oece.ufsc.br>



## Apêndice V

Conjunto de treinamento referente à seção 8.5<sup>(1)</sup>

Amostra	$x_1$	$x_2$	$x_3$	Amostra	$x_1$	$x_2$	$x_3$
1	0,3417	0,2857	0,2397	61	0,4856	0,6600	0,4798
2	0,2288	0,2874	0,2153	62	0,4114	0,7230	0,5108
3	0,1975	0,3015	0,1965	63	0,5871	0,7935	0,5829
4	0,3414	0,3166	0,1074	64	0,4875	0,7928	0,5532
5	0,2587	0,1918	0,2634	65	0,5172	0,7147	0,5774
6	0,2455	0,2075	0,1344	66	0,5483	0,6773	0,4942
7	0,3183	0,1679	0,1725	67	0,5740	0,6682	0,5335
8	0,2704	0,2605	0,1411	68	0,4587	0,6981	0,5900
9	0,1871	0,2965	0,1231	69	0,5794	0,7410	0,4759
10	0,3474	0,2715	0,1958	70	0,4712	0,6734	0,5677
11	0,2059	0,2828	0,2639	71	0,5128	0,8141	0,5224
12	0,2442	0,2272	0,2384	72	0,5537	0,7749	0,4342
13	0,2126	0,3437	0,1128	73	0,4916	0,8267	0,4586
14	0,2562	0,2542	0,1599	74	0,4623	0,8129	0,4950
15	0,1640	0,2269	0,2627	75	0,5850	0,7358	0,5107
16	0,2795	0,1889	0,1627	76	0,4433	0,7030	0,4594
17	0,3463	0,1513	0,2281	77	0,4155	0,7516	0,5524
18	0,3430	0,1908	0,1881	78	0,4887	0,7027	0,5888
19	0,1981	0,2821	0,1294	79	0,5462	0,7378	0,5107
20	0,2392	0,3025	0,2191	80	0,5251	0,8124	0,5686

<sup>(1)</sup> Tabelas disponíveis em <http://lapuscle.ces.usp.br>

Amostra	$x_1$	$x_2$	$x_3$	Amostra	$x_1$	$x_2$	$x_3$
21	0,7352	0,2722	0,6962	81	0,4635	0,7339	0,5638
22	0,7191	0,1825	0,7470	82	0,5907	0,7144	0,4718
23	0,6921	0,1537	0,8172	83	0,4962	0,8335	0,4597
24	0,6833	0,2048	0,8490	84	0,5242	0,7325	0,4079
25	0,8012	0,2684	0,7673	85	0,4075	0,6372	0,4271
26	0,7860	0,1734	0,7198	86	0,5934	0,8284	0,5107
27	0,7205	0,1542	0,7295	87	0,5463	0,6766	0,5639
28	0,6549	0,3286	0,8153	88	0,4403	0,8495	0,4806
29	0,6968	0,3173	0,7389	89	0,4531	0,7760	0,5276
30	0,7448	0,2095	0,6847	90	0,5109	0,7387	0,5373
31	0,6748	0,3277	0,6725	91	0,4383	0,7780	0,4965
32	0,7897	0,3801	0,7879	92	0,5879	0,7156	0,5022
33	0,6369	0,3067	0,7063	93	0,5762	0,7761	0,5908
34	0,6064	0,3206	0,7205	94	0,5997	0,7504	0,5678
35	0,8357	0,3220	0,7879	95	0,4138	0,6975	0,5148
36	0,7438	0,3230	0,8384	96	0,5490	0,6674	0,4472
37	0,8172	0,3319	0,7628	97	0,4719	0,7527	0,4491
38	0,6248	0,2614	0,6405	98	0,4458	0,8063	0,4253
39	0,6979	0,2142	0,7309	99	0,4983	0,6131	0,5625
40	0,6804	0,3181	0,7917	100	0,5742	0,6789	0,5997
41	0,6973	0,3194	0,7522	101	0,5269	0,7354	0,4718
42	0,7910	0,2239	0,7018	102	0,5927	0,7738	0,5390
43	0,7052	0,2148	0,6868	103	0,5199	0,7131	0,4028
44	0,8088	0,1908	0,7563	104	0,5716	0,6550	0,4451
45	0,7640	0,1676	0,6994	105	0,5075	0,7045	0,4233
46	0,7616	0,2681	0,8287	106	0,4886	0,7004	0,4608
47	0,8188	0,2461	0,7273	107	0,5527	0,8243	0,5772
48	0,7920	0,3178	0,7497	108	0,4816	0,6969	0,4676
49	0,7902	0,1871	0,8102	109	0,5809	0,6557	0,4266
50	0,7333	0,2543	0,8194	110	0,5851	0,7585	0,4003
51	0,6921	0,1529	0,7759	111	0,5334	0,8446	0,4934
52	0,6833	0,2197	0,6943	112	0,4603	0,7992	0,4816
53	0,7660	0,1745	0,7639	113	0,5491	0,6504	0,4063
54	0,8009	0,3062	0,8491	114	0,4288	0,8455	0,5047
55	0,7793	0,1935	0,6738	115	0,5636	0,7884	0,5417
56	0,7373	0,2698	0,7864	116	0,5349	0,6736	0,4541
57	0,7648	0,2380	0,7825	117	0,5549	0,8293	0,5652
58	0,8383	0,2657	0,7733	118	0,4729	0,7702	0,5325
59	0,6878	0,2126	0,6961	119	0,5472	0,8454	0,5449
60	0,6651	0,3462	0,6737	120	0,5805	0,7349	0,4464

