

Redes Neurais Artificiais

PARA ENGENHARIA E CIÊNCIAS APLICADAS

fundamentos teóricos e aspectos práticos



Ivan Nunes da Silva
Danilo Hernane Spatti
Rogério Andrade Flauzino

Artliber
EDITORIAL

Ivan Nunes da Silva
Danilo Hernane Spatti
Rogério Andrade Flauzino

Universidade de São Paulo

Redes Neurais Artificiais

para engenharia e ciências aplicadas

2ª edição
revisada e ampliada

Artliber
EDITORA

Copyright© 2019 by Artliber Editora Ltda.
Copyright© 2016 by Artliber Editora Ltda.
Copyright© 2010 by Artliber Editora Ltda.

2ª edição - 2016

Revisão:

Maria Antonieta M. Eckersdorff

Capa e composição eletrônica:

Perfil 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; Danilo Hernane Spatti; Rogério Andrade Flauzino. – São Paulo: Artliber, 2010.

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

10-0725

CDD-004.032.62

Índices para catálogo sistemático:

1. Redes neurais: Engenharia e ciências aplicadas 004.032.62

2019

Todos os direitos desta edição são reservados à

Artliber Editora Ltda.

Av. Diógenes Ribeiro de Lima, 3294

05083-010 – São Paulo – SP – Brasil

Tel.: (11) 3641-3893

info@artliber.com.br

www.artliber.com.br



Sumário

Prefácio.....	13
Organização	15
Agradecimentos.....	17
Parte I – Arquiteturas de redes neurais artificiais e seus aspectos teóricos.....	19
Capítulo 1 – Introdução.....	21
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
Capítulo 2 – Arquiteturas de redes neurais artificiais e processos de treinamento.....	45
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	Processos 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 lote 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 – Rede <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>Perceptron</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 (<i>RBF</i>).....		173
6.1	Introdução	173
6.2	Processo de treinamento de redes <i>RBF</i>	174
6.2.1	Ajuste dos neurônios da camada intermediária (estágio I)	174
6.2.2	Ajuste dos neurônios da camada de saída (estágio II).....	181
6.3	Aplicabilidades das redes <i>RBF</i>	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 produto 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 (<i>SOM</i>)	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>LVQ-1</i>	248
9.3.2	Algoritmo de treinamento <i>LVQ-2</i>	252
9.4	Redes <i>counter-propagation</i>	254
9.4.1	Aspectos da camada <i>outstar</i>	256
9.4.2	Algoritmo de treinamento da rede <i>counter-propagation</i>	257
9.5	Exercícios.....	259
9.6	Projeto prático	260
Capítulo 10	– Redes <i>ART</i> (adaptive resonance theory).....	263
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
Parte II	– Aplicações de redes neurais artificiais em problemas de engenharia e ciências aplicadas	287
Capítulo 11	– Estimação da qualidade global de café utilizando o <i>Perceptron</i> multicamadas	289
11.1	Introdução	289
11.2	Características da Rede PMC.....	290
11.3	Resultados computacionais.....	292
Capítulo 12	– Análise do tráfego de redes de computadores utilizando protocolo <i>SNMP</i> e rede <i>LVQ</i>	295
12.1	Introdução	295
12.2	Características da rede <i>LVQ</i>	297
12.3	Resultados computacionais.....	299
Capítulo 13	– Previsão de tendências do mercado de ações utilizando redes recorrentes.....	301
13.1	Introdução	301
13.2	Características da rede recorrente	303
13.3	Resultados computacionais.....	304
Capítulo 14	– Sistema para diagnóstico de doenças utilizando redes <i>ART</i>	309
14.1	Introdução	309

14.2	Características da Rede <i>ART</i>	311
14.3	Resultados computacionais	312
Capítulo 15 – Identificação de padrões de adulterantes em pó de café		
	usando mapas de Kohonen	315
15.1	Introdução	315
15.2	Características da rede de Kohonen	316
15.3	Resultados computacionais	319
Capítulo 16 – Reconhecimento de distúrbios relacionados à qualidade		
	da energia elétrica utilizando redes PMC	321
16.1	Introdução	321
16.2	Características da rede PMC	325
16.3	Resultados computacionais	326
Capítulo 17 – Controle de trajetória de robôs móveis usando sistemas		
	<i>fuzzy</i> e redes PMC	329
17.1	Introdução	329
17.2	Características da rede PMC	331
17.3	Resultados computacionais	334
Capítulo 18 – Método para classificação de tomates usando visão computacional		
	e redes PMC	339
18.1	Introdução	339
18.2	Características da rede neural	341
18.3	Resultados computacionais	345
Capítulo 19 – Análise de desempenho de redes <i>RBF</i> e PMC em		
	classificação de padrões	347
19.1	Introdução	347
19.2	Características das redes <i>RBF</i> e PMC	348
19.3	Resultados computacionais	349
Capítulo 20 – Resolução de problemas de otimização com restrições		
	por redes de Hopfield	355
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
Capítulo 21 – Classificação de carne bovina utilizando ressonância magnética nuclear e		
	redes neurais	371

21.1	Introdução	371
21.2	Características da rede PMC	375
21.3	Resultados computacionais	377
Capítulo 22 – Sistema inteligente para sensoriamento virtual de oxigênio em veículos de injeção eletrônica		
22.1	Introdução	379
22.2	Características da rede PMC	383
22.3	Resultados computacionais	384
Capítulo 23 – Estimação de desempenho em sistemas elétricos usando redes PMC		
23.1	Introdução	387
23.2	Características da rede PMC	388
23.3	Resultados computacionais	391
Capítulo 24 – Identificação de transitórios eletromagnéticos em redes elétricas		
24.1	Introdução	395
24.2	Características da rede PMC	397
24.3	Resultados computacionais	399
Capítulo 25 – Projeto otimizado de brake-lights automotivos usando redes PMC		
25.1	Introdução	401
25.2	Características da rede PMC	405
25.3	Resultados computacionais	406
Bibliografia		409
Apêndice I		417
Apêndice II		418
Apêndice III		419
Apêndice IV		424
Apêndice V		427
Índice remissivo		429

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.

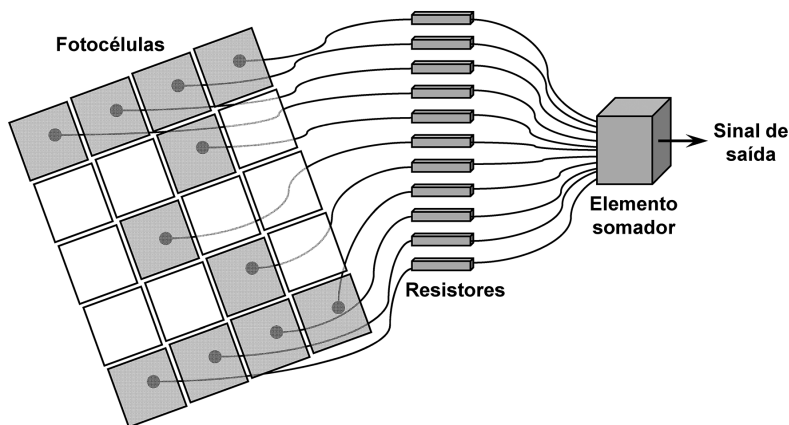


Figura 3.1 – Modelo ilustrativo do *Perceptron* para reconhecimento de padrões

A simplicidade da rede *Perceptron* 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 *Perceptron* constituída de n sinais de entrada, representativos do problema a ser mapeado, e somente uma saída, pois este tipo de rede é composto de um único neurônio.

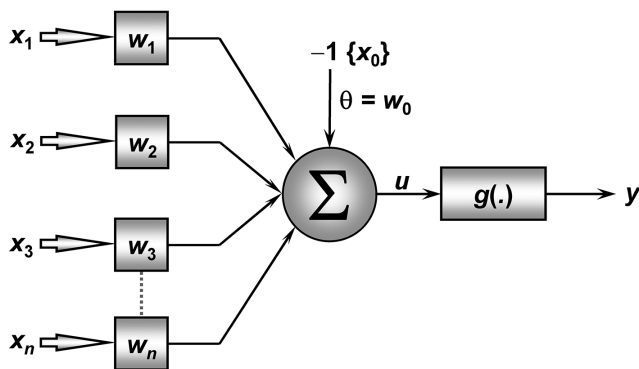


Figura 3.2 – Ilustração da rede *Perceptron*

Embora seja uma rede simples, o *Perceptron* 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 reside 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 $\{w_i\}$ 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 descrito pelas seguintes expressões:

$$\left\{ \begin{array}{l} u = \sum_{i=1}^n w_i \cdot x_i - \theta \\ y = g(u) \end{array} \right. \quad (3.1)$$

$$(3.2)$$

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

Tipicamente, devido às suas características estruturais, as funções de ativação normalmente usadas no *Perceptron* 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 *Perceptron*. 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 *Perceptron*.

Tabela 3.1 – Aspectos dos parâmetros característicos do *Perceptron*

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	θ	Real (iniciado aleatoriamente)
Saída	y	Binária
Função de ativação	$g(.)$	Degrau ou degrau bipolar
Processo de treinamento	-----	Supervisionado
Regra de aprendizado	-----	Regra de Hebb

O ajuste dos pesos e limiar do *Perceptron* 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 *Perceptron* 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 *Perceptron* 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.

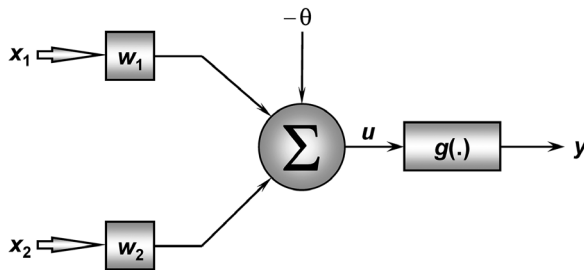


Figura 3.3 – *Perceptron* constituído de duas entradas

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

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

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:

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

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.

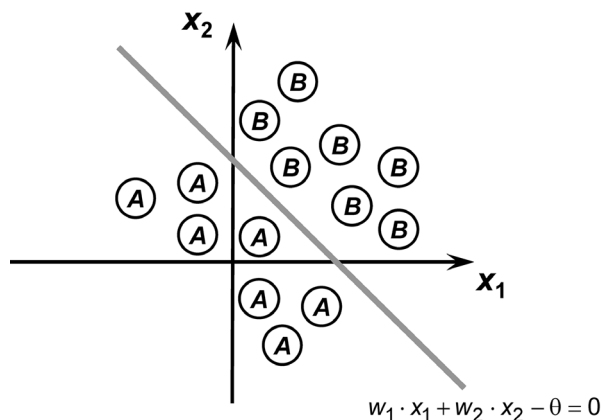


Figura 3.4 – Ilustração de fronteira de separação (neurônio com duas entradas)

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.

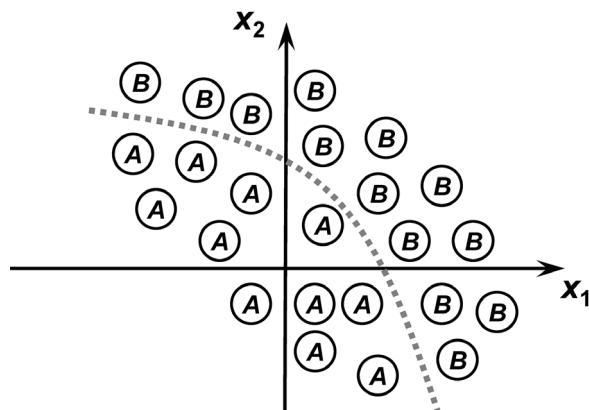


Figura 3.5 – Ilustração de fronteira não linearmente separável

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 não coincidente com a saída desejada, os pesos sinápticos e limiares da rede serão então incrementados (condição excitatória) proporcionalmente aos valores de seus sinais de entrada; caso contrário, ou seja, a saída produzida pela rede é igual ao valor desejado, os pesos sinápticos e limiar permanecerão então inalterados (condição inibitória). 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 $\{\theta\}$ do neurônio podem ser expressas, respectivamente, pelas seguintes expressões:

$$\begin{cases} w_i^{atual} = w_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot x_i^{(k)} \\ \theta_i^{atual} = \theta_i^{anterior} + \eta \cdot (d^{(k)} - y) \cdot (-1) \end{cases} \quad (3.5)$$

$$\quad \quad \quad (3.6)$$

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 $\{\theta\}$ 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:

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

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

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

onde: $\mathbf{w} = [\theta \ w_1 \ w_2 \ \dots \ w_n]^T$ 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;

y é valor da saída produzida pelo *Perceptron*;

η é uma constante que define a taxa de aprendizagem da rede.

A taxa de aprendizagem $\{\eta\}$ exprime o quão rápido o processo de treinamento da rede estará sendo conduzido rumo à sua convergência (estabilização). A escolha de η 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 $d^{(k)}$, supõe-se que um problema a ser mapeado pelo *Perceptron* 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: $\Omega^{(x)} = \{ [0,1 \ 0,4 \ 0,7]; [0,3 \ 0,7 \ 0,2]; [0,6 \ 0,9 \ 0,8]; [0,5 \ 0,7 \ 0,1] \}$. Considerando-se ainda que os respectivos valores de saída para cada uma dessas amostras seja dado por $\Omega^{(d)} = \{ [1]; [-1]; [-1]; [1] \}$, então se pode adotar a seguinte forma matricial para suas representações:

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

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

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

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

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

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

Entretanto, em se tratando de implementações computacionais, a disposição dos dados por meio de variáveis indexadas se torna bem mais apropriada 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 {  $\mathbf{x}^{(k)}$  };
<2> Associar a saída desejada {  $d^{(k)}$  } para cada amostra obtida;
<3> Iniciar o vetor  $\mathbf{w}$  com valores aleatórios pequenos;
<4> Especificar a taxa de aprendizagem {  $\eta$  };
<5> Iniciar o contador de número de épocas {  $\acute{e}poca \leftarrow 0$  };
<6> Repetir as instruções:
    {
        <6.1> erro  $\leftarrow$  "inexiste";
        <6.2> Para todas as amostras de treinamento {  $\mathbf{x}^{(k)}, d^{(k)}$  }, fazer:
            {
                <6.2.1>  $u \leftarrow \mathbf{w}^T \cdot \mathbf{x}^{(k)}$ ;
                <6.2.2>  $y \leftarrow \text{sinal}(u)$ ;
                <6.2.3> Se  $y \neq d^{(k)}$ 
                    {
                        <6.2.3.1> Então {  $\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot (d^{(k)} - y) \cdot \mathbf{x}^{(k)}$  }
                        erro  $\leftarrow$  "existe"
                    }
                <6.3>  $\acute{e}poca \leftarrow \acute{e}poca + 1$ ;
            }
        Até que: erro = "inexiste"
    }

```

Fim {Algoritmo *Perceptron* – Fase de Treinamento}

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

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, considera-se uma rede composta de apenas duas entradas $\{x_1 \text{ e } x_2\}$.

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.

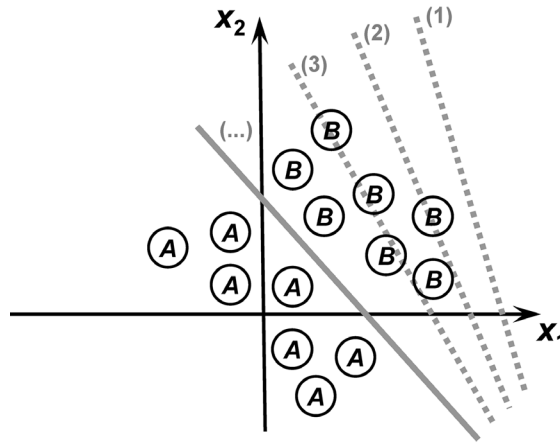


Figura 3.6 – Ilustração do processo de treinamento do *Perceptron*

Analisando 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.

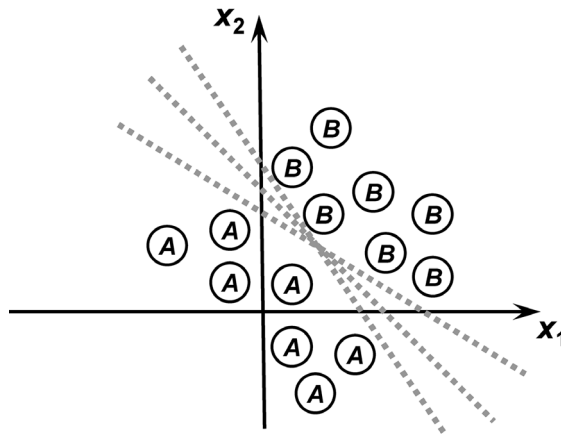


Figura 3.7 – Ilustração de conjunto de retas passíveis de separar as classes

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 $\{\mathbf{w}\}$, 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 $\{\mathbf{w}\}$.

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 $\{x_1$ 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 $\{P_1 \text{ e } P_2\}$ a partir da medição de três grandezas $\{x_1, x_2 \text{ e } x_3\}$, 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 I, tomando 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.

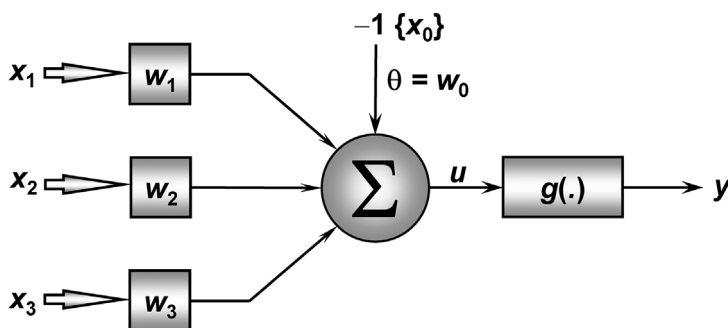


Figura 3.8 – Arquitetura do *Perceptron* para o projeto prático

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 $\{\mathbf{w}\}$ 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 este 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	-1,8842	-0,2805	1,2548					

- 4) Explique por que o número de épocas de treinamento, em relação a esta aplicação, varia a cada vez que executamos o treinamento do *Perceptron*.
- 5) Para a aplicação em questão, discorra se é possível afirmar se as suas classes são linearmente separáveis.



Frank Rosenblatt

Redes Perceptron multicamadas

5.1 – Introdução

As redes *Perceptron* 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 efetivado 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.

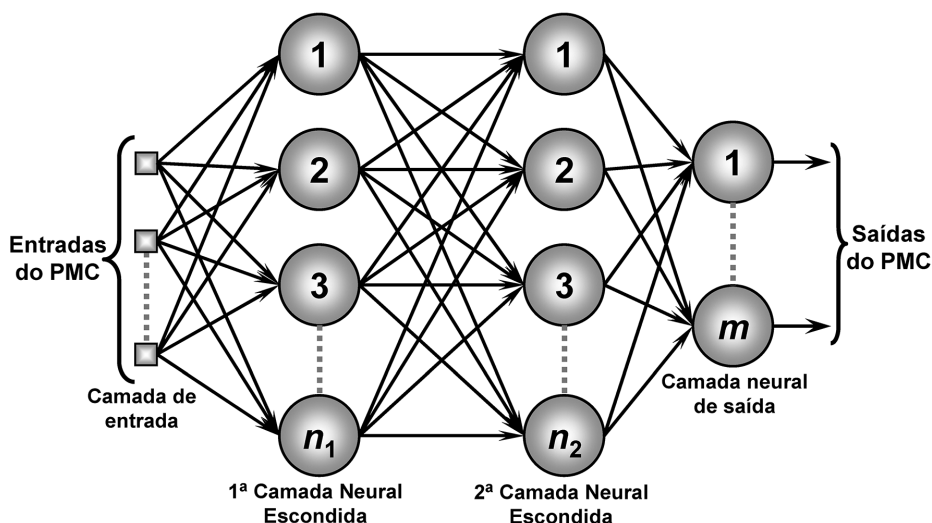


Figura 5.1 – Ilustração de rede *Perceptron* multicamadas

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 alii*, 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 de saída 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 PMC é denominado *backpropagation* ou algoritmo de retropropagação do erro.

5.3 – Processo de treinamento do *Perceptron* multicamadas

O processo de treinamento de redes PMC 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 PMC 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).

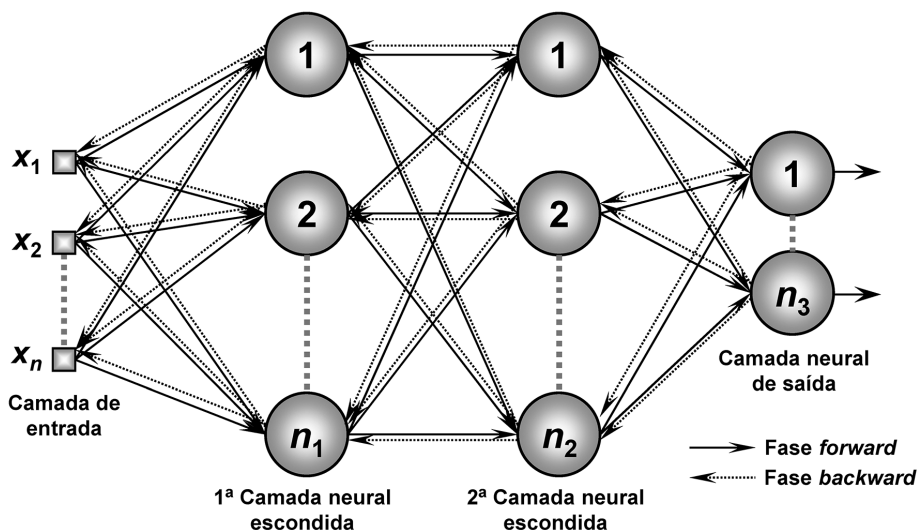


Figura 5.2 – Ilustração das duas fases de treinamento da rede PMC

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 tão 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_3 neurônios em sua camada de saída, os respectivos n_3 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 norteiam a derivação do algoritmo.

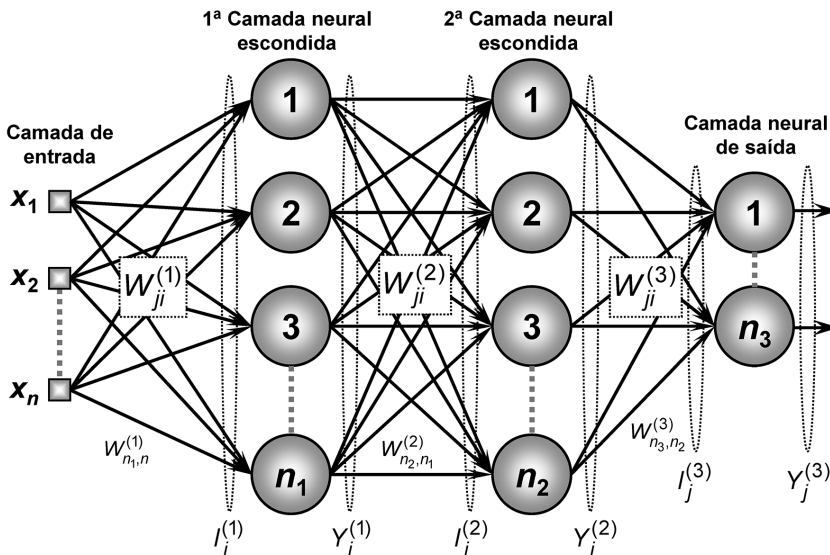


Figura 5.3 – Notação para derivação do algoritmo *backpropagation*

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.

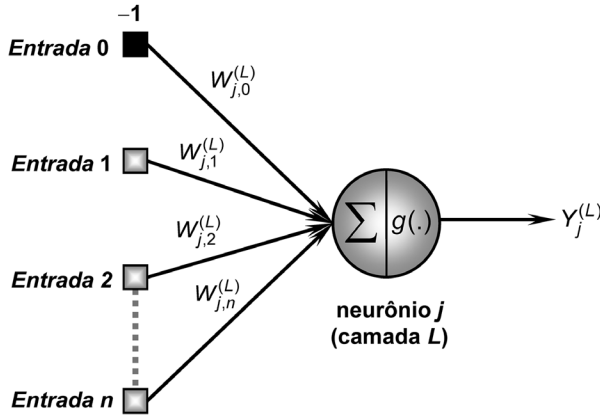


Figura 5.4 – Configuração de neurônio utilizado na derivação do algoritmo *backpropagation*

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

- $W_{ji}^{(L)}$ 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:
 - $W_{ji}^{(3)}$ é o peso sináptico conectando o j -ésimo neurônio da camada de saída ao i -ésimo neurônio da camada 2;
 - $W_{ji}^{(2)}$ é o peso sináptico conectando o j -ésimo neurônio da camada escondida 2 ao i -ésimo neurônio da camada 1;
 - $W_{ji}^{(1)}$ 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:

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

$$I_j^{(2)} = \sum_{i=0}^{n_1} W_{ji}^{(2)} \cdot Y_i^{(1)} \Leftrightarrow I_j^{(2)} = W_{1,0}^{(2)} \cdot Y_0^{(1)} + W_{1,1}^{(2)} \cdot Y_1^{(1)} + \dots + W_{1,n_1}^{(2)} \cdot Y_{n_1}^{(1)} \quad (5.2)$$

$$I_j^{(3)} = \sum_{i=0}^{n_2} W_{ji}^{(3)} \cdot Y_i^{(2)} \Leftrightarrow I_j^{(3)} = W_{1,0}^{(3)} \cdot Y_0^{(2)} + W_{1,1}^{(3)} \cdot Y_1^{(2)} + \dots + W_{1,n_2}^{(3)} \cdot Y_{n_2}^{(2)} \quad (5.3)$$

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

$$Y_j^{(1)} = g(I_j^{(1)}) \quad (5.4)$$

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

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

Como exemplo desta terminologia adotada, considera-se o PMC representado na figura 5.5, composto de duas entradas x_1 e x_2 ($n = 2$), 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.

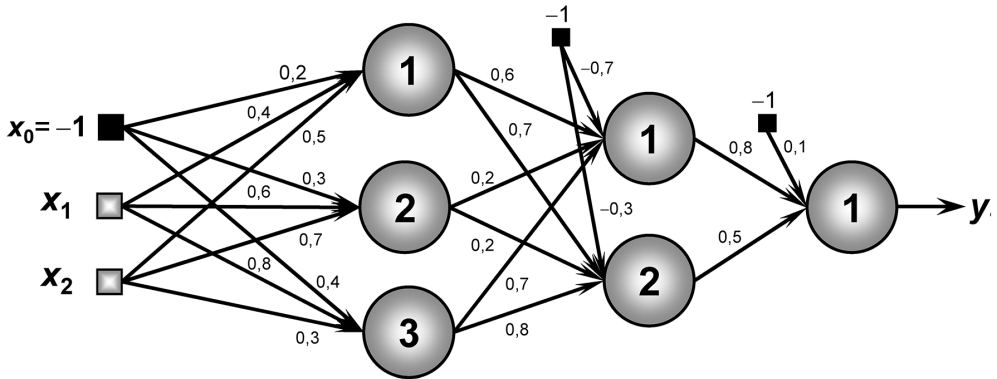


Figura 5.5 – Exemplo de *Perceptron* multicamadas

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

$$W_{ji}^{(1)} = \begin{bmatrix} 0,2 & 0,4 & 0,5 \\ 0,3 & 0,6 & 0,7 \\ 0,4 & 0,8 & 0,3 \end{bmatrix}; \quad W_{ji}^{(2)} = \begin{bmatrix} -0,7 & 0,6 & 0,2 & 0,7 \\ -0,3 & 0,7 & 0,2 & 0,8 \end{bmatrix}; \quad W_{ji}^{(3)} = \begin{bmatrix} 0,1 & 0,8 & 0,5 \end{bmatrix}$$

Assumindo-se um sinal de entrada definido por $x_1 = 0,3$ e $x_2 = 0,7$, os vetores $I_j^{(1)}$ e $Y_j^{(1)}$ seriam então representados por:

$$I_j^{(1)} = \begin{bmatrix} I_1^{(1)} \\ I_2^{(1)} \\ I_3^{(1)} \end{bmatrix} = \begin{bmatrix} W_{1,0}^{(1)} \cdot x_0 + W_{1,1}^{(1)} \cdot x_1 + W_{1,2}^{(1)} \cdot x_2 \\ W_{2,0}^{(1)} \cdot x_0 + W_{2,1}^{(1)} \cdot x_1 + W_{2,2}^{(1)} \cdot x_2 \\ W_{3,0}^{(1)} \cdot x_0 + W_{3,1}^{(1)} \cdot x_1 + W_{3,2}^{(1)} \cdot x_2 \end{bmatrix} = \begin{bmatrix} 0,2 \cdot (-1) + 0,4 \cdot 0,3 + 0,5 \cdot 0,7 \\ 0,3 \cdot (-1) + 0,6 \cdot 0,3 + 0,7 \cdot 0,7 \\ 0,4 \cdot (-1) + 0,8 \cdot 0,3 + 0,3 \cdot 0,7 \end{bmatrix} = \begin{bmatrix} 0,27 \\ 0,37 \\ 0,05 \end{bmatrix}$$

$$Y_j^{(1)} = \begin{bmatrix} Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} g(I_1^{(1)}) \\ g(I_2^{(1)}) \\ g(I_3^{(1)}) \end{bmatrix} = \begin{bmatrix} \tanh(0,27) \\ \tanh(0,37) \\ \tanh(0,05) \end{bmatrix} = \begin{bmatrix} 0,26 \\ 0,35 \\ 0,05 \end{bmatrix} \xrightarrow{Y_0^{(1)} = -1} Y_j^{(1)} = \begin{bmatrix} Y_0^{(1)} \\ Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} -1 \\ 0,26 \\ 0,35 \\ 0,05 \end{bmatrix}$$

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

Os vetores $I_j^{(2)}$ e $Y_j^{(2)}$ referentes à segunda camada neural seriam representados por:

$$I_j^{(2)} = \begin{bmatrix} I_1^{(2)} \\ I_2^{(2)} \end{bmatrix} = \begin{bmatrix} W_{1,0}^{(2)} \cdot Y_0^{(1)} + W_{1,1}^{(2)} \cdot Y_1^{(1)} + W_{1,2}^{(2)} \cdot Y_2^{(1)} + W_{1,3}^{(2)} \cdot Y_3^{(1)} \\ W_{2,0}^{(2)} \cdot Y_0^{(1)} + W_{2,1}^{(2)} \cdot Y_1^{(1)} + W_{2,2}^{(2)} \cdot Y_2^{(1)} + W_{2,3}^{(2)} \cdot Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} 0,96 \\ 0,59 \end{bmatrix}$$

$$Y_j^{(2)} = \begin{bmatrix} Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} g(I_1^{(2)}) \\ g(I_2^{(2)}) \end{bmatrix} = \begin{bmatrix} \tanh(0,96) \\ \tanh(0,59) \end{bmatrix} = \begin{bmatrix} 0,74 \\ 0,53 \end{bmatrix} \xrightarrow{Y_0^{(2)} = -1} Y_j^{(2)} = \begin{bmatrix} Y_0^{(2)} \\ Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} -1 \\ 0,74 \\ 0,53 \end{bmatrix}$$

Finalmente, os vetores $I_j^{(3)}$ e $Y_j^{(3)}$ referentes à terceira camada neural seriam representados por:

$$I_j^{(3)} = [I_1^{(3)}] = [W_{1,0}^{(3)} \cdot Y_0^{(2)} + W_{1,1}^{(3)} \cdot Y_1^{(2)} + W_{1,2}^{(3)} \cdot Y_2^{(2)}] = [0, 76]$$

$$Y_j^{(3)} = [0, 76] = [g(I_1^{(3)})] = [\tanh(0, 76)] = [0, 64]$$

Nesta última expressão, dispensa-se a inserção do termo $Y_0^{(3)} = -1$, pois já se trata da última camada neural, sendo que o valor de $Y_1^{(3)}$ é a própria saída y_1 produzida por esta rede.

O próximo passo para o início 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:

$$E(k) = \frac{1}{2} \sum_{j=1}^{n_3} (d_j(k) - Y_j^{(3)}(k))^2 \quad (5.7)$$

onde $Y_j^{(3)}(k)$ 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 respectivo 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 $W_{ji}^{(3)}$, 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 $W_{ji}^{(2)}$ e $W_{ji}^{(1)}$.

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 de método baseado no gradiente da função erro quadrático dada em (5.7).

Parte I – Ajuste dos pesos sinápticos da camada de saída

O objetivo do processo de treinamento para a camada neural de saída consiste de ajustar a matriz de pesos $W_{ji}^{(3)}$ 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 neurônio 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:

$$\nabla E^{(3)} = \frac{\partial E}{\partial W_{ji}^{(3)}} = \frac{\partial E}{\partial Y_j^{(3)}} \cdot \frac{\partial Y_j^{(3)}}{\partial I_j^{(3)}} \cdot \frac{\partial I_j^{(3)}}{\partial W_{ji}^{(3)}} \quad (5.9)$$

A partir das definições anteriores, tem-se:

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

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

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

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:

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

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

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

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

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

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

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

onde η é a taxa de aprendizagem do algoritmo *backpropagation*. Em notação algébrica, esta expressão é equivalente à seguinte:

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

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 fase *backward* para a topologia ilustrada na figura 5.3, somente após o ajuste dos neurônios da camada neural de saída é que se iniciam as correções dos pesos para aqueles da segunda camada intermediária. Nesta condição específica, indisponem-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 esse erro para os neurônios das camadas anteriores, ponderando-se estes 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 $W_{ji}^{(2)}$ 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:

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

A partir das definições anteriores, tem-se:

$$\frac{\partial l_j^{(2)}}{\partial W_{ji}^{(2)}} = Y_i^{(1)} \quad \{\text{Obtido a partir de (5.2)}\} \quad (5.19)$$

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

$$\frac{\partial E}{\partial Y_j^{(2)}} = \sum_{k=1}^{n_3} \underbrace{\frac{\partial E}{\partial l_k^{(3)}} \cdot \frac{\partial l_k^{(3)}}{\partial Y_j^{(2)}}}_{\text{parcela (i)}} = \sum_{k=1}^{n_3} \underbrace{\frac{\partial E}{\partial l_k^{(3)}}}_{\text{parcela (i)}} \cdot \underbrace{\frac{\partial (\sum_{k=1}^{n_3} W_{kj}^{(3)} \cdot Y_j^{(2)})}{\partial Y_j^{(2)}}}_{\text{parcela (ii)}} \quad (5.21)$$

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

$$\frac{\partial E}{\partial Y_j^{(2)}} = \sum_{k=1}^{n_3} \underbrace{\frac{\partial E}{\partial l_k^{(3)}}}_{\text{parcela (i)}} \cdot \underbrace{W_{kj}^{(3)}}_{\text{parcela (ii)}} \quad (5.22)$$

Torna-se importante observar que o valor da parcela (ii) 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 $W_{ji}^{(3)}$ 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.

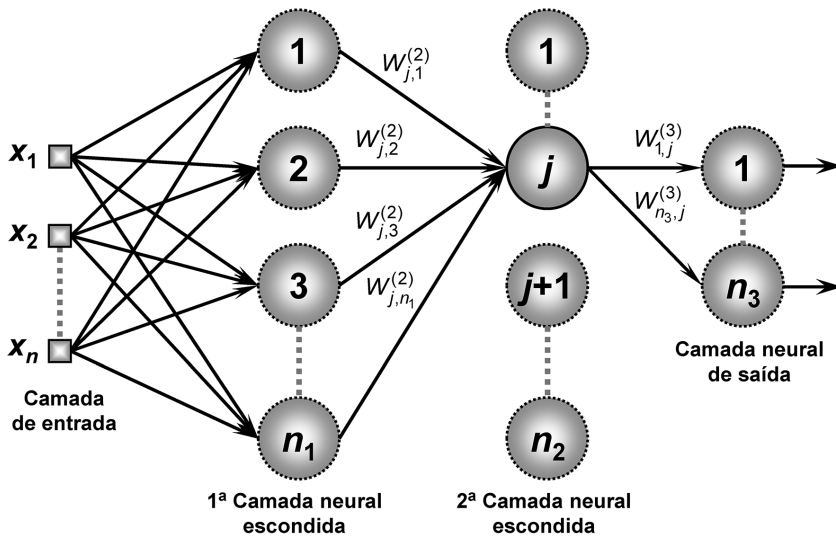


Figura 5.6 – Representação de interligação neural visando correção dos pesos sinápticos de neurônios da segunda camada escondida

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 de $\delta_j^{(3)}$ explicitado em (5.15). Assim, levando-se em consideração tais substituições, a expressão (5.22) pode então ser representada por:

$$\frac{\partial E}{\partial Y_j^{(2)}} = - \sum_{k=1}^{n_3} \delta_k^{(3)} \cdot W_{kj}^{(3)} \quad (5.23)$$

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

$$\frac{\partial E}{\partial W_{ji}^{(2)}} = - \left(\sum_{k=1}^{n_3} \delta_k^{(3)} \cdot W_{kj}^{(3)} \right) \cdot g'(I_j^{(2)}) \cdot Y_i^{(1)} \quad (5.24)$$

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

$$\Delta W_{ji}^{(2)} = -\eta \cdot \frac{\partial E}{\partial W_{ji}^{(2)}} \Leftrightarrow \Delta W_{ji}^{(2)} = \eta \cdot \delta_j^{(2)} \cdot Y_i^{(1)} \quad (5.25)$$

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

$$\delta_j^{(2)} = -\left(\sum_{k=1}^{n_3} \delta_k^{(3)} \cdot W_{kj}^{(3)}\right) \cdot g'(I_j^{(2)}) \quad (5.26)$$

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

$$W_{ji}^{(2)}(t+1) = W_{ji}^{(2)}(t) + \eta \cdot \delta_j^{(2)} \cdot Y_i^{(1)} \quad (5.27)$$

Em notação algorítmica, esta expressão é equivalente à seguinte:

$$W_{ji}^{(2)} \leftarrow W_{ji}^{(2)} + \eta \cdot \delta_j^{(2)} \cdot Y_i^{(1)} \quad (5.28)$$

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 $W_{ji}^{(1)}$ 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:

$$\nabla E^{(1)} = \frac{\partial E}{\partial W_{ji}^{(1)}} = \frac{\partial E}{\partial Y_j^{(1)}} \cdot \frac{\partial Y_j^{(1)}}{\partial I_j^{(1)}} \cdot \frac{\partial I_j^{(1)}}{\partial W_{ji}^{(1)}} \quad (5.29)$$

A partir das definições anteriores, tem-se:

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

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

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

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:

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

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 $W_{ji}^{(2)}$ já foram ajustados no passo anterior com base em valores retropropagados de erro que tiveram como base o ajuste dos pesos $W_{ji}^{(3)}$, 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.

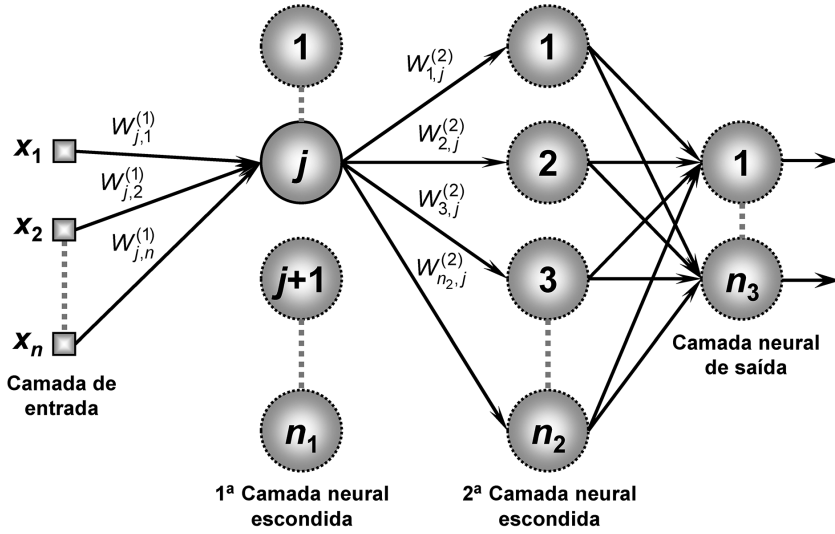


Figura 5.7 – Representação de interligação neural visando correção dos pesos sinápticos de neurônios da primeira camada escondida

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 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:

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

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:

$$\delta_j^{(1)} = -\left(\sum_{k=1}^{n_2} \delta_k^{(2)} \cdot W_{kj}^{(2)}\right) \cdot g'(I_j^{(1)}) \quad (5.37)$$

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 $\mathbf{x}^{(k)}$, representando a k -ésima amostra de treinamento, assim como para os respectivos valores desejados, que são agora inseridos no vetor $\mathbf{d}^{(k)}$, 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 $\{d_1, d_2\}$ 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:

$$\Omega^{(x)} = \begin{matrix} & \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \mathbf{x}^{(3)} & \mathbf{x}^{(4)} \\ \begin{matrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{matrix} & \begin{bmatrix} -1 \\ 0,2 \\ 0,9 \\ 0,4 \end{bmatrix} & \begin{bmatrix} -1 \\ 0,1 \\ 0,3 \\ 0,5 \end{bmatrix} & \begin{bmatrix} -1 \\ 0,9 \\ 0,7 \\ 0,8 \end{bmatrix} & \begin{bmatrix} -1 \\ 0,6 \\ 0,4 \\ 0,3 \end{bmatrix} \end{matrix}; \quad \Omega^{(d)} = \begin{matrix} & \mathbf{d}^{(1)} & \mathbf{d}^{(2)} & \mathbf{d}^{(3)} & \mathbf{d}^{(4)} \\ \begin{matrix} d_1 \\ d_2 \end{matrix} & \begin{bmatrix} 0,7 \\ 0,3 \end{bmatrix} & \begin{bmatrix} 0,6 \\ 0,4 \end{bmatrix} & \begin{bmatrix} 0,9 \\ 0,5 \end{bmatrix} & \begin{bmatrix} 0,2 \\ 0,8 \end{bmatrix} \end{matrix}$$

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

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

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

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

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

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:

$$|E_M^{atual} - E_M^{anterior}| \leq \varepsilon \quad (5.40)$$

onde ϵ 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  $\{\mathbf{x}^{(k)}\}$ ;
<2> Associar o vetor de saída desejada  $\{\mathbf{d}^{(k)}\}$  para cada amostra;
<3> Iniciar  $w_{ji}^{(1)}$ ,  $w_{ji}^{(2)}$  e  $w_{ji}^{(3)}$  com valores aleatórios pequenos;
<4> Especificar taxa de aprendizagem  $\{\eta\}$  e precisão requerida  $\{\epsilon\}$ ;
<5> Iniciar o contador de número de épocas  $\{\acute{e}poca \leftarrow 0\}$ ;
<6> Repetir as instruções:
    {
        <6.1>  $E_M^{anterior} \leftarrow E_M$ ; {conforme (5.8)}
        <6.2> Para todas as amostras de treinamento  $\{\mathbf{x}^{(k)}, \mathbf{d}^{(k)}\}$ , fazer:
            {
                <6.2.1> Obter  $I_j^{(1)}$  e  $Y_j^{(1)}$ ; {conforme (5.1) e (5.4)}
                <6.2.2> Obter  $I_j^{(2)}$  e  $Y_j^{(2)}$ ; {conforme (5.2) e (5.5)}
                <6.2.3> Obter  $I_j^{(3)}$  e  $Y_j^{(3)}$ ; {conforme (5.3) e (5.6)}
            } Passo Forward
            {
                <6.2.4> Determinar  $\delta_j^{(3)}$ ; {conforme (5.15)}
                <6.2.5> Ajustar  $w_{ji}^{(3)}$ ; {conforme (5.17)}
                <6.2.6> Determinar  $\delta_j^{(2)}$ ; {conforme (5.26)}
                <6.2.7> Ajustar  $w_{ji}^{(2)}$ ; {conforme (5.28)}
                <6.2.8> Determinar  $\delta_j^{(1)}$ ; {conforme (5.37)}
                <6.2.9> Ajustar  $w_{ji}^{(1)}$ ; {conforme (5.39)}
            } Passo Backward
        <6.3> Obter  $Y_j^{(3)}$  ajustado; {conforme <6.2.1>, <6.2.2> e <6.2.3>}
        <6.4>  $E_M^{atual} \leftarrow E_M$ ; {conforme (5.8)}
        <6.5>  $\acute{e}poca \leftarrow \acute{e}poca + 1$ ;
    }
    Até que:  $|E_M^{atual} - E_M^{anterior}| \leq \epsilon$ 

```

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 $\{ \mathbf{x} \}$; | |
| | <2> Assumir $W_{ji}^{(1)}$, $W_{ji}^{(2)}$ e $W_{ji}^{(3)}$ 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 as 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 *resilient-propagation* 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:

$$W_{ji}^{(L)}(t+1) = W_{ji}^{(L)}(t) + \underbrace{\alpha \cdot (W_{ji}^{(L)}(t) - W_{ji}^{(L)}(t-1))}_{\text{termo de momentum}} + \underbrace{\eta \cdot \delta_j^{(L)} \cdot Y_i^{(L-1)}}_{\text{termo de aprendizagem}} \quad (5.41)$$

onde α 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 $\mathbf{W}^{(L)}$ em direção ao mínimo da função erro. A execução de tal tarefa ficará então a cargo do termo de *momentum*, pois este 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 ínfimas, pois a variação do erro quadrático entre duas iterações sucessivas será baixa e, conseqüentemente, 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^{OT} da função erro quadrático.

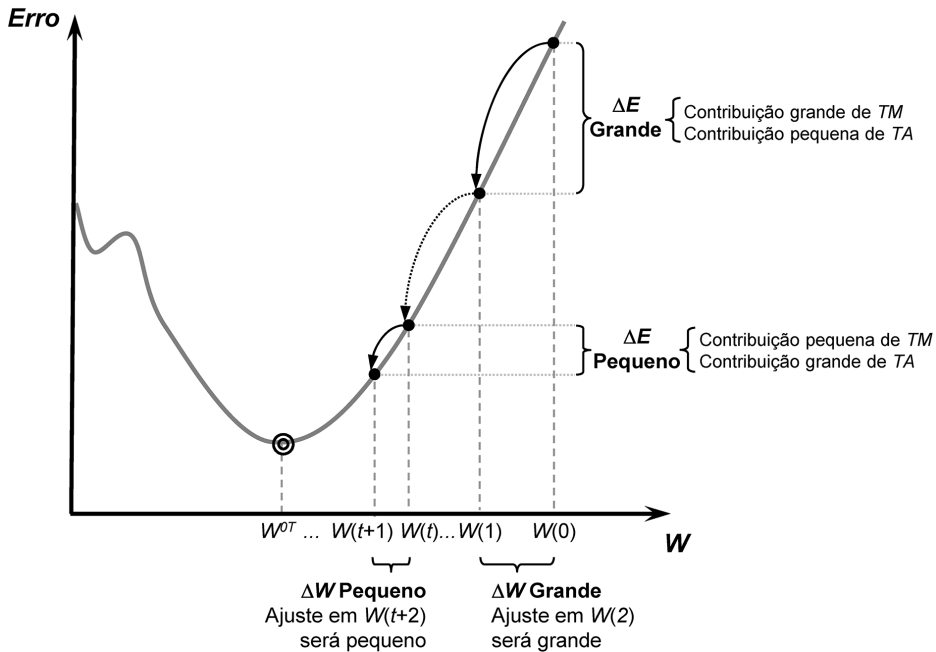


Figura 5.8 – Ilustração do processo de treinamento utilizando o método de inserção do termo de *momentum*

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 alii*, 1986].

(B) Método *resilient-propagation*

As funções de ativação do tipo logística, ou tangente hiperbólica, usadas nos neurônios das redes PMC, 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 $g'(u)$.

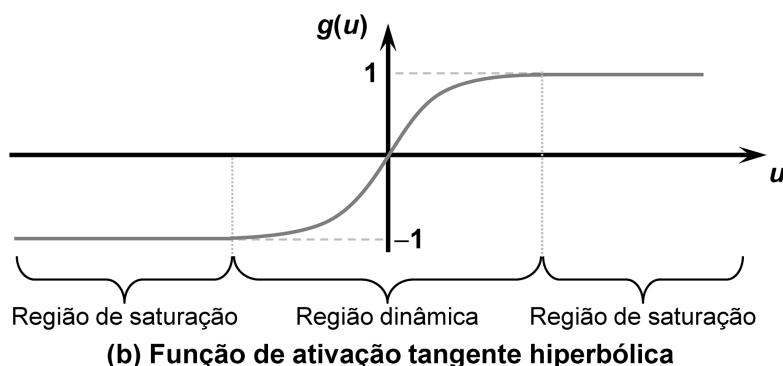
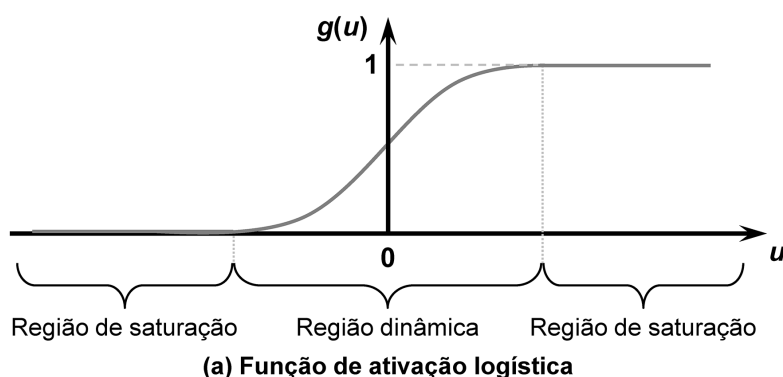


Figura 5.9 – Ilustração dos intervalos de saturação e de variação dinâmica das funções logística e tangente hiperbólica

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 ele, 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.

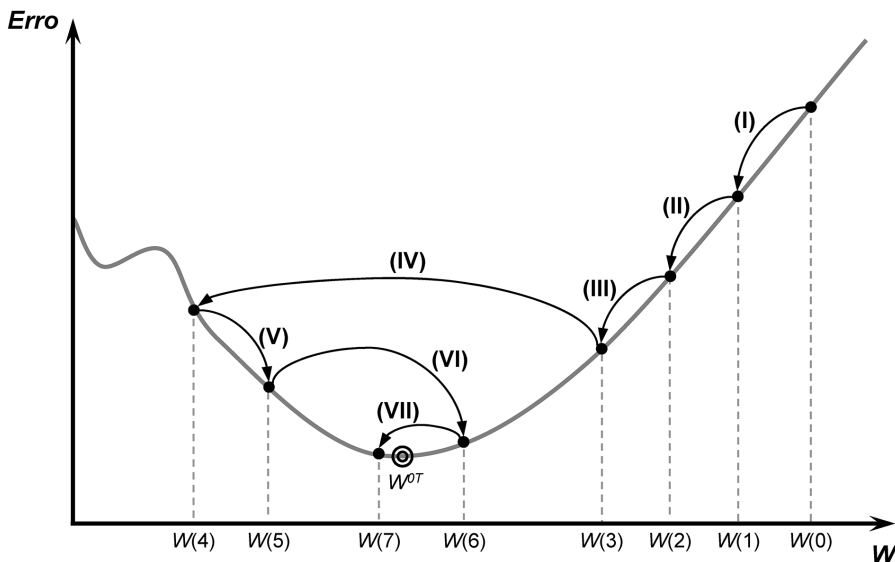


Figura 5.10 – Ilustração do mecanismo de convergência do método *resilient-propagation*

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

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

onde $\Lambda_{ji}^{(L)}$ são consideradas as taxas de aprendizado individuais associadas a cada um dos elementos $W_{ji}^{(L)}$ 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:

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

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:

$$\begin{aligned}
 V &= \frac{1}{2p} \cdot \sum_{k=1}^p \sum_{j=1}^{n_3} (d_j(k) - Y_j^{(3)}(k))^2 \\
 V &= \frac{1}{2p} \cdot \sum_{k=1}^p (\mathbf{d}(k) - \mathbf{Y}^{(3)}(k))^T \cdot (\mathbf{d}(k) - \mathbf{Y}^{(3)}(k)) \\
 V &= \frac{1}{2p} \cdot \sum_{k=1}^p \mathbf{E}^T(k) \cdot \mathbf{E}(k)
 \end{aligned} \tag{5.44}$$

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

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

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; Foresee & Hagan, 1997]. Por sua vez, a minimização de uma função $V(\mathbf{z})$ em relação a um vetor paramétrico $\mathbf{z}$ é dada pelo seguinte procedimento iterativo:

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

onde $\nabla^2 \mathbf{V}(\mathbf{z})$ denota a matriz Hessiana (matriz de derivadas de segunda ordem) e $\nabla \mathbf{V}(\mathbf{z})$ é a matriz Jacobiana (matriz de derivadas de primeira ordem) de $\mathbf{V}(\mathbf{z})$. Assumindo-se que $\mathbf{V}(\mathbf{z})$ 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:

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

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

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

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

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

$$\mathbf{J}(\mathbf{z}) = \begin{bmatrix} \frac{\partial e_1(\mathbf{z})}{\partial z_1} & \frac{\partial e_1(\mathbf{z})}{\partial z_2} & \cdots & \frac{\partial e_1(\mathbf{z})}{\partial z_q} \\ \frac{\partial e_2(\mathbf{z})}{\partial z_1} & \frac{\partial e_2(\mathbf{z})}{\partial z_2} & \cdots & \frac{\partial e_2(\mathbf{z})}{\partial z_q} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_N(\mathbf{z})}{\partial z_1} & \frac{\partial e_N(\mathbf{z})}{\partial z_2} & \cdots & \frac{\partial e_N(\mathbf{z})}{\partial z_q} \end{bmatrix} \quad (5.50)$$

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

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

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:

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

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

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

onde $\mathbf{W} \in \Re^{((n+1) \cdot n_1 + (n_1+1) \cdot n_2 + (n_2+1) \cdot n_3)}$.

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

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

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

$$\mathbf{J}(\mathbf{W}^{(3)}) = \begin{bmatrix} \frac{\partial \mathbf{E}(1)}{\partial W_{1,1}^{(3)}} \cdots \frac{\partial \mathbf{E}(1)}{\partial W_{1,n_2}^{(3)}} & \frac{\partial \mathbf{E}(1)}{\partial W_{2,1}^{(3)}} \cdots \frac{\partial \mathbf{E}(1)}{\partial W_{2,n_2}^{(3)}} & \cdots & \frac{\partial \mathbf{E}(1)}{\partial W_{n_3,1}^{(3)}} \cdots \frac{\partial \mathbf{E}(1)}{\partial W_{n_3,n_2}^{(3)}} \\ \frac{\partial \mathbf{E}(2)}{\partial W_{1,1}^{(3)}} \cdots \frac{\partial \mathbf{E}(2)}{\partial W_{1,n_2}^{(3)}} & \frac{\partial \mathbf{E}(2)}{\partial W_{2,1}^{(3)}} \cdots \frac{\partial \mathbf{E}(2)}{\partial W_{2,n_2}^{(3)}} & \cdots & \frac{\partial \mathbf{E}(2)}{\partial W_{n_3,1}^{(3)}} \cdots \frac{\partial \mathbf{E}(2)}{\partial W_{n_3,n_2}^{(3)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{E}(p)}{\partial W_{1,1}^{(3)}} \cdots \frac{\partial \mathbf{E}(p)}{\partial W_{1,n_2}^{(3)}} & \frac{\partial \mathbf{E}(p)}{\partial W_{2,1}^{(3)}} \cdots \frac{\partial \mathbf{E}(p)}{\partial W_{2,n_2}^{(3)}} & \cdots & \frac{\partial \mathbf{E}(p)}{\partial W_{n_3,1}^{(3)}} \cdots \frac{\partial \mathbf{E}(p)}{\partial W_{n_3,n_2}^{(3)}} \end{bmatrix} \quad (5.56)$$

onde $\mathbf{J}(\mathbf{W}^{(1)}) \in \Re^{(p) \times (n \cdot n_1)}$, $\mathbf{J}(\mathbf{W}^{(2)}) \in \Re^{(p) \times (n_1 \cdot n_2)}$ e $\mathbf{J}(\mathbf{W}^{(3)}) \in \Re^{(p) \times (n_2 \cdot n_3)}$.

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:

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

onde $\mathbf{E} = [\mathbf{E}(1) \ \mathbf{E}(2) \cdots \mathbf{E}(p)]^T$ é o vetor erro referente às p amostras de treinamento.

Finalmente, os elementos das matrizes $\mathbf{J}(\mathbf{W}^{(1)})$, $\mathbf{J}(\mathbf{W}^{(2)})$ e $\mathbf{J}(\mathbf{W}^{(3)})$ 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 $\mathbf{J}(\mathbf{z})$, 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, constata-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 daquelas 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 a ambientes restritos. Nesta 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 abstraído, 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_{OR}), envolvendo a lógica booleana, como ilustrado na figura 5.11.

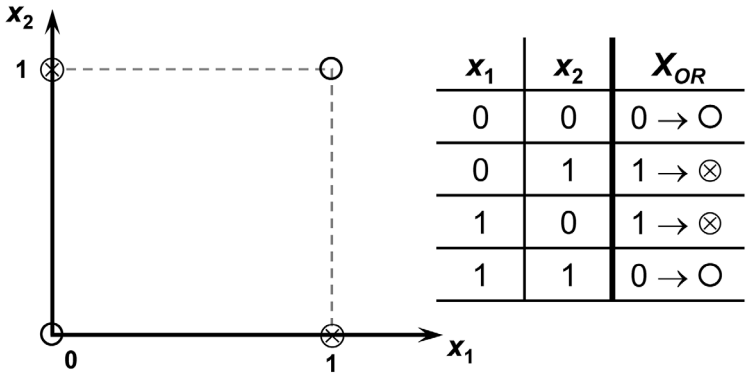


Figura 5.11 – Ilustração do problema do ou-exclusivo

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 PMC de duas camadas neurais, tal como esta representada na figura 5.12.

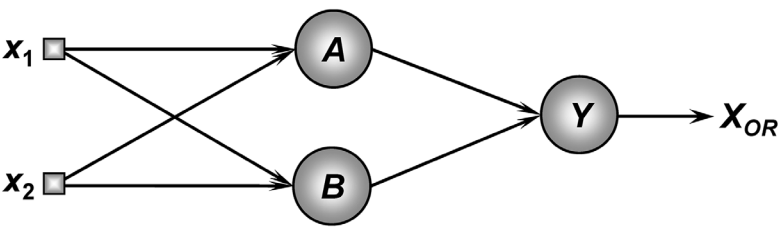


Figura 5.12 – Rede PMC aplicada no problema do ou-exclusivo

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*.

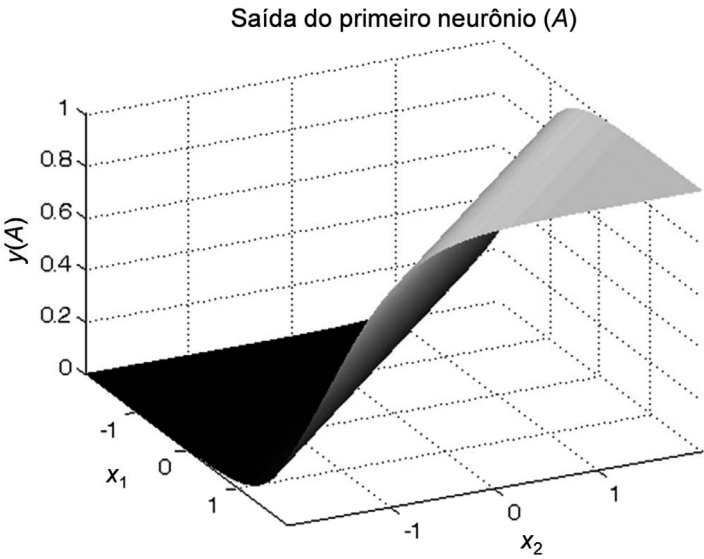


Figura 5.14 – Ilustração da saída proporcionada pela função logística do neurônio A

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

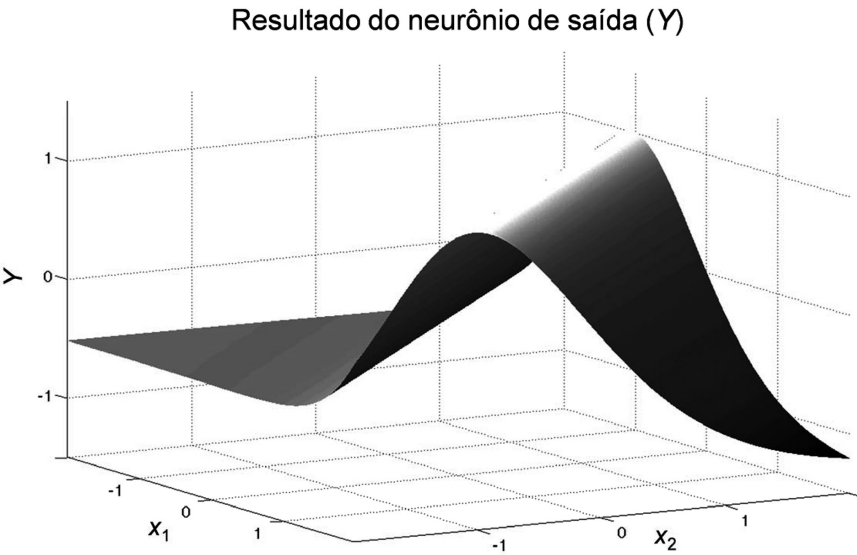


Figura 5.15 – Ilustração da saída do neurônio Y que combina as duas funções logísticas advindas dos neurônios A e B

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 interseção da superfície ilustrada na figura 5.15 próximo ao plano $y = 0$.

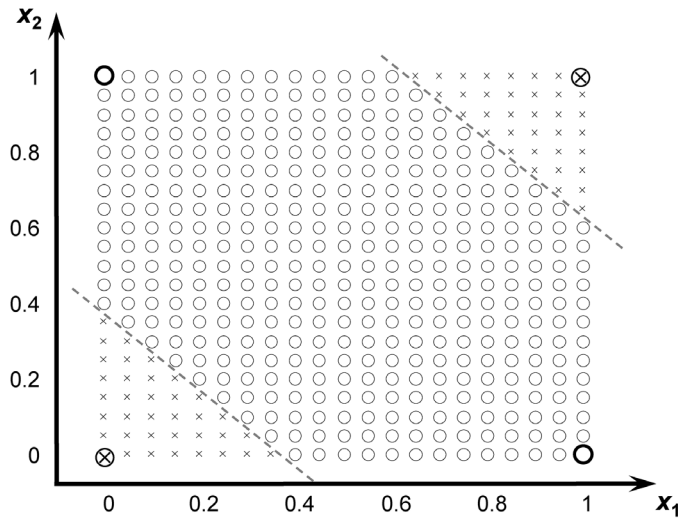


Figura 5.16 – Fronteiras de classificação para o problema do ou-exclusivo

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.

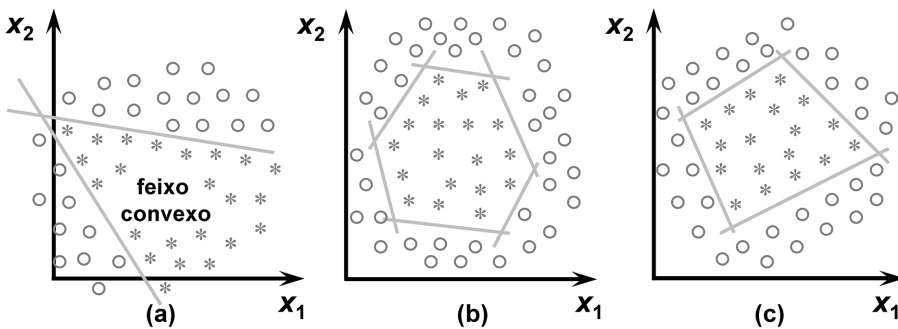


Figura 5.17 – Exemplos de regiões convexas para o problema de classificação de padrões

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 cinco 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.

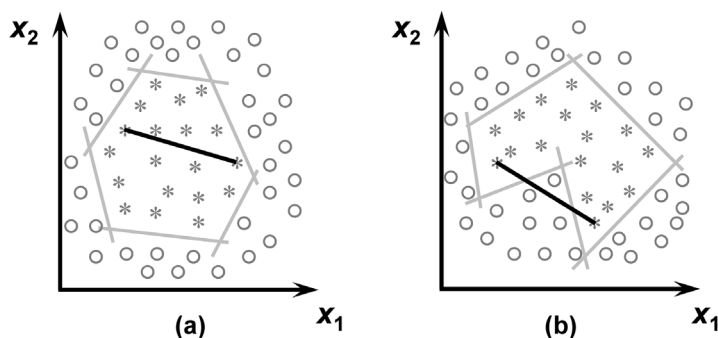


Figura 5.18 – Ilustração de região convexa e região não-convexa

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.

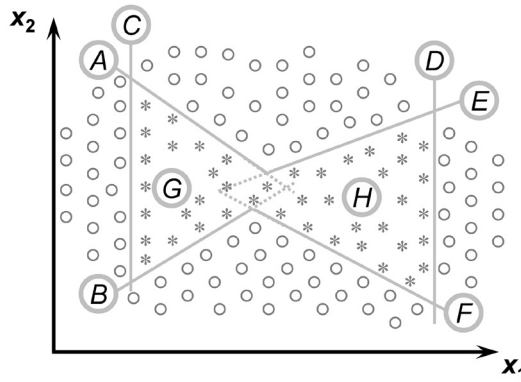


Figura 5.19 – Separabilidade em problemas com fronteiras não-convexas

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.

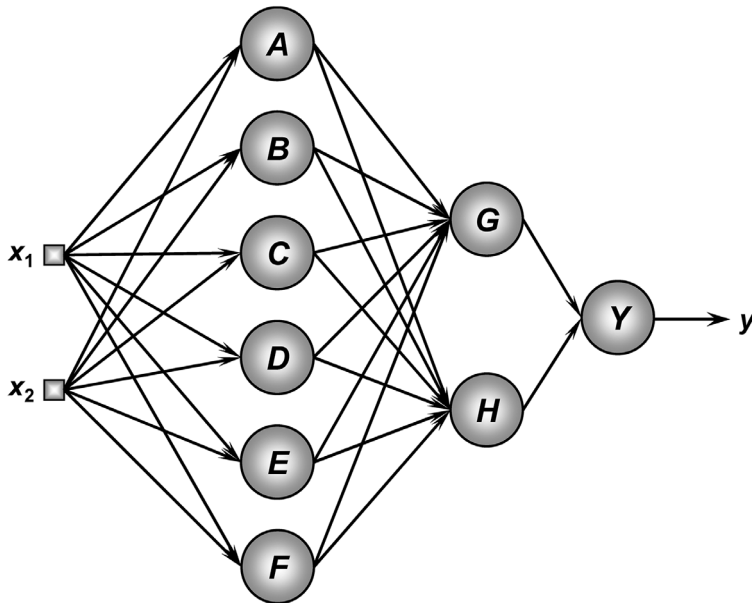


Figura 5.20 – Exemplo de rede PMC aplicada em problemas de classificação de padrões com fronteiras não-convexas

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 a mesma classe.

Finalmente, o neurônio *Y* da figura 5.20 estaria então incumbido de efetuar 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.

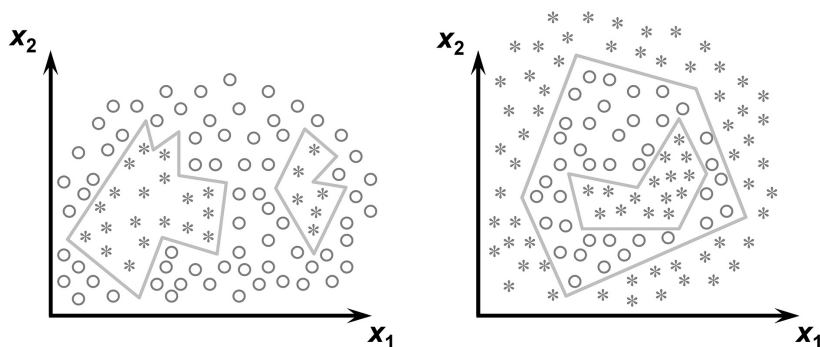


Figura 5.21 – Exemplos de regiões disjuntas em classificação de padrões

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 Lui (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.

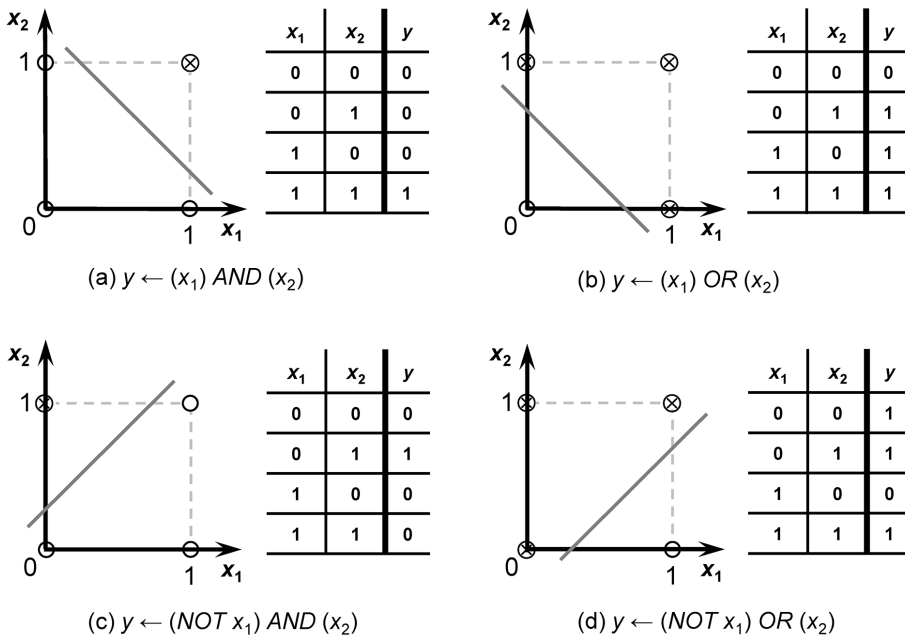


Figura 5.22 – Exemplos de operações lógicas linearmente separáveis

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 & Pearl-mutter, 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.

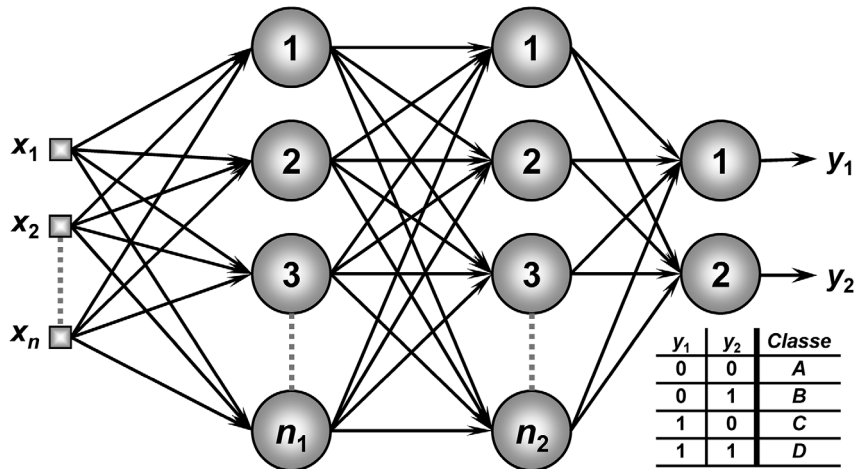


Figura 5.23 – Codificação sequencial para problemas de classificação de padrões envolvendo quatro classes possíveis

Alternativamente, um dos métodos mais utilizados é aquele denominado *one of c-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.

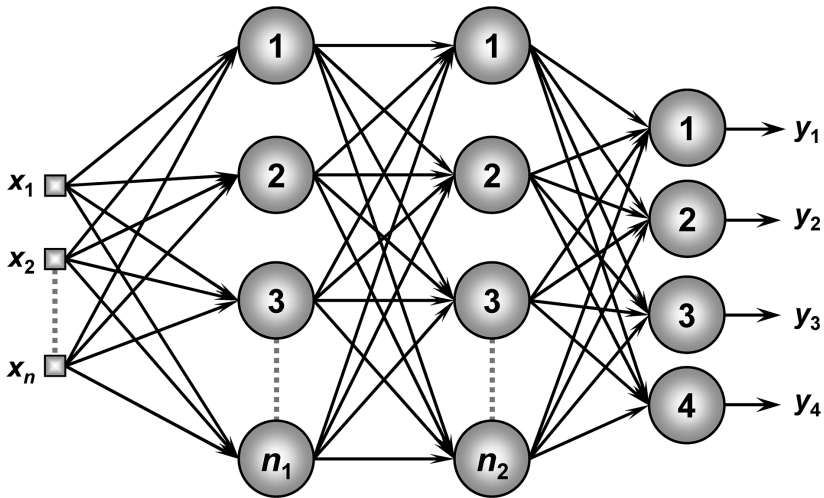


Figura 5.24 – Regra *one of c-classes* para classificação de padrões envolvendo quatro classes possíveis

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		
y_1	y_2	Classe
0	0	A
0	1	B
1	0	C
1	1	D

Codificação <i>one of c-classes</i>				
y_1	y_2	y_3	y_4	Classe
1	0	0	0	A
0	1	0	0	B
0	0	1	0	C
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_2 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 alii*, 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 $y_i^{saída}$ fornecidos 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 $y_i^{pós}$ advindos desta operação de pós-processamento podem ser obtidos pela aplicação da seguinte sistemática:

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

onde $\lim^{sup}$ e $\lim^{inf}$ 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 $\lim^{sup}$, atribui-se então o valor unitário; caso contrário, se for menor ou igual ao $\lim^{inf}$, 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 $\lim^{sup} \in [0,5 \ 0,9]$ e $\lim^{inf} \in [0,1 \ 0,5]$. 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 baseando-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/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/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 as funções de ativação $g(.)$ a serem adotadas 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:

$$y(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n_1} \underbrace{\lambda_i}_{\text{parcela (i)}} \cdot \underbrace{g_i^{(1)}(u_i^{(1)})}_{\text{parcela (ii)}} \quad (5.59)$$

$$u_i^{(1)} = \sum_{j=1}^n w_{ji}^{(1)} \cdot x_j - \theta_i \quad (5.60)$$

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

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 (ii)}, representadas pelos termos $g_i^{(1)}(u_i^{(1)})$, as quais são ponderadas pelos fatores λ_i {parcela (i)}. 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 (1.11), 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.

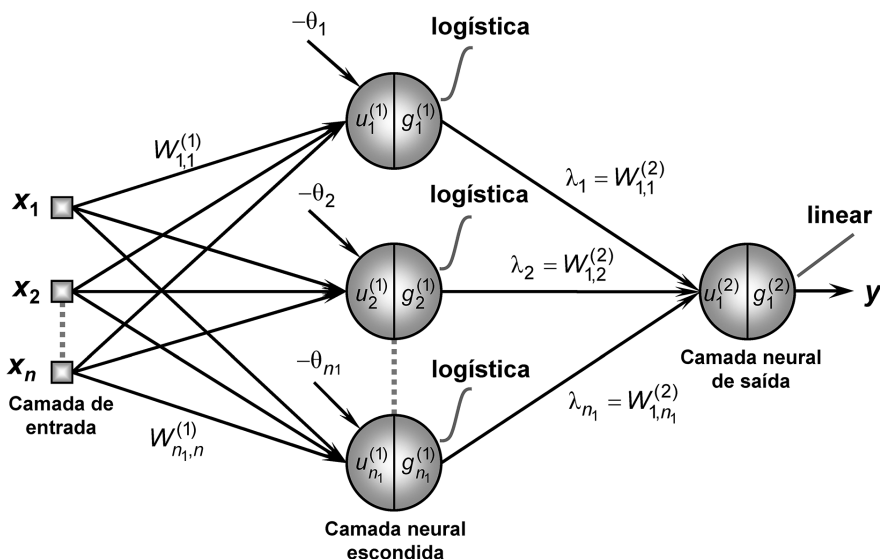


Figura 5.25 – Ilustração de rede PMC aplicada em aproximação funcional

Portanto, após o processo de treinamento da rede PMC, a matriz de pesos referentes ao neurônio de saída corresponderá aos próprios parâmetros λ_i da expressão (5.59), isto é, $\lambda_i = W_{1,i}^{(2)}$.

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.

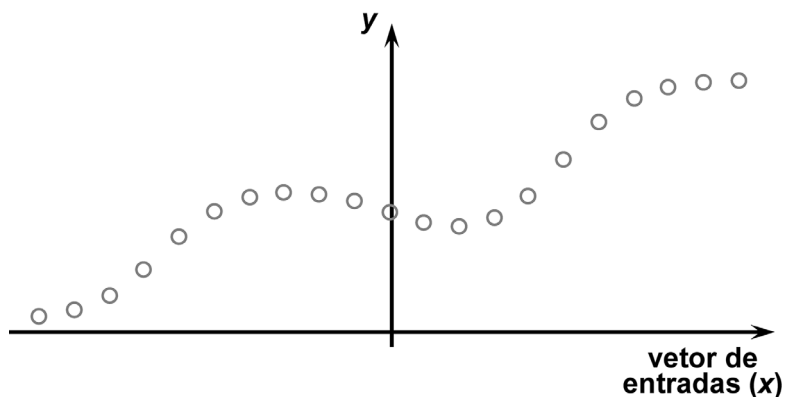


Figura 5.26 – Amostras de treinamento ao problema de aproximação funcional

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.

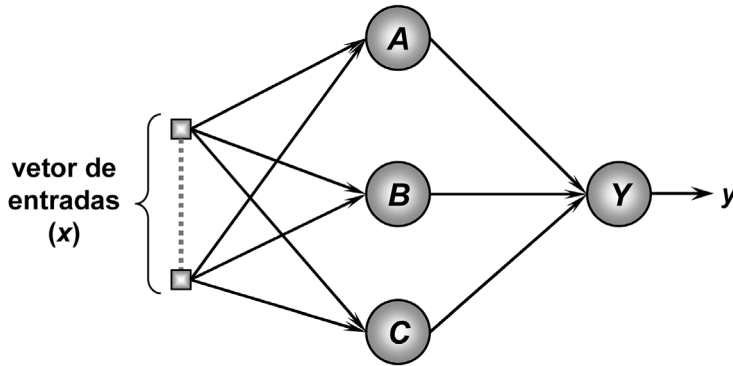


Figura 5.27 – Topologia de PMC usada no problema de aproximação funcional

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 $\{\theta\}$ 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 $\{\lambda_i\}$ do neurônio de saída Y são responsáveis pelo escalamento das mesmas.

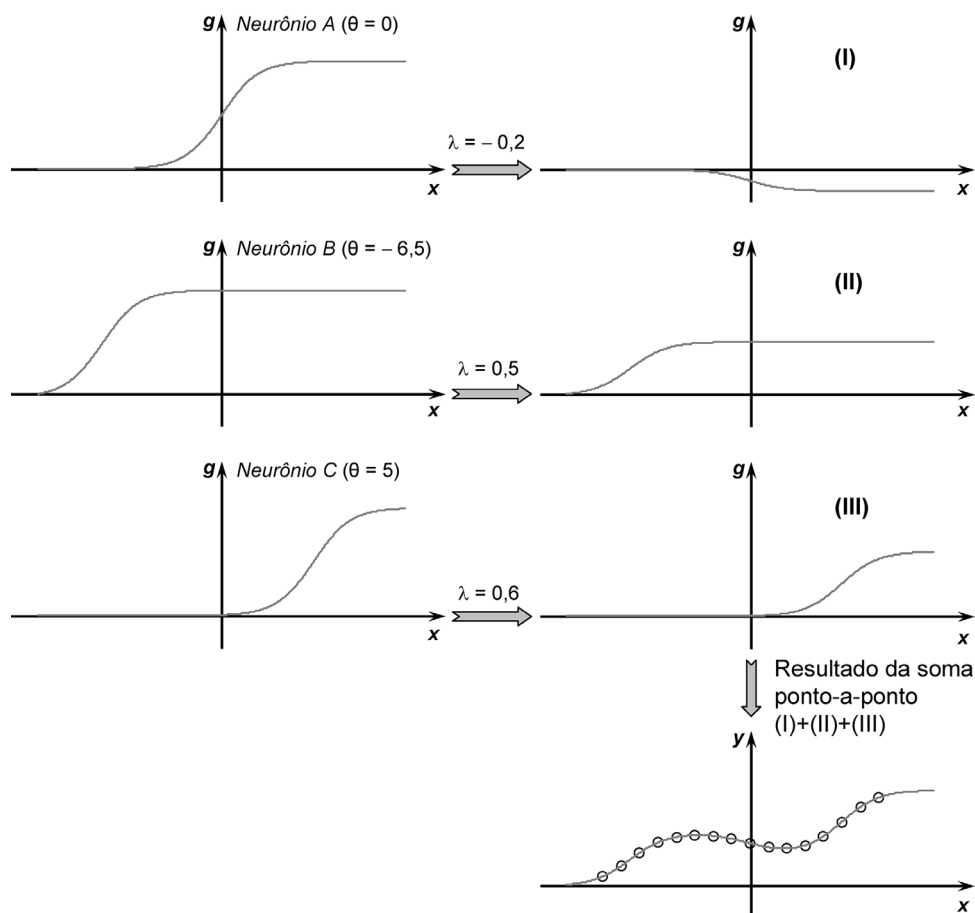


Figura 5.28 – Ilustração de superposição de funções do tipo logística para um problema de aproximação funcional

De forma similar aos problemas de classificação de padrões, embora um PMC com apenas uma camada escondida seja suficiente para mapear 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 necessitado.

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.

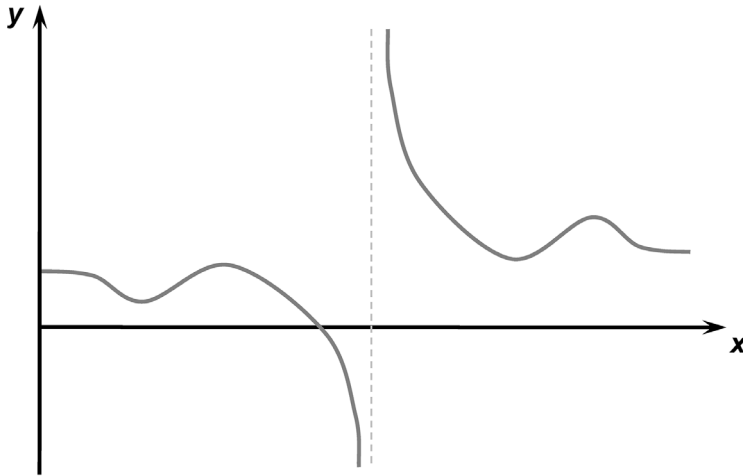


Figura 5.29 – Representação de uma função mapeando o comportamento de problema inverso

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 envolvidos com aproximação de funções e aqueles relacionados com sistemas dinâmicos, a figura 5.30 ilustra o domínio de definição referente aos dados de treinamento/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 $x_i^{\max}$.

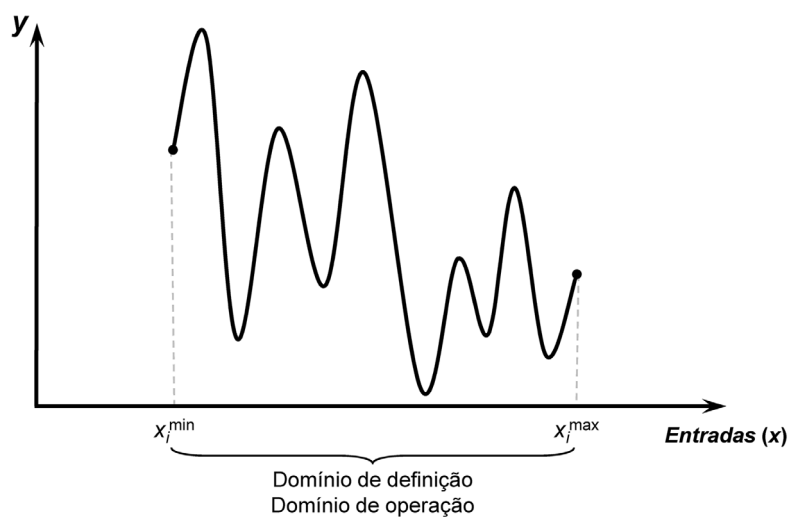


Figura 5.30 – Domínios de definição e de operação de PMC aplicado em problemas de aproximação de funções

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, utilizam-se então os dados de treinamento/teste para ajustar os parâmetros internos da rede. Em seguida, esta estará apta para estimar valores futuros que pertencem ao seu domínio de operação.

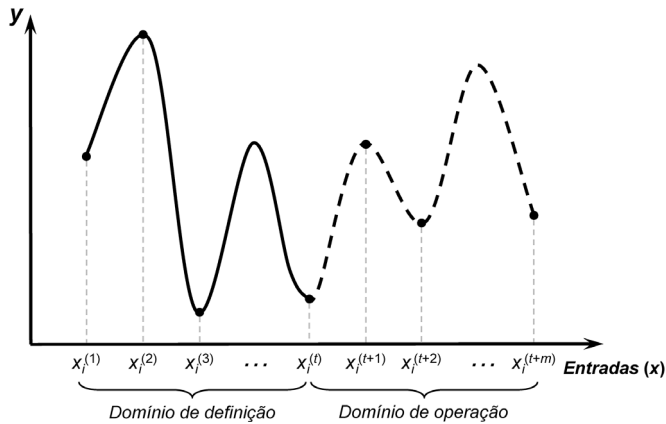


Figura 5.31 – Domínios de definição e de operação de PMC aplicado em problemas envolvendo sistemas dinâmicos

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 estimação 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 $f(\cdot)$ 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-lagged 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.

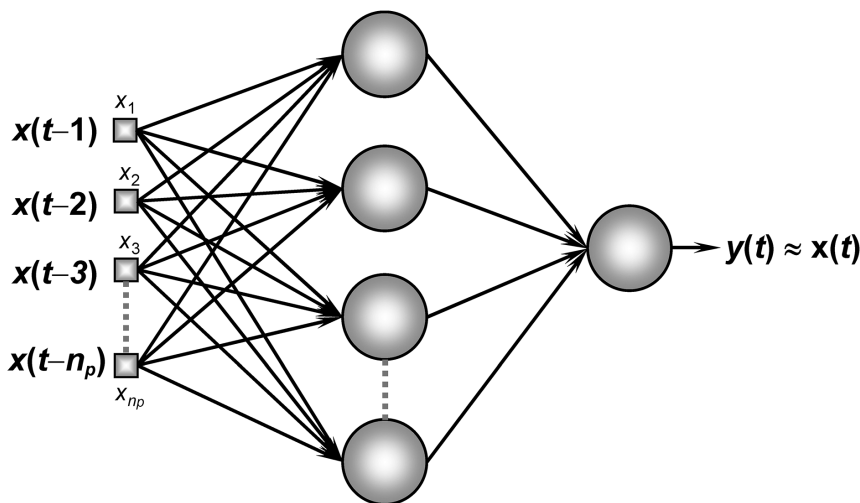


Figura 5.32 – Topologia de PMC com entradas atrasadas no tempo

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:

$$\begin{array}{cccccccc} t = 1 & t = 2 & t = 3 & t = 4 & t = 5 & t = 6 & t = 7 & t = 8 \\ \mathbf{x}(t) = [& 0,11 & 0,32 & 0,53 & 0,17 & 0,98 & 0,67 & 0,83 & 0,79 &]^T \end{array} \quad (5.63)$$

$x(1) \quad x(2) \quad x(3) \quad x(4) \quad x(5) \quad x(6) \quad x(7) \quad x(8)$

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				$4 \leq t \leq 8$ (ordem 3) $n_p = 3$	conjunto de treinamento					
	x_1	x_2	x_3		saída desejada		x_1	x_2	x_3	d
$t = 4$	$x(3)$	$x(2)$	$x(1)$	$x(4)$		$\mathbf{x}^{(1)}$	0,53	0,32	0,11	$d^{(1)} = 0,17$
$t = 5$	$x(4)$	$x(3)$	$x(2)$	$x(5)$		$\mathbf{x}^{(2)}$	0,17	0,53	0,32	$d^{(2)} = 0,98$
$t = 6$	$x(5)$	$x(4)$	$x(3)$	$x(6)$		$\mathbf{x}^{(3)}$	0,98	0,17	0,53	$d^{(3)} = 0,67$
$t = 7$	$x(6)$	$x(5)$	$x(4)$	$x(7)$		$\mathbf{x}^{(4)}$	0,67	0,98	0,17	$d^{(4)} = 0,83$
$t = 8$	$x(7)$	$x(6)$	$x(5)$	$x(8)$		$\mathbf{x}^{(5)}$	0,83	0,67	0,98	$d^{(5)} = 0,79$

O valor $x_0 = -1$, associado ao limiar do neurônio, deverá ser considerado em todos eles.

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

$$\begin{aligned} \mathbf{x}(t) &= [\underbrace{0,11 \quad 0,32 \quad 0,53 \quad 0,17}_{\text{janela 1 (} t = 4 \text{)}} \quad 0,98 \quad 0,67 \quad 0,83 \quad 0,79]^T \\ \mathbf{x}(t) &= [0,11 \quad \underbrace{0,32 \quad 0,53 \quad 0,17 \quad 0,98}_{\text{janela 2 (} t = 5 \text{)}} \quad 0,67 \quad 0,83 \quad 0,79]^T \\ \mathbf{x}(t) &= [0,11 \quad 0,32 \quad \underbrace{0,53 \quad 0,17 \quad 0,98 \quad 0,67}_{\text{janela 3 (} t = 6 \text{)}} \quad 0,83 \quad 0,79]^T \\ \mathbf{x}(t) &= [0,11 \quad 0,32 \quad 0,53 \quad \underbrace{0,17 \quad 0,98 \quad 0,67 \quad 0,83}_{\text{janela 4 (} t = 7 \text{)}} \quad 0,79]^T \\ \mathbf{x}(t) &= [0,11 \quad 0,32 \quad 0,53 \quad 0,17 \quad \underbrace{0,98 \quad 0,67 \quad 0,83 \quad 0,79}_{\text{janela 5 (} t = 8 \text{)}}]^T \end{aligned}$$

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 (posteriores), considerando-se agora o seu domínio de operação $\{t \geq 9\}$. Como exemplo, tem-se:

Predição de valores futuros				
	x_1	x_2	x_3	saída estimada
$t = 9$	$x(8)$	$x(7)$	$x(6)$	$x(9) \approx y(9)$
$t = 10$	$x(9)$	$x(8)$	$x(7)$	$x(10) \approx y(10)$
$t = 11$	$x(10)$	$x(9)$	$x(8)$	$x(11) \approx y(11)$
(...)	(...)	(...)	(...)	(...)

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 estimação 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 recorrentes às entradas

Diferentemente das redes PMC com linha de entradas atrasadas no tempo, a arquitetura com saídas recorrentes 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 “relembrar” 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 recorrentes ou realimentadas. 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:

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

onde n_p é a ordem do *preditor* e indica o número de medidas (amostras) anteriores que serão necessárias para a estimação de $x(t)$. O valor n_q expressa a ordem de contexto, ou seja, a quantidade de saídas passadas que serão também utilizadas na estimação de $x(t)$. 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_q 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.

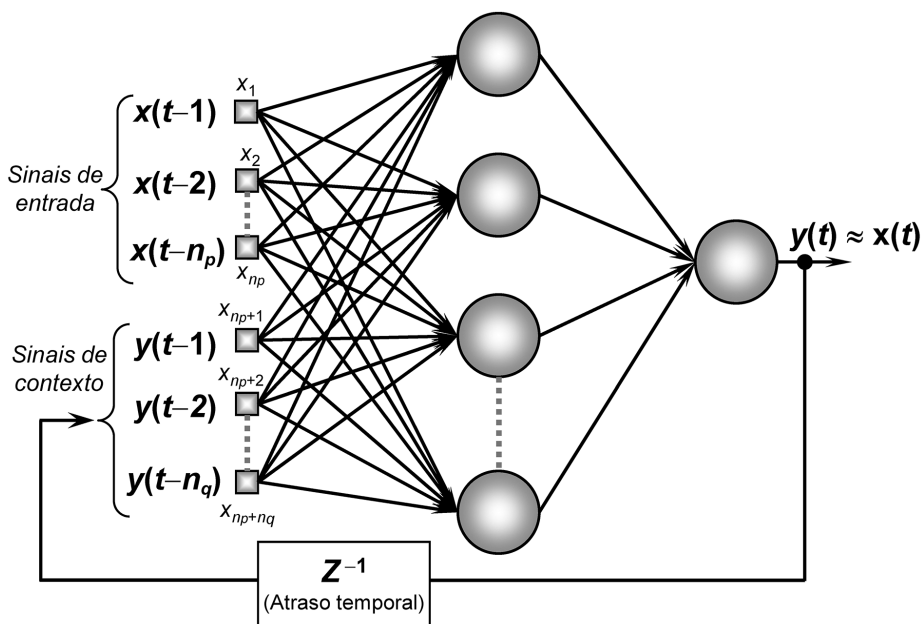


Figura 5.33 – Topologia de PMC com saídas realimentadas às entradas

De fato, em terminologia da área de identificação de sistemas, a rede PMC recorrente que estará 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 $E(t)$ entre o valor esperado de $\mathbf{x}(t)$ frente à resposta $\mathbf{y}(t)$ 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 $n_p = 3$ (ordem de predição usando três entradas passadas) e $n_q = 2$ (ordem de contexto utilizando duas saídas passadas) seria constituído por:

relação entradas/saídas								conjunto de treinamento							
	x_1	x_2	x_3	x_4	x_5	saída desejada	saída da rede		x_1	x_2	x_3	x_4	x_5	d	
$t = 4$	$x(3)$	$x(2)$	$x(1)$	0	0	$x(4)$	$y(4)$	$4 \leq t \leq 8$ (ordem 3) $n_p = 3$ $n_q = 2$	$\mathbf{x}^{(1)}$	0,53	0,32	0,11	0	0	$d^{(1)} = 0,17$
$t = 5$	$x(4)$	$x(3)$	$x(2)$	$y(4)$	0	$x(5)$	$y(5)$		$\mathbf{x}^{(2)}$	0,17	0,53	0,32	$y(4)$	0	$d^{(2)} = 0,98$
$t = 6$	$x(5)$	$x(4)$	$x(3)$	$y(5)$	$y(4)$	$x(6)$	$y(6)$		$\mathbf{x}^{(3)}$	0,98	0,17	0,53	$y(5)$	$y(4)$	$d^{(3)} = 0,67$
$t = 7$	$x(6)$	$x(5)$	$x(4)$	$y(6)$	$y(5)$	$x(7)$	$y(7)$		$\mathbf{x}^{(4)}$	0,67	0,98	0,17	$y(6)$	$y(5)$	$d^{(4)} = 0,83$
$t = 8$	$x(7)$	$x(6)$	$x(5)$	$y(7)$	$y(6)$	$x(8)$	$y(8)$		$\mathbf{x}^{(5)}$	0,83	0,67	0,98	$y(7)$	$y(6)$	$d^{(5)} = 0,79$

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. Sucessivamente, 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_p e n_q 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
$t = 9$	$x(8)$	$x(7)$	$x(6)$	$y(8)$	$y(7)$	$x(9) \approx y(9)$
$t = 10$	$x(9)$	$x(8)$	$x(7)$	$y(9)$	$y(8)$	$x(10) \approx y(10)$
$t = 11$	$x(10)$	$x(9)$	$x(8)$	$y(10)$	$y(9)$	$x(11) \approx y(11)$
(...)	(...)	(...)	(...)	(...)	(...)	(...)

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 Jordan ou rede recorrente simples [Jordan, 1986]. Por outro lado, a configuração poderá também ser denominada de rede de Elman, quando somente os resultados dos neurônios das camadas intermediárias são realimentados para as suas unidades de contexto [Elman, 1990], 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 alli* (2006) e Suykens *et alli* (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 $\rightarrow$ 5 neurônios na camada escondida.

Topologia candidata 2 $\rightarrow$ 10 neurônios na camada escondida.

Topologia candidata 3 $\rightarrow$ 15 neurônios na camada escondida.

Topologia candidata 4 $\rightarrow$ 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.

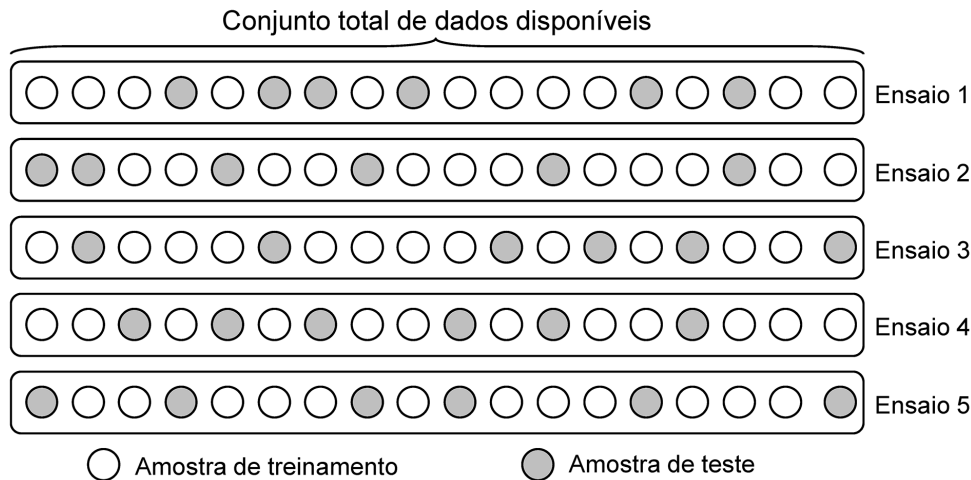


Figura 5.34 – Método de validação cruzada usando amostragem aleatória

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.

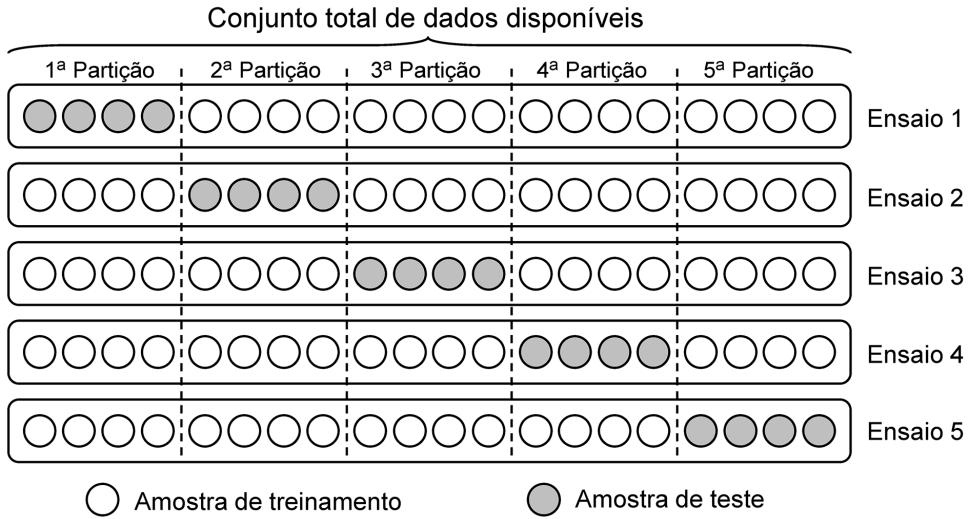


Figura 5.35 – Método de validação cruzada usando k -partições

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 (*leave-one-out 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.

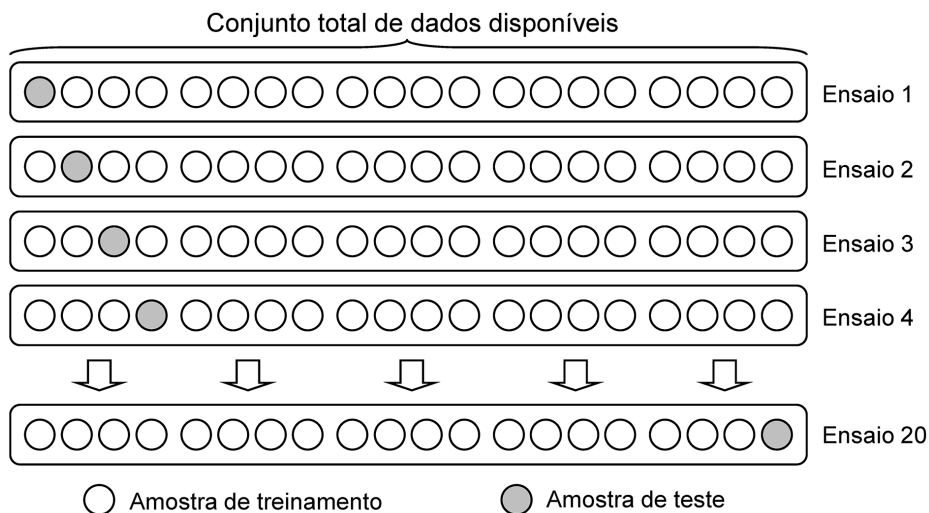


Figura 5.36 – Método de validação cruzada por unidade

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-

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

Alternativamente aos métodos de *cross-validation*, 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.

Inspecionando 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.

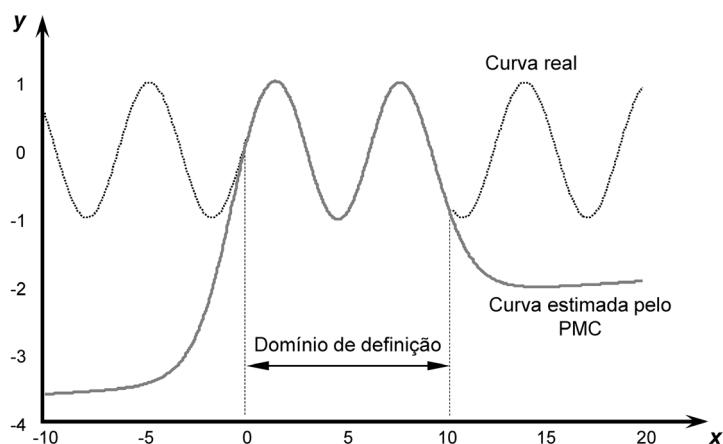


Figura 5.37 – Aplicação de PMC considerando pontos situados dentro e fora de seu domínio de definição

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 decorando 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*.

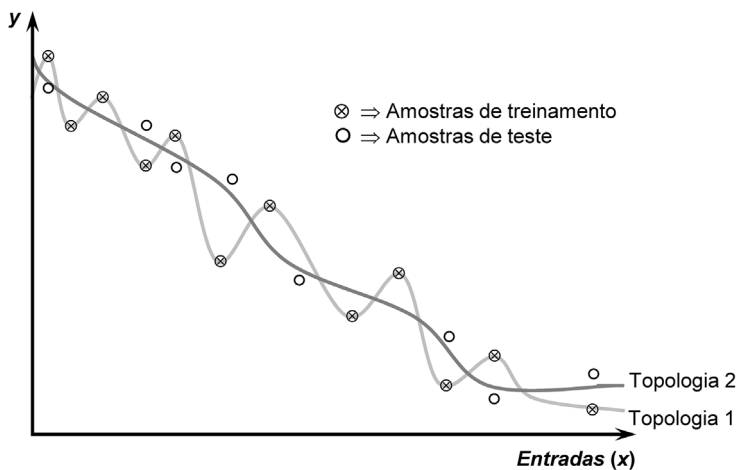


Figura 5.38 – Comportamento do PMC operando em situação de *overfitting*

Percebe-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 ruídos).

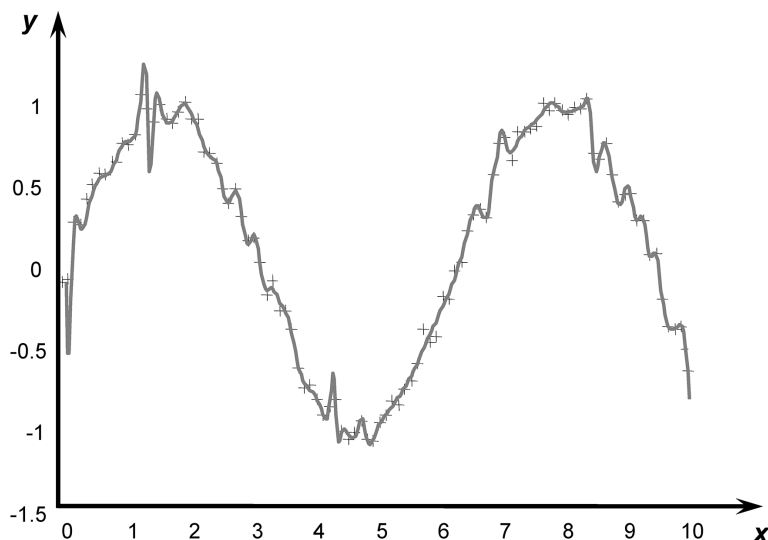


Figura 5.39 – PMC usado para mapear a função seno (com *overfitting*)

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 serão bem mais satisfatórias, visto que seu formato geométrico ficou mais próximo daquele esperado para a função seno. Além disso, o PMC em questão acabou naturalmente descon siderando os ruídos, pois deixou de realizar uma memorização extremamente excessiva ao ponto de incluí-los.

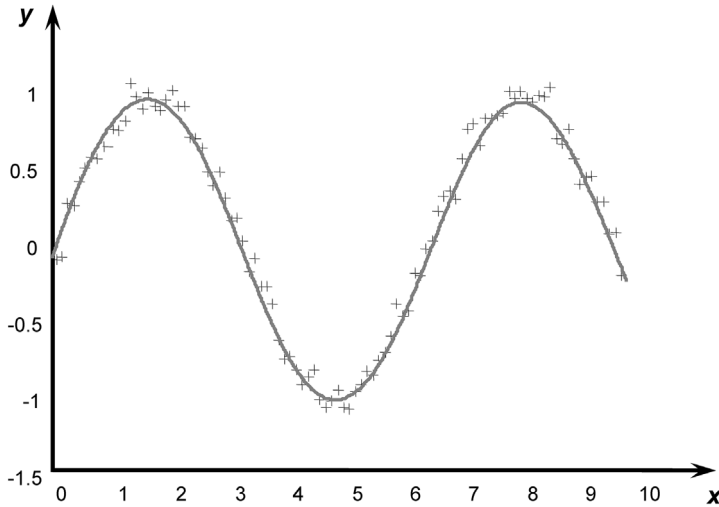


Figura 5.40 – PMC usado para mapear a função seno (sem *overfitting*)

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 será bem significativo.

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 [Finnof *et alii*, 1993]. No exemplo na figura 5.41, o processo de aprendizado terminaria na décima época de treinamento.

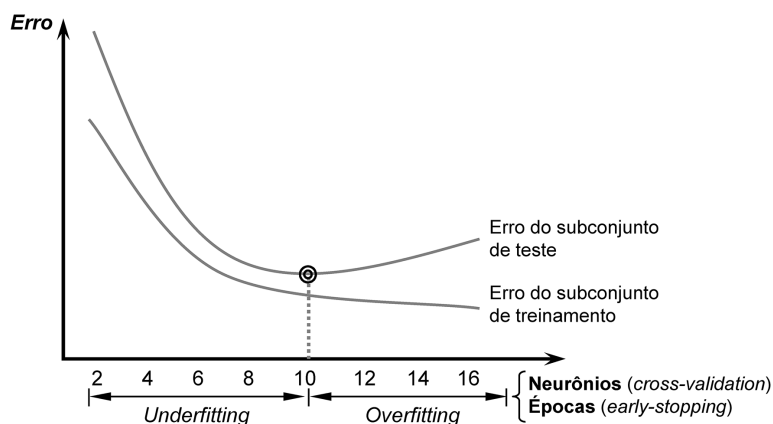


Figura 5.41 – Situações envolvendo *underfitting* e *overfitting*

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 candidada. Entretanto, a seleção da melhor continua atribuída à aplicação da técnica de *cross-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 $p^{(2)}$, certamente, a mesma produziria resultados mais favoráveis que aqueles proporcionados pela sua eventual convergência para $p^{(1)}$ ou $p^{(3)}$, pois o valor do *Erro* em $p^{(2)}$ 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 $\mathbf{W}$ 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 $\mathbf{W}^{(a)}$, a tendência de convergência seria para o ponto de mínimo $p^{(1)}$; ao passo que se tivesse iniciada em $\mathbf{W}^{(b)}$, a propensão seria em direção a $p^{(2)}$.

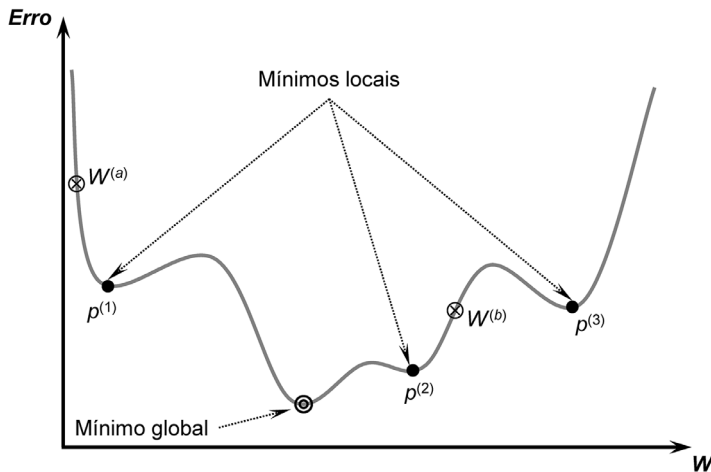


Figura 5.42 – Pontos de mínimos locais associados à função erro

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 recozimento simulado (*simulated annealing*) [Kirkpatrick *et alii*, 1983], em busca tabu [Glover e Laguna, 1998], em algoritmos genéticos [Goldberg, 2002] e em Monte Carlo [Pincus, 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.

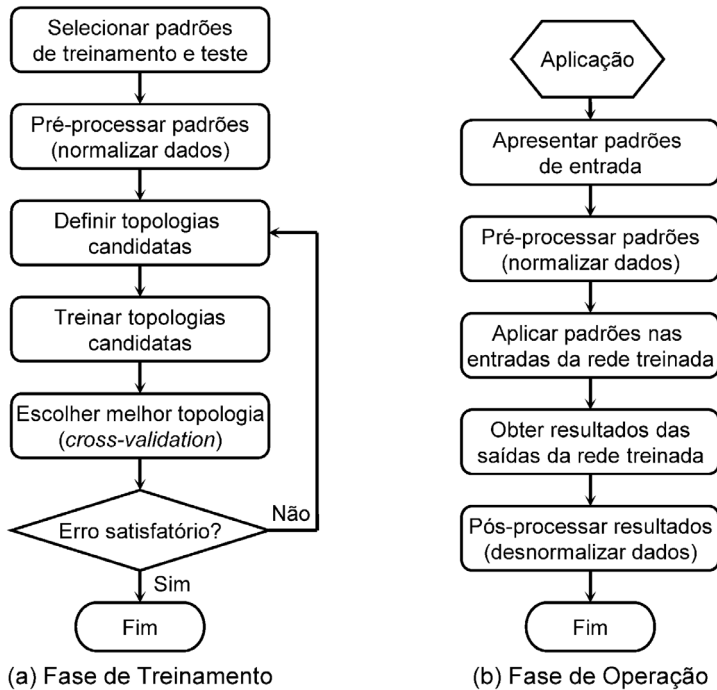


Figura 5.43 – Diagrama de blocos das fases de treinamento e operação

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 $x^{\min}$ e $x^{\max}$, ou seja, $x \in [x^{\min}, x^{\max}]$, 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.

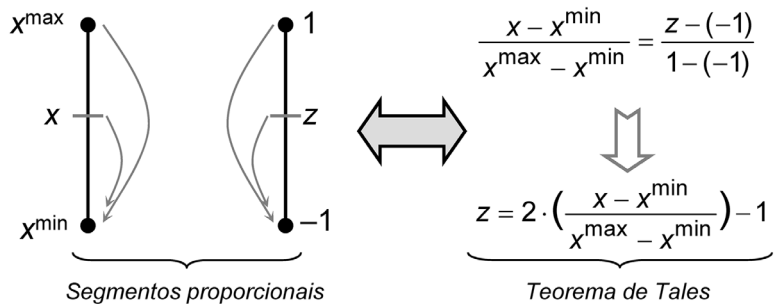


Figura 5.44 – Princípio de normalização dos padrões de treinamento e teste

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.

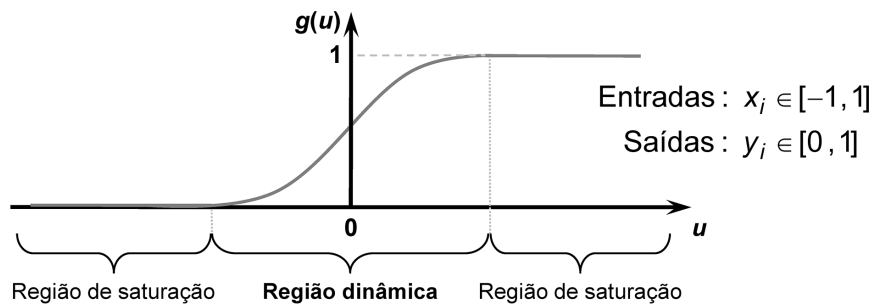


Figura 5.45 – Domínios de normalização para a função de ativação logística

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.

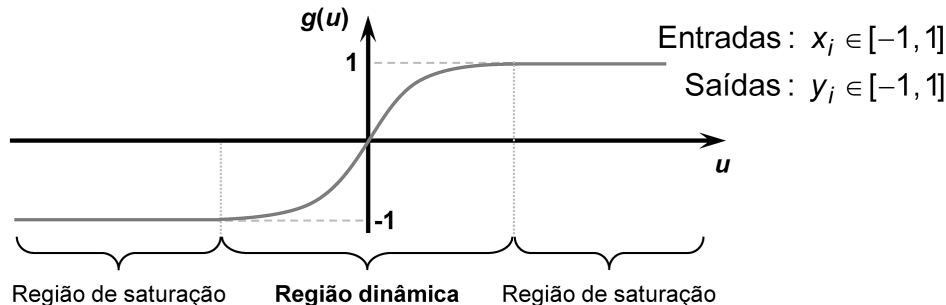


Figura 5.46 – Domínios de normalização para a função de ativação tangente hiperbólica

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 estão generalizando com o mesmo grau de precisão, 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 $W_{ji}^{(L)}$ com valores aleatórios pequenos.

Uma estratégia bem utilizada é aquela advinda das análises efetuadas em LeCun (1989), cujos valores devem estar compreendidos entre $(-2,4/N \leq W_{ji}^{(L)} \leq 2,4/N)$, 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 (η) e termos de momentum (α), 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 $\{\epsilon\}$ 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 *Resilient-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 as amostras de entrada e saída visando evitar as regiões de saturação das funções de ativação;
- Adotar a tangente hiperbólica como função de ativação para os neurônios das camadas escondidas, pois a sua característica de antissimetria (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 estimação (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.

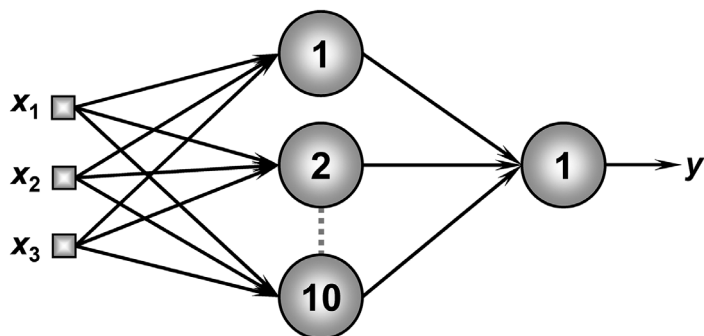


Figura 5.47 – Topologia de PMC (projeto prático 1)

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^{-6} . 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
1° (T1)		
2° (T2)		
3° (T3)		
4° (T4)		
5° (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 $\{E_M\}$ 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. Forneça 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	0,7663					
7	0,6492	0,0007	0,6422	0,5666					
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.

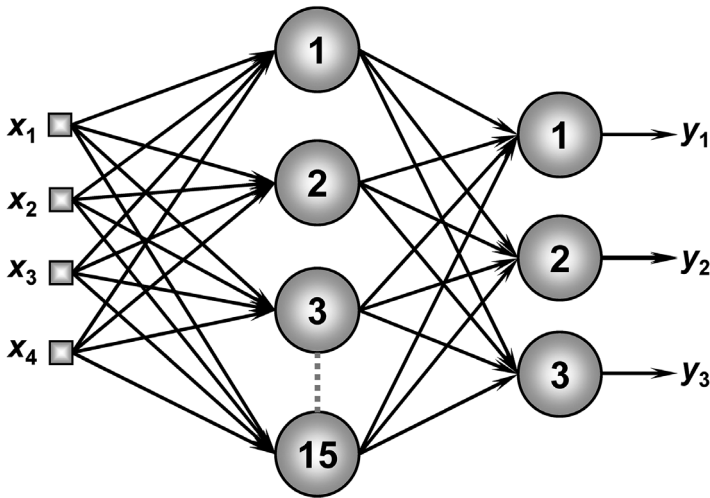


Figura 5.48 – Topologia de PMC (projeto prático 2)

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 *Perceptron*, 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^{-6} .

2) Efetue, em seguida, o treinamento da rede *Perceptron* 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 (sigmóide) para todos os neurônios, com taxa de aprendizado $\{\eta\}$ de 0,1, fator de *momentum* $\{\alpha\}$ de 0,9 e precisão $\{\epsilon\}$ de 10^{-6} .

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:

$$y_i^{pós} = \begin{cases} 1, & \text{se } y_i \geq 0,5 \\ 0, & \text{se } y_i < 0,5 \end{cases} \quad (5.68)$$

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 àquelas 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	$y_1^{pós}$	$y_2^{pós}$	$y_3^{pós}$
1	0,8622	0,7101	0,6236	0,7894	0	0	1			
2	0,2741	0,1552	0,1333	0,1516	1	0	0			
3	0,6772	0,8516	0,6543	0,7573	0	0	1			
4	0,2178	0,5039	0,6415	0,5039	0	1	0			
5	0,7260	0,7500	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	0,0705	0,4717	0,2921	0,2954	1	0	0			
10	0,1187	0,2568	0,3140	0,3037	1	0	0			
11	0,5673	0,7011	0,4083	0,5552	0	1	0			
12	0,3164	0,2251	0,3526	0,2560	1	0	0			
13	0,7884	0,9568	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	0,6597	0,8129	0	0	1			
18	0,7325	0,4761	0,3888	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.

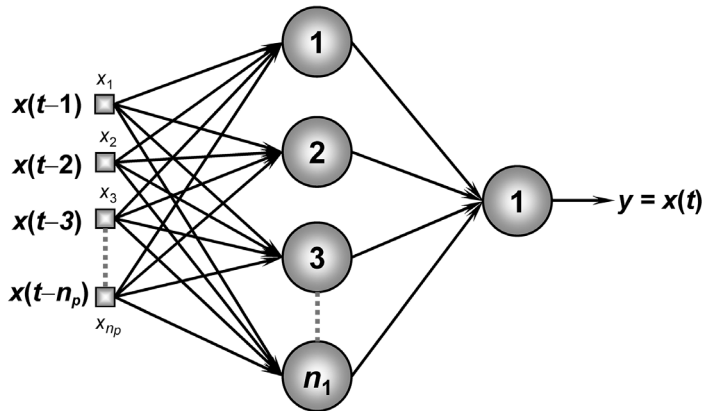


Figura 5.49 – Topologia de PMC (projeto prático 3)

As topologias candidatas passíveis de serem aplicadas no mapeamento deste problema são especificadas por:

TDNN 1 → 5 entradas ($p = 05$), com $n_1 = 10$

TDNN 2 → 10 entradas ($p = 10$), com $n_1 = 15$

TDNN 3 → 15 entradas ($p = 15$), com $n_1 = 25$

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 (sigmóide) para todos os neurônios, com taxa de aprendizado $\{\eta\}$ de 0,1, fator de *momentum* $\{\alpha\}$ de 0,8 e precisão $\{\epsilon\}$ 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_M	Épocas	E_M	Épocas	E_M	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. Forneça, 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)

[illegible]

4) Para cada uma das topologias candidatas da tabela 5.7, considerando ainda o melhor treinamento $\{T1, T2 \text{ ou } T3\}$ 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 melhor treinamento $\{T1, T2 \text{ ou } T3\}$ 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 $\{TDNN 1, TDNN 2 \text{ ou } TDNN 3\}$, e com qual configuração final de treinamento $\{T1, T2 \text{ ou } T3\}$, seria a mais adequada para realizar de previsões neste processo.



Walter Pitts

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, indispondo 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 macroeconômicas.

No mercado de capitais existem dois modelos que regem as estratégias de compras e vendas de ações. O modelo fundamentalista analisa os aspectos econômicos e contábeis das empresas, bem como a influência de variáveis macroeconômicas. 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.

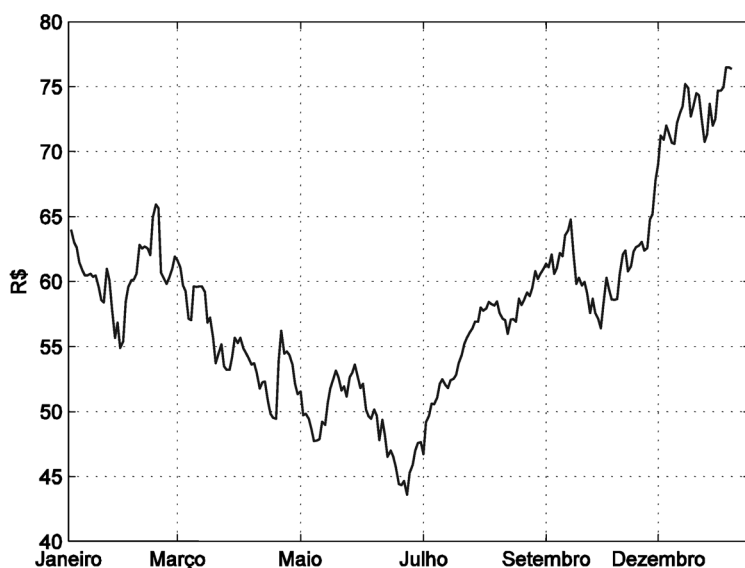


Figura 13.1 – Gráfico de evolução do preço das ações do grupo Pão de Açúcar no ano de 2005

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 estimação dos valores de ações é uma rede recorrente (simples), também denominada de Jordan (subseção 5.4.3), semelhante àquela mostrada na figura 13.2.

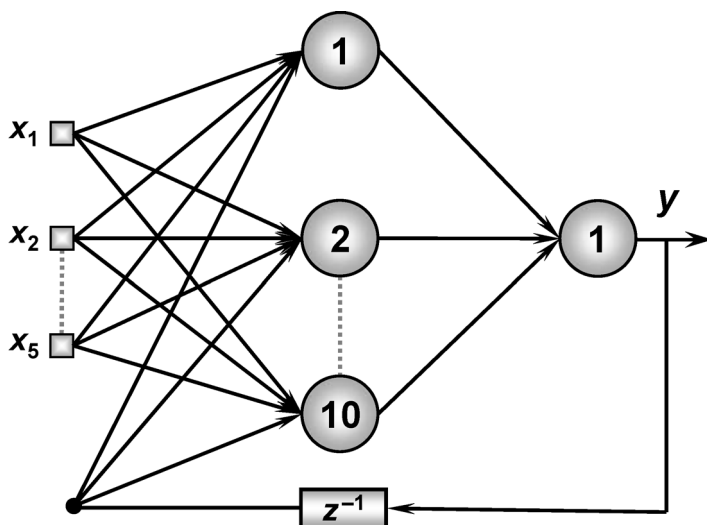


Figura 13.2 – Ilustração de rede neural recorrente

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-

ço 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 estimação das ações para o Grupo Pão de Açúcar, levando-se em consideração apenas os dados históricos de preço das ações.

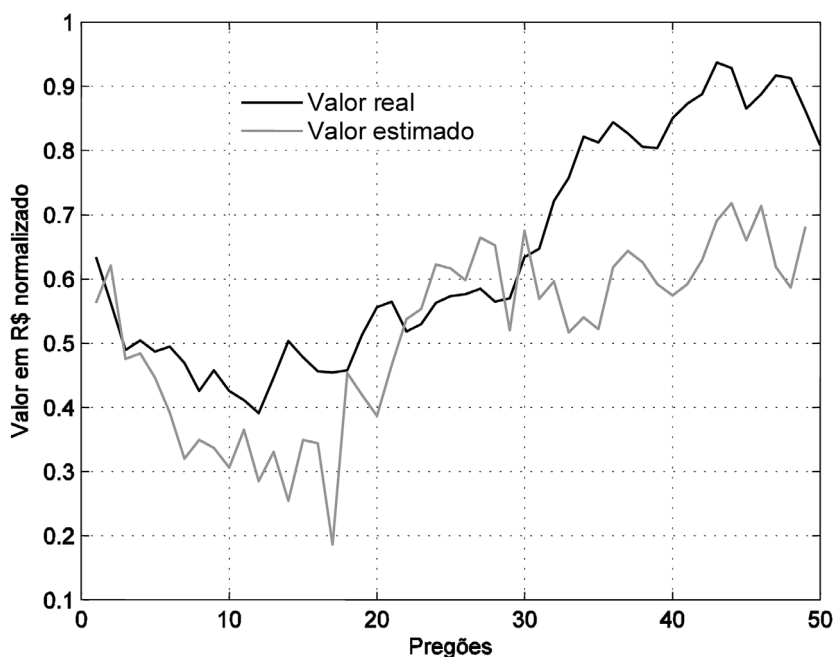


Figura 13.3 – Estimação de preços do Grupo Pão de Açúcar utilizando-se dados históricos

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 macroeconômicas descritas anteriormente. A previsão apresentou uma melhora de 85% em relação à utilização apenas das variáveis históricas.

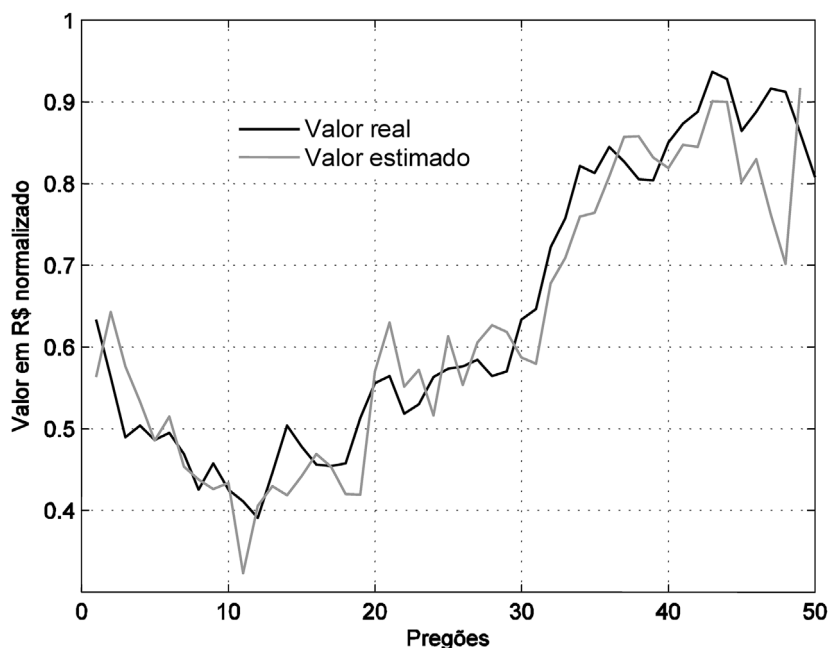


Figura 13.4 – Estimação de preços do Grupo Pão de Açúcar utilizando-se dados históricos e variáveis econômicas

Os resultados utilizando-se as variáveis econômicas 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-sa, contando com a análise da previsão do preço das ações com e sem a influência das variáveis econômicas.

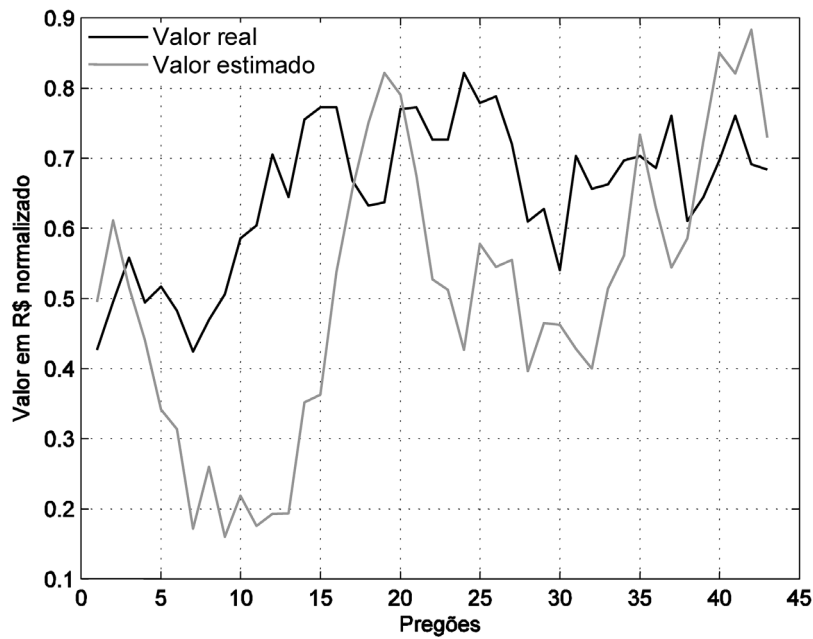


Figura 13.5 – Estimação de preços do Grupo Itausa utilizando-se dados históricos

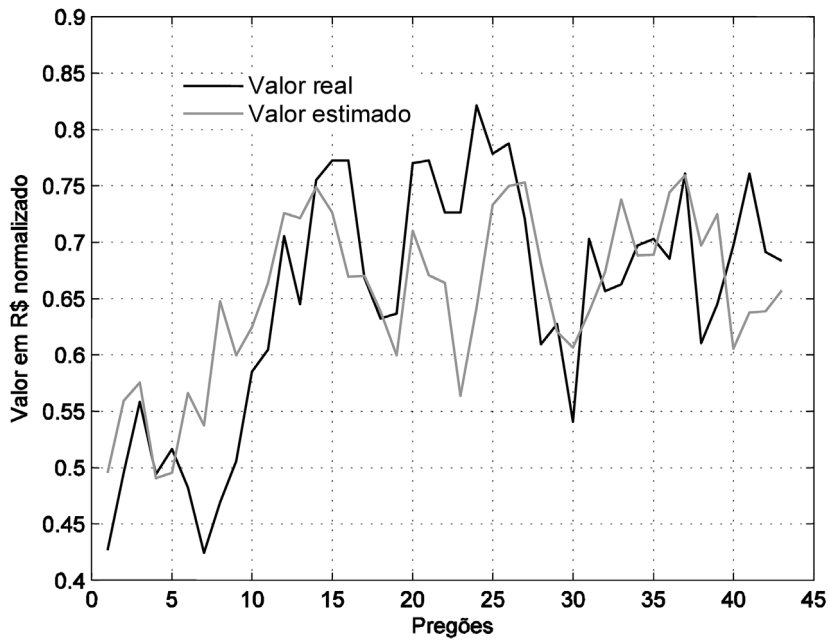


Figura 13.6 – Estimação de preços do Grupo Itausa utilizando-se dados históricos e variáveis econômicas

Mais uma vez constata-se um melhor desempenho na estimação quando da presença das variáveis econômicas na rede neural recorrente.

Nas figuras 13.7 e 13.8 encontram-se os resultados de estimação para a Embraer.

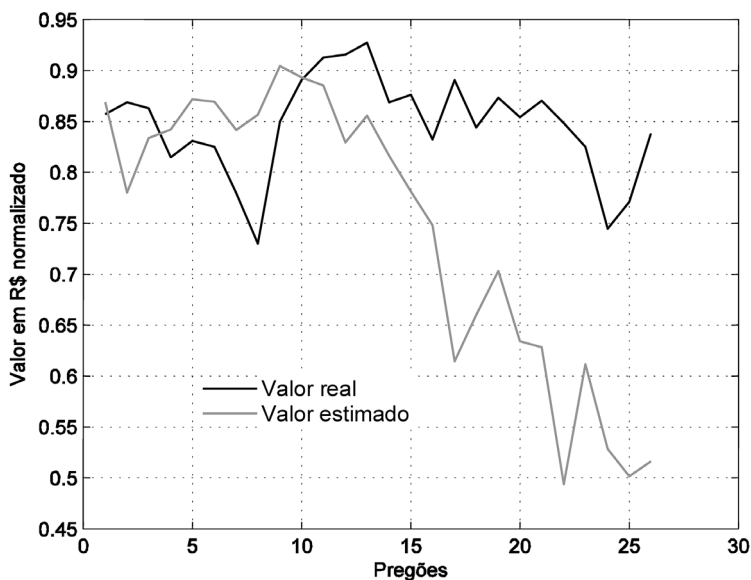


Figura 13.7 – Estimação de preços da Embraer utilizando-se dados históricos

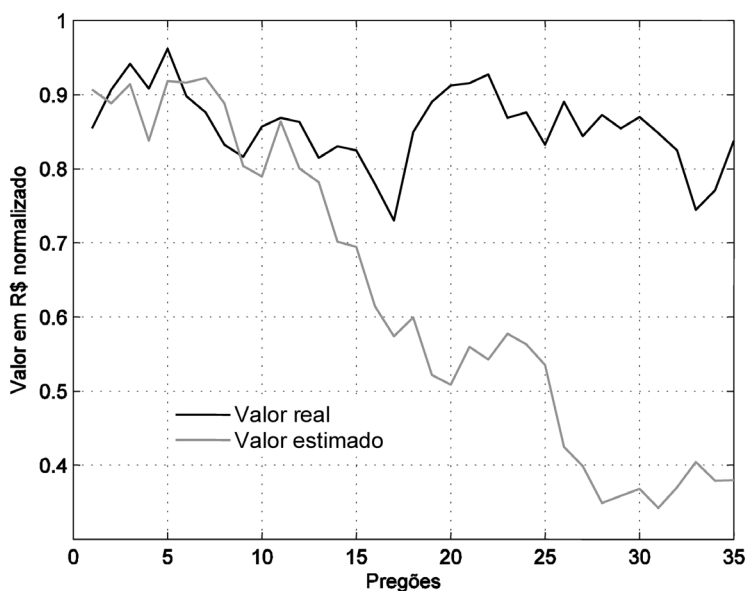


Figura 13.8 – Estimação de preços da Embraer utilizando-se dados históricos e variáveis econômicas

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 estimação 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 estimação, 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.

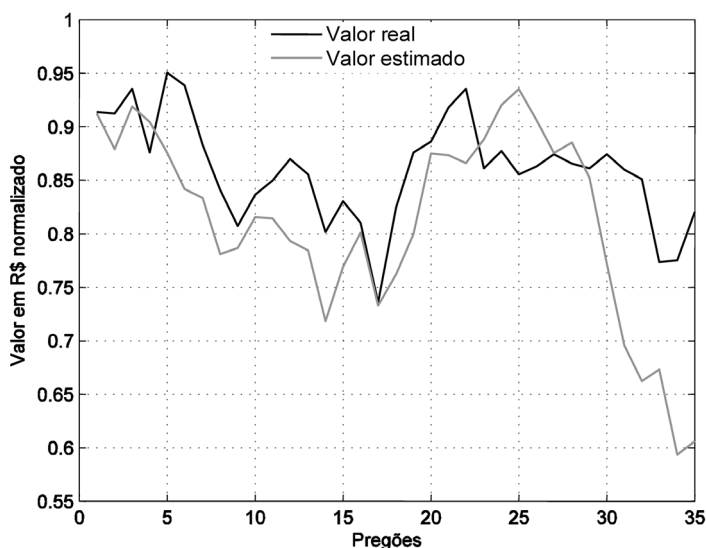


Figura 13.9 – Estimação de preços das ações da Embraer com a base de dados incrementada

A utilização das variáveis econômicas possibilitou uma melhora substancial nos resultados de estimação, 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 estimação, 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 mastóide;
- 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 congênitas, deformidades e anomalias cromossômicas.
- 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.

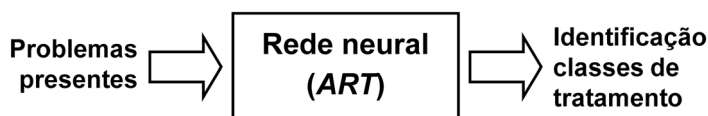


Figura 14.1 – Sistema de apoio ao diagnóstico do paciente

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 ₁	D ₂	D ₃	D ₄	D ₅	(...)	D ₂₁
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.

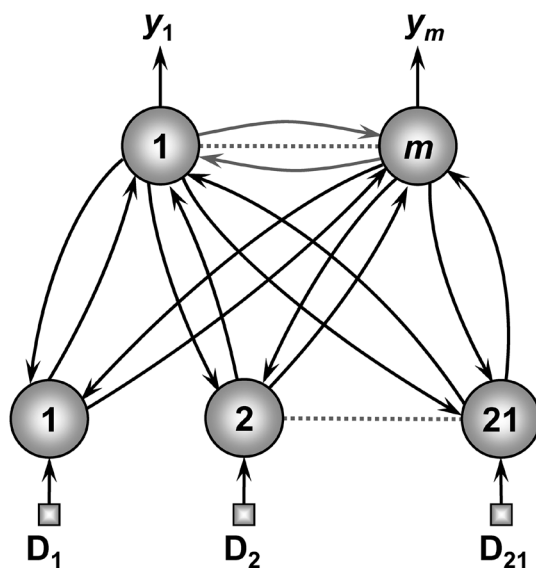


Figura 14.2 – Topologia da rede *ART* utilizada no diagnóstico de doenças

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 ativas e quais situações estavam nos respectivos agrupamentos. Na figura 14.3 é possível conferir os resultados para diferentes parâmetros de vigilância.

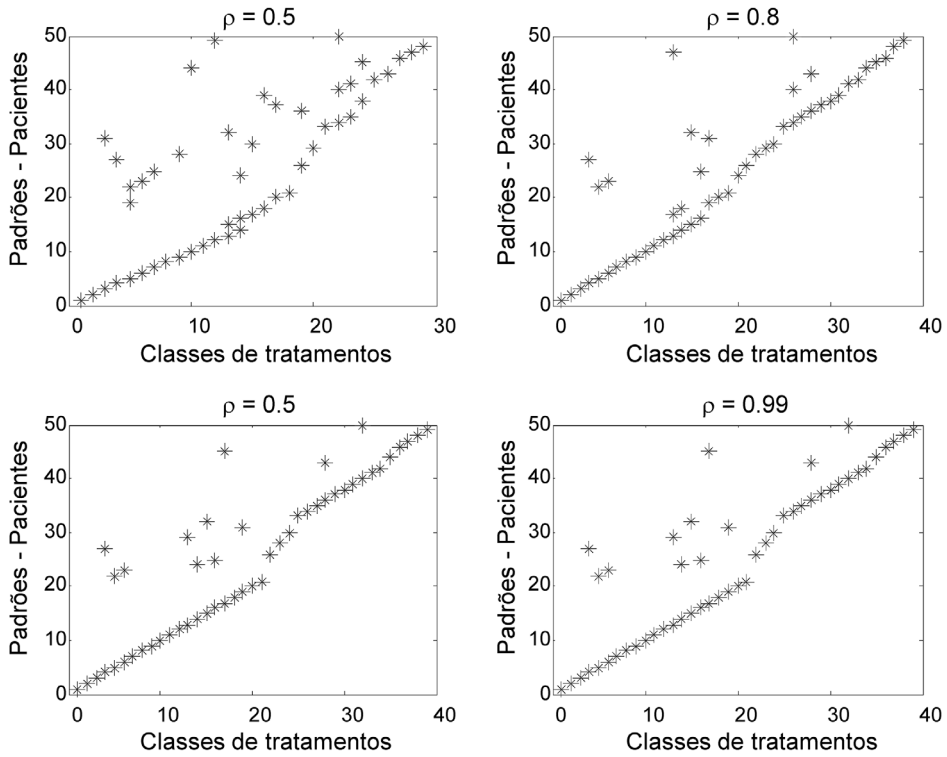


Figura 14.3 – Resposta da rede ART para diferentes parâmetros de vigilância

A figura 14.4 apresenta o número de classes requeridas para os pacientes avaliados pela rede ART, conforme a variação do parâmetro de vigilância.

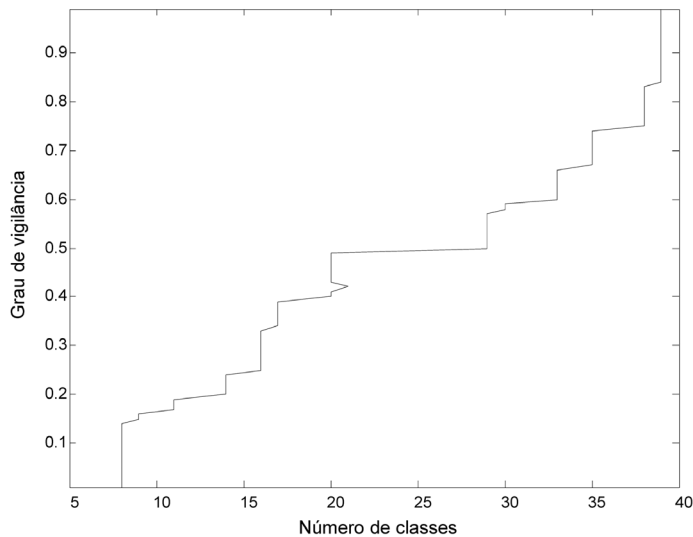


Figura 14.4 – Número de classes em função do grau de vigilância

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.

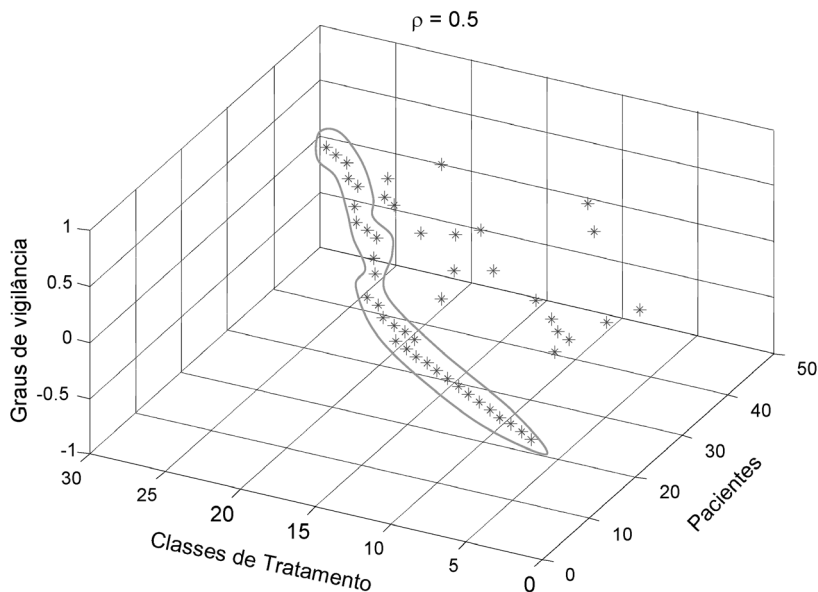


Figura 14.5 – Classes de tratamento para um parâmetro de vigilância de 0,5

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 objetivo 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).

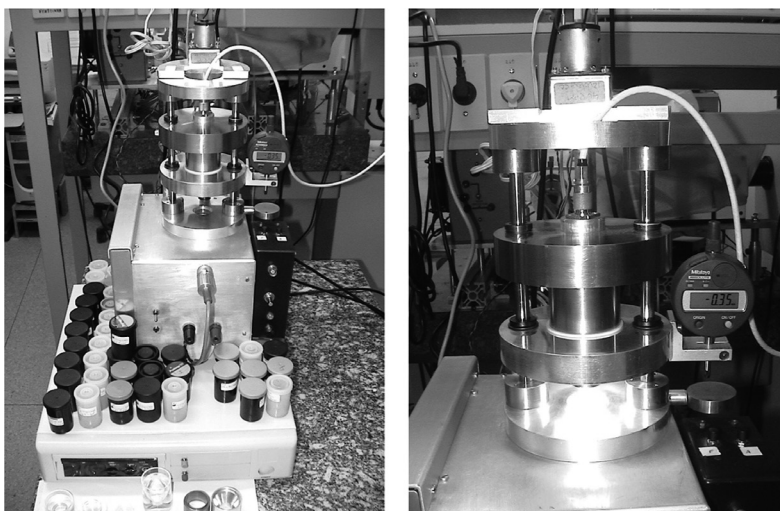


Figura 15.1 – Instrumento Ali-C para identificar padrões de adulterantes

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 Kononen. 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 pirlétrico, 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 borra 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 descritores 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.

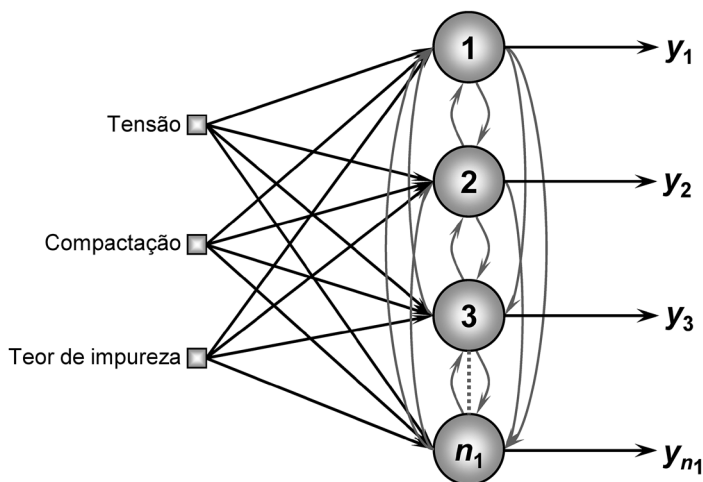


Figura 15.2 – Topologia da rede de Kohonen empregada na classificação

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 ($n_1 = 16$).

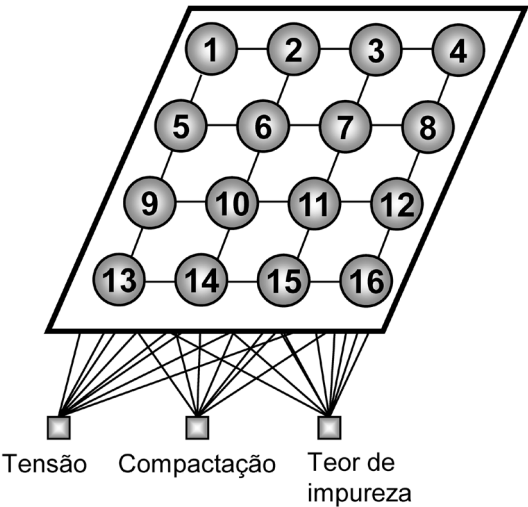


Figura 15.3 – Mapa topológico em grid constituído de 4 x 4 neurônios

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.

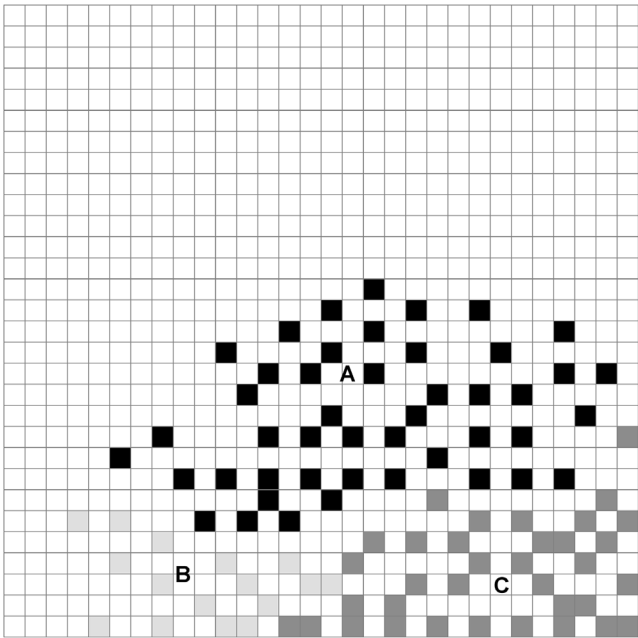


Figura 15.4 – Agrupamento de classes no mapa de Kohonen

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 *clusters* 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	0,996	1,000	A	A
9	0,631	0,596	0,686	B	B
10	0,786	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 *clusters* identificados condizem com as classes verdadeiras (reais).

Em suma, os resultados obtidos pela rede de Kohonen foram considerados bem satisfatórios, pois corroboram 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, conseqüentemente, 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 tratarem 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-circuito) no sistema elétrico. A figura 16.1 ilustra um exemplo de afundamento de tensão.

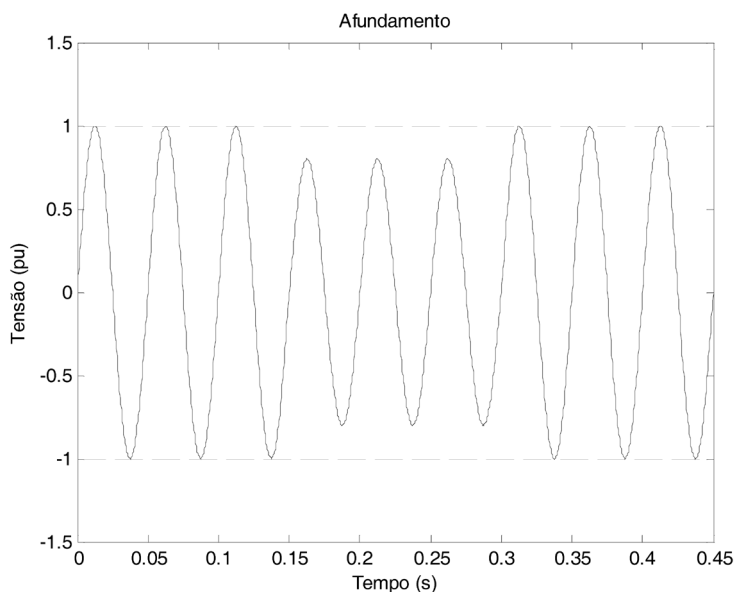


Figura 16.1 – Exemplo de afundamento de tensão

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.

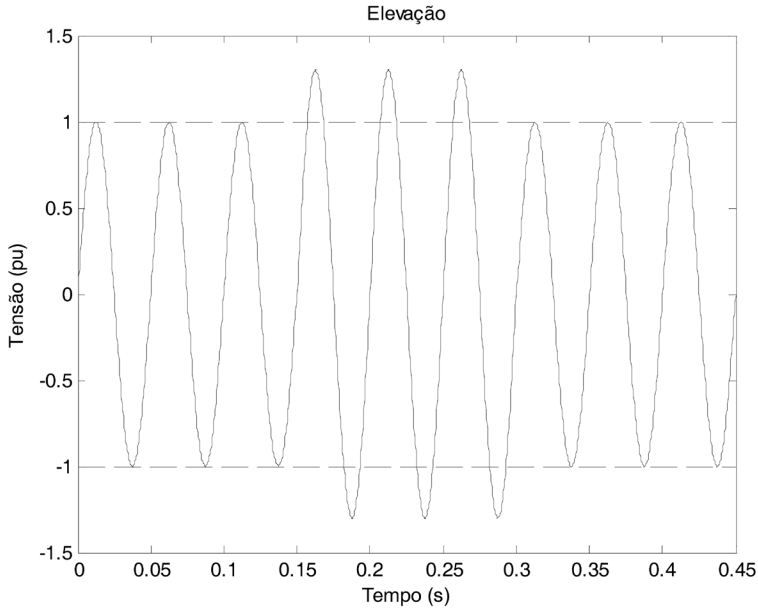


Figura 16.2 – Exemplo de elevação de tensão

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.

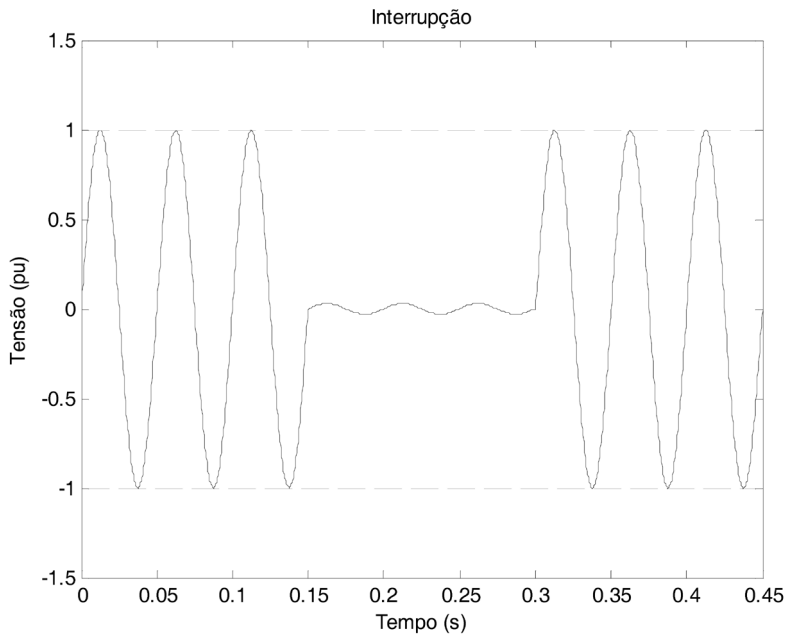


Figura 16.3 – Exemplo de interrupção de tensão

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.

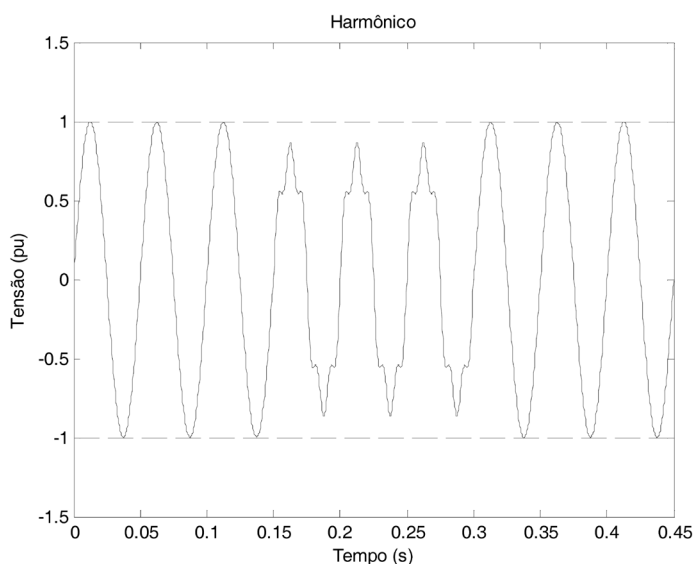


Figura 16.4 – Exemplo de distorção harmônica

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.

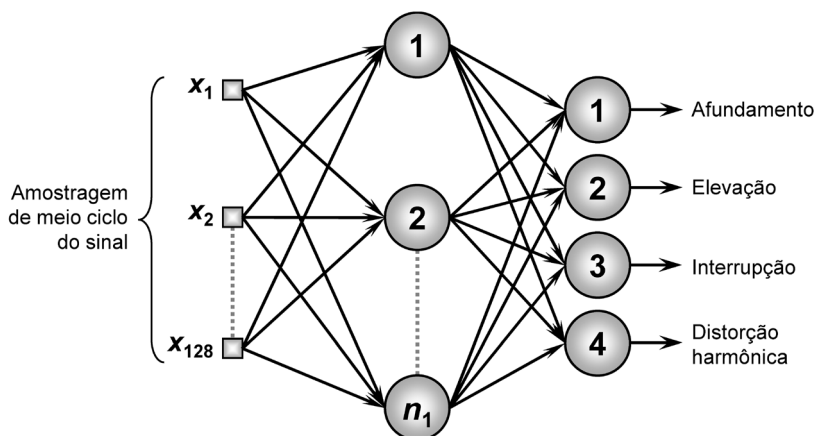


Figura 16.5 – Topologia da rede neural

O parâmetro n_1 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 (*alternative 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, conseqüentemente, 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.

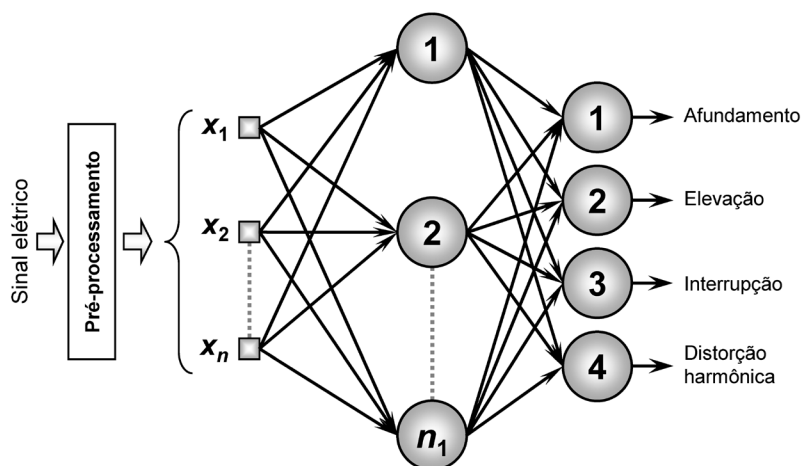


Figura 16.6 – Topologia alternativa para investigação de distúrbios de QEE

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.

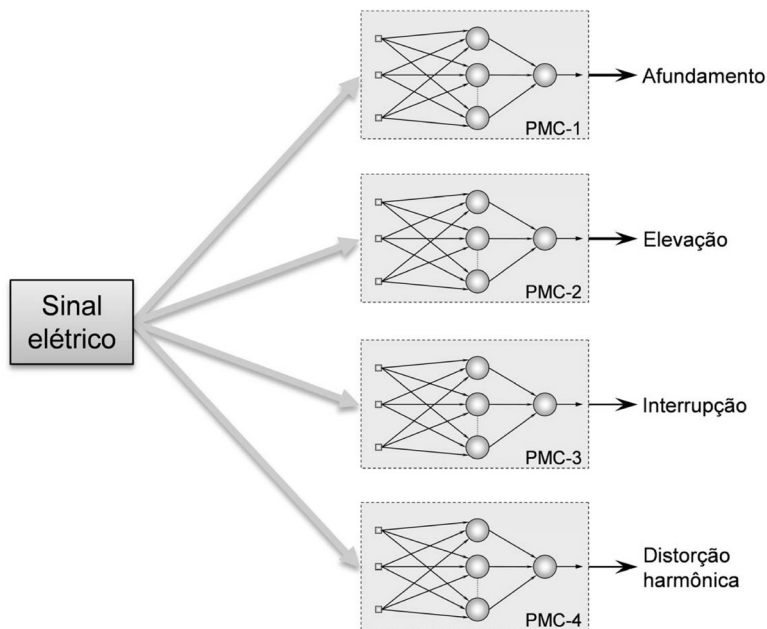


Figura 16.7 – Redes neurais artificiais especialistas para classificação de distúrbios

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.