Universidade Estadual de Campinas
Instituto de Matemática, Estatística e Computação Científica

DISSERTAÇÃO DE MESTRADO

Redes Neurais Morfológicas: Alguns aspectos teóricos e resultados experimentais em problemas de classificação

Autor: Alexandre Monteiro da Silva
Orientador: Prof. Dr. Peter Sussner

Campinas, SP
Junho/2007

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por **Alexandre Monteiro da Silva** e aprovada pela comissão julgadora.

Campinas, 25 de Junho de 2007

Prof. Dr. Peter Sussner

Banca Examinadora

1. Alvaro Rodolfo De Pierro, Dr. ..... DMA/IMECC/Unicamp
2. Clodoaldo Aparecido de Moraes Lima, Dr. .. Univ. Presb. Mackenzie
3. Rosangela Ballini, Dra. ..... IE/Unicamp
4. Peter Sussner, Dr. ..... DMA/IMECC/Unicamp

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica, UNICAMP, como requisito parcial para obtenção do Título de Mestre em Matemática Aplicada.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Júlia Milani Rodrigues

Silva, Alexandre Monteiro da
Si38r Redes neurais morfológicas: alguns aspectos teóricos e resultados experimentais em problemas de classificação / Alexandre Monteiro da Silva -- Campinas, [S.P. :s.n.], 2007.

Orientador : Peter Sussner
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Redes neurais (Computação). 2. Morfologia matemática. 3. Reconhecimento de padrões. I. Sussner, Peter. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Título em inglês: Morphological neural networks: some theoretical aspects and experimental results on classification problems.

Palavras-chave em inglês (Keywords): 1. Neural networks (Computer science). 2. Mathematical morphology. 3. Pattern recognition.

Área de concentração: Matemática Aplicada

Titulação: Mestre em Matemática Aplicada

Banca examinadora: Prof. Dr. Alvaro Rodolfo De Pierro (IMECC-UNICAMP)
Prof. Dr. Clodoaldo Aparecido de Moraes Lima(Univ.Presb. Mackenzie)
Profa. Dra. Rosangela Ballini (IE-UNICAMP)
Prof. Dr. Peter Sussner (IMECC-UNICAMP)

Data da defesa: 25-06-2007

Programa de pós-graduação: Mestrado em Matemática Aplicada

Dissertação de Mestrado defendida em 25 de junho de 2007 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.

Prof. (a). Dr (a). PETER SUSSNER

Prof. (a). Dr (a). CLODOALDO APARECIDO DE MORAES LIMA

Prof. (a). Dr (a). ROSANGELA BALLINI

Resumo

A teoria de redes neurais morfológicas e suas aplicações têm experimentado um crescimento contínuo e crescente nos últimos anos. Neste contexto, calcular o próximo estado de um neurônio, ou de uma camada, envolve uma das operações elementares da morfologia matemática. Nesta dissertação, forneceremos a caracterização de alguns modelos de redes neurais morfológicas, bem fundamentados pela teoria de morfologia matemática em reticulados completos, e também apresentaremos uma comparação do desempenho dos modelos em problemas de classificação.

Palavras-chave: Redes Neurais, Morfologia Matemática, Classificação de Padrões.

Abstract

The theory of morphological neural networks and its applications have experiencied a steady and consistent growth in the last few years. In this setting, computing the next state of a neuron or performing the next layer computation involves one of the elementary operations of mathematical morphology. In this dissertation, we will provide a characterization of several morphological neural networks, well conduct by the theory of mathematical morphology over complete lattices, and we will also present a comparison of the performance of the models over classification problems

Keywords: Neural Networks, Mathematical Morphology, Pattern Classification.