## Índice Remissivo

<i>Adaptive Linear Element (ADELINE)</i>	26, 73	Classificação de padrões	28
<i>Adaptive Resonance Theory (ART)</i>	263	Clusterização	28, 179, 222, 246, 265, 280
Agentes inteligentes	24	Computação evolutiva	24
Agrupamento de dados	28	Critério de máxima verossimilhança	180
Algoritmo de aprendizagem	51	Diagrama de Voronoi	246
Algoritmo LAM	76	Dilema plásticidade/castabilidade	264
Aprendizagem competitiva	222, 255	Early stopping	155
Aprendizagem off-line	53	Época de treinamento	51, 66, 81, 110
Aprendizagem se-dar	54	Erro quadrático	77, 99
Aproximador universal	27	Erro quadrático médio	99
Arquitetura em estrutura reticulada	50	Estratégia <i>follow-the-leader</i>	280
Arquitetura <i>feedforward</i>	46, 47	Estratégia <i>winner-take-all</i>	225
Arquitetura recorrente	49	<i>Feedforward time-delayed feedforward network</i>	140
Avatares	206	Fronteira de decisão ótima	85
Backpropagation	27, 92, 95, 102, 111	Fronteira de separabilidade	62
Camadas neurais	46	Fronteiras de Bayes	253
camada de entrada	46	Função de energia	200, 204
camada de saída	46	Função de Lyapunov	204
camada escondida	46	Funções de ativação	33, 36
Características principais	24	degrau	36
adaptação por experiência	24	degrau bipolar (simil)	36
armazenamento distribuído	25	gradiente	41
capacidade de aprendizado	25	linear	42
fácilidade de prototipagem	25	logística	38
habilidade de generalização	25	rampa simétrica	37
organização de dados	25	tangente hiperbólica	39
tolerância a falhas	25	Funções de base radial. Consulte: Rede RBF	

Gradiente descendente	76, 157	Rede ADALINE	26, 47, 73
Inteligência coletiva	24	emergência	79
Inteligência computacional	23	fase de operação	82
<i>Learning vector quantization (LVQ)</i>	243	fase de treinamento	81
Límite de ativação	31, 34	funcionamento	74
ML-ADALINE	26, 74	processo de treinamento	76
Mapas auto-organizáveis	229	Rede ART	263
Mapas topológicos	229	aspectos de aprendizado	269
Mapamento de problema inverso	137	camada de comparação	266
Memórias associativas	28, 207	camada de reconhecimento	267
Método da pseudo-inversa	210	estrutura topológica	265
Método de Fletcher-Gill	151	grupos topológicos	264
Método de Kolmogorov	151	intensificação de contraste	263
Método de Levenberg-Marquardt	116	memória de curto prazo	263
Método do produto externo	208	normalização de sinapses	263
Método $\Delta$ -mar	177, 179	parâmetro de vigilância	270
Método <i>out of n classes</i>	130	parâmetros de ganho	281
Método <i>nearest neighbor</i>	114	ressonância adaptativa	268
Mínimos quadrados	84, 117,	Rede <i>nearest neighbor</i>	254
Mínimos quadrados recursivo	180	atriz de Kohonen	255
Modelo de Hodgkin & Huxley	30, 33	camada Oculta	256
Mosaico de Dirichlet	246	fase de operação	259
<i>Multilayer Proprietary</i> . Consulte: Rede Proprietary multicamadas		fase de treinamento	258
Neurônio artificial	25, 33	Rede de Hopfield	199
combinador linear	34	atratores	206
função de ativação	35	condições de estabilidade	204
limiar de ativação	34	estados espáticos	213
pesos sinápticos	34	função de energia	200, 204
potencial de ativação	34	funcionamento	201
sinais de entrada	34	hardware	216
sinal de saída	35	<i>image model</i>	200
Neurônio biológico	29	memórias associativas	207
Neurônios efetuadores	30	pontos de equilíbrio	200
Neurônios sensitivos	29	Rede de Kohonen	221
Normalização de dados	160	aprendizagem competitiva	222
Overfitting	153	fase de operação	236
Pesada antecipada	155	fase de treinamento	234
Ponto de equilíbrio	200	mapas auto-organizáveis	229
Parâmetro de vigilância	270	mapas topológicos	229
Potencial de ação	31, 32	Rede LVQ	243
Potencial de ativação	32, 34	quantização vetorial	244
Potencial de repouso	32	treinamento LVQ-1	248
Princípio da ressonância adaptativa	268	treinamento LVQ-2	252
Problema do ou-exclusivo	26, 122, 124	treinamento LVQ-2.1	253
Processo de treinamento	50	Rede Proprietary	26, 57
Quantização vetorial	244	discriminador linear	61
<i>Radial basis function</i> . Consulte: Rede RBF		fase de operação	66
		fase de treinamento	65

- funcionamento ..... 59  
 processo de treinamento ..... 63  
 teorema de convergência ..... 63  
**Rede Perceptron multicamadas** ..... 91  
 algoritmo *backpropagation* ..... 95  
 aproximação funcional ..... 91, 132  
 classificação de padrões ..... 121  
 convergência ..... 111, 115, 115, 157  
 fase de operação ..... 111  
 fase de treinamento ..... 110  
 funcionamento ..... 92  
 implementação ..... 158  
 módulos locais ..... 157  
*overfitting* ..... 153  
 pós-processamento ..... 161  
 sistemas variantes no tempo ..... 137  
 termo de momentum ..... 112  
*underfitting* ..... 153  
 validação cruzada ..... 147  
**Rede RBF** ..... 173  
 aproximação de funções ..... 183  
 campo receptivo ..... 176  
 classificação de padrões ..... 186  
 fase de operação ..... 183  
 fronteira de separabilidade ..... 177  
 método *k-means* ..... 177, 179  
 princípio estágio de treinamento ..... 178  
 processo de treinamento ..... 174  
 segundo estágio de treinamento ..... 182  
**Rede recorrente** ..... 49, 143, 199, 303  
**Regra de Hebb** ..... 47, 60, 84, 255  
**Regra delta** ..... 47, 84  
**Regra delta generalizada** ..... 48, 73, 94, 174  
**Rosenblatt** ..... 26, 57  
*Self-organizing map* ..... 229, 247  
*Simulado annealing* ..... 158  
**Sistemas fuzzy** ..... 24  
**Sistemas inteligentes** ..... 24  
**SOM**. Consulte: *Self-organizing map*  
**Subconjunto de teste** ..... 151  
**Subconjunto de treinamento** ..... 151  
*Support vector machines (SVM)* ..... 27  
 Taxa de aprendizagem ..... 64, 79, 115, 224, 249  
 Teorema da aproximação universal ..... 133, 136  
 Teorema de convergência do Perceptron ..... 63  
 Teorema de Kolmogorov ..... 153, 183  
 Teoria da ressonância adaptativa ..... 263  
 Termo de momentum ..... 112  
*Time delay neural network* ..... 139  
 Topologia de redes neurais ..... 45  
 Treinamento com reforço ..... 53  
 Treinamento não-supervisionado ..... 52, 264  
**Treinamento**  
 supervisão ..... 51, 60, 93, 243, 264  
*Underfitting* ..... 153  
 Validação cruzada ..... 147  
 amostragem aleatória ..... 147  
 k-partições ..... 148  
 parada antecipada ..... 155  
 por unidade ..... 149  
 Vetor quantizadores ..... 244