Agradecimentos

Agradeço primeiramente a Deus por ter me dado a oportunidade de me tornar um aluno de pós graduação na Unicamp. Agradeço aos meus pais e minha família pelo apoio incondicional e pela credibilidade dada a mim. Agradeço a todos os professores que contribuíram na minha formação acadêmica ao longo dos meus anos de estudo. Agradeço ao meu orientador de mestrado, o professor Dr. Peter Sussner, pela paciência, por ter depositado sua confiança em mim e por ter me direcionado ao longo de todo o trabalho desenvolvido. Agradeço a todos os meus amigos, tanto aqueles que estiveram de longe, mas que certamente estavam presentes a todo momento no meu coração, quanto aqueles que estiveram de perto acompanhando as minhas jornadas de trabalho. Agradeço também ao saxofonista e compositor de jazz norte americano John Coltrane que através de suas belíssimas músicas providenciou um ambiente totalmente favorável para a minha pesquisa Finalmente, agradeço à Capes pelo suporte financeiro. Espero que este seja o começo de uma vida acadêmica bem sucedida.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nesta dissertação fornecemos um resumo geral a respeito da caracterização de modelos de redes neurais morfológicas [13, 40, 46, 47] por meio da teoria de morfologia matemática em reticulados completos [3, 24], e aplicamos os modelos de redes neurais estudados e implementados em três problemas de classificação: o problema sintético, disponível em [45], e os problemas de diagnóstico de diabetes em uma população de índios Pima e de segmentação de imagens, ambos disponíveis em [36]. Apresentaremos no texto as teorias relevantes e os conceitos que substanciam as redes neurais morfológicas, bem como conceitos fundamentais de redes neurais [8, 21, 22, 45]. De fato, a nossa motivação para o desenvolvimento desta dissertação foi apresentar uma fundamentação teórica das redes neurais morfológicas e uma comparação dos diferentes modelos de redes neurais morfológicas com a rede neural MLP em problemas de classificação. Compararemos os diferentes modelos morfológicos estudados com respeito aos erros de classificação e ao esforço computacional nos problemas de classificação considerados.

Com respeito a organização da dissertação, consideramos inicialmente no segundo capítulo, uma introdução à teoria de morfologia matemática [55, 56], ressaltando que esta teoria pode ser muito bem conduzida por meio da teoria de reticulados [24]. Fornecemos a base da teoria de reticulados para a morfologia matemática, tratamos os casos: binário e em tons de cinza [66]. No capítulo 3 apresentamos a relação da morfologia matemática com outras teorias matemáticas relevantes, dando destaque especial para a álgebra minimax [11, 12]. O capítulo 4 tem a finalidade de introduzir conceitos básicos de redes neurais artificiais tradicionais tais como arquitetura, aprendizagem, treinamento e generalização.

No capítulo 5 introduzimos o conceito de neurônio morfológico e de rede neural morfológica [46, 47, 49]. Apresentamos os modelos morfológicos estudados: perceptron morfológico (PM) [60], perceptron morfológico com dentritos (PMD) [50], rede fuzzy lattice neural network (FLNN) [40], rede neural linear/posto/morfológica híbrida (MRL) [38] e rede neural morfológica modular (MMNN) [14]. Definimos as unidades de processamento que dizem respeito a cada

um dos modelos, bem como as arquiteturas e algoritmos de treinamento. Devemos ressaltar que contribuímos com duas modificações, que são apontadas na seção 5.2, no algoritmo de treinamento do perceptron morfológico. No capítulo 6 apresentamos os problemas de classificação que aplicamos: o problema sintético de Ripley [45], o problema de diabetes nos Indios Pima [36] e o problema de segmentação de imagens retirado de [36]. Discutimos a forma como os conjuntos de dados foram divididos, o modo de treinamento e técnicas para tratar da generalização. Finalmente no capítulo 7 apresentamos uma conclusão geral da dissertação que visa discutir certas nuances dos modelos de redes neurais morfológicas apresentados, resultados experimentais obtidos e perspectivas futuras.

Capítulo 2

Morfologia matemática

A morfologia matemática tem alcançado o status de método poderoso no que se refere ao processamento não linear e análise de imagens que além de ser aplicada à várias áreas como minerologia, visão robótica, microscopia, processamento de imagens médicas, metalurgia e leitura automática de caracteres, tem também se tornado uma teoria matemática sólida baseada em conceitos de álgebra, topologia e geometria integral.

O desenvolvimento inicial desta teoria deu-se inicialmente em 1964 pela contribuição de George Matheron [32] e Jean Serra [56]. Matheron redescobriu as operações de adição e subtração de conjuntos definidas por H. Minkowski [34] em 1903. Estas operações foram estudadas com detalhes em um trabalho desenvolvido por H. Hadwiger [20]. A abordagem de Hadwiger baseada na geometria integral tem tentado construir modelos de conjuntos e descrevê-los com números apropriados.

Originalmente, a morfologia matemática foi desenvolvida para imagens binárias que podem ser representadas matematicamente como conjuntos. Mas desde o começo do desenvolvimento da teoria de morfologia matemática, sentia-se a necessidade de uma teoria mais geral, incluindo outros espaços de objetos tais como subconjuntos fechados de um espaço topológico, bem como espaços de funções para a descrição das imagens em tons de cinza. Serra foi o primeiro a observar que a teoria de morfologia matemática requeria essencialmente que o espaço onde as imagens são definidas fosse um reticulado completo.

2.1 Base da teoria de reticulado para a Morfologia Matemática

O conjunto sobre o qual se aplicam as operações morfológicas será sempre um *conjunto parcialmente ordenado*, ou seja, tal conjunto possui uma relação de ordem parcial $\leq$ definida sobre

ele. Uma relação $\leq$ em um conjunto X é dita ser uma relação de *ordem parcial* se, e somente se, $\leq$ satisfaz as seguintes propriedades:

1) Reflexividade: $x \leq x$, $\forall x \in X$;

2) Anti-Simetria: $x \leq y$, $y \leq x \Rightarrow x = y$ $\forall x, y \in X$;

3) Transitividade: $x \leq y$, $y \leq z \Rightarrow x \leq z$ $\forall x, y, z \in X$;

Seja X um conjunto parcialmente ordenado e seja $Y \subseteq X$. Dizemos que um elemento $l \in X$ é um *limite inferior* de Y se $l \leqslant y$, $\forall y \in Y$. É interessante notar que l, nesta definição, não precisa necessariamente pertencer a Y . De modo semelhante podemos definir a noção de *limite superior*. Dizemos que $l_0 \in X$ é o *ínfimo* de Y se l_0 satisfaz as seguintes propriedades:

(i) l_0 representa um *limite inferior* de y

(ii) $l \leqslant l_0$ para todos os limites inferiores l de Y.

Similarmente podemos definir a noção de *supremo*. Denotamos o ínfimo de Y por $\bigwedge Y$, e o supremo de Y por $\bigvee Y$. Escrevemos $x \wedge y$ em vez de $\bigwedge\{x, y\}$ e $x \vee y$ em vez de $\bigvee\{x, y\}$. Um *conjunto parcialmente ordenado* X é chamado um *reticulado* se, e somente se, existe um *ínfimo* e um *supremo* em X para todo subconjunto *finito* de X. Um *reticulado* é chamado *completo* se todo subconjunto de X tem um *ínfimo* e um *supremo* em X. Daqui em diante, denotamos um *reticulado completo* pela letra $\mathbb{L}$.

Suponha que X e Y são *reticulados*. Uma função $f : X \to Y$ que satisfaz as seguintes equações para todo $x \in X$ e para todo $y \in Y$ recebe o nome de *homomorfismo de reticulados*.

$$f(x \vee y) = f(x) \vee f(y), \tag{2.1}$$

$$f(x \wedge y) = f(x) \wedge f(y). \tag{2.2}$$

Um homomorfismo bijetivo de reticulados é dito um *isomorfismo de reticulados*.

A morfologia matemática binária será aplicada sobre reticulados completos $\wp(X)$, onde X denota o espaço euclideano $\mathbb{R}^d$ ou o espaço $\mathbb{Z}^d$. As operações elementares da morfologia matemática são: *erosão*,*dilatação*,*anti-erosão* e *anti-dilatação*. Sejam $\varepsilon, \delta, \bar{\varepsilon}, \bar{\delta}$ operadores do

reticulado completo $\mathbb{L}$ para o reticulado completo $\mathbb{M}$, e seja $Y \subseteq \mathbb{L}$.

$$\varepsilon \quad \text{é chamada erosão} \Longleftrightarrow \varepsilon(\bigwedge Y) = \bigwedge_{y \in Y} \varepsilon(y); \tag{2.3}$$

$$\delta \quad \text{é chamada dilatação} \Longleftrightarrow \delta(\bigvee Y) = \bigvee_{y \in Y} \delta(y); \tag{2.4}$$

$$\bar{\varepsilon} \quad \text{é chamada anti - erosão} \Longleftrightarrow \bar{\varepsilon}(\bigwedge Y) = \bigvee_{y \in Y} \bar{\varepsilon}(y); \tag{2.5}$$

$$\bar{\delta} \quad \text{é chamada anti-dilatação} \Longleftrightarrow \bar{\delta}(\bigvee Y) = \bigwedge_{y \in Y} \bar{\delta}(y). \tag{2.6}$$

Os operadores *dilatação* e *erosão* são frequentemente associados em termos de uma relação de dualidade. Autores como Deng [15], e Heijimans [24] denominam tal relação de *adjunção*. Dados dois operadores arbitrários $\delta, \varepsilon : \mathbb{L} \to \mathbb{L}$. Dizemos que (ε, δ) é uma adjunção em $(\mathbb{L}, \leq)$ se

$$\delta(a) \leq b \Leftrightarrow a \leq \varepsilon(b), \forall a, b \in \mathbb{L}. \tag{2.7}$$

A proposição que se segue mostra essencialmente que a adjunção constitue uma dualidade entre erosões e dilatações.

Proposição 1 *Seja $\mathbb{L}$ um reticulado completo e dois operadores $\delta, \varepsilon : \mathbb{L} \times \mathbb{L} \to \mathbb{L}$*

1. *Se (ε, δ) é uma adjunção, então δ é uma dilatação e ε é uma erosão.*

2. *Para toda dilatação δ, existe uma única erosão ε, tal que (ε, δ) corresponde a uma adjunção. A erosão adjunta é dada por*

$$\varepsilon(b) = \bigvee \{a \in \mathbb{L} : \delta(a) \leqslant b\}, \tag{2.8}$$

   para todo $b \in \mathbb{L}$

3. *Para toda erosão ε, existe uma única dilatação δ tal que (ε, δ) é uma adjunção. A dilatação adjunta é dada por*

$$\delta(a) = \bigwedge \{b \in \mathbb{L} : \varepsilon(b) \leqslant a\}, \tag{2.9}$$

   para todo $a \in \mathbb{L}$

2.2 Morfologia Matemática binária

O processamento morfológico é baseado na teoria de conjuntos. Os operadores básicos são a *erosão* e a *dilatação* e a partir deles todos os outros operadores morfológicos são definidos por meio de operações de união, intersecção e subtração de conjuntos.

Uma *imagem binária* $\mathbf{A}$ é um subconjunto de um espaço topológico $\mathbf{X} = \mathbb{Z}^n$ ou $\mathbf{X} = \mathbb{R}^n$, em particular o espaço euclideano n dimensional, e para o caso de processamento de imagens considera-se $n = 2$. Para as implementações digitais considera-se uma imagem discreta como um subconjunto do grid cartesiano $2D$.

O prossessamento morfológico de imagens binárias, em um certo sentido geométrico, baseia-se em analisar uma imagem com um *elemento de estruturação*. Um *elemento de estruturação* é uma imagem em um espaço métrico qualquer, discreto ou contínuo, que é transladada sobre este mesmo espaço métrico. No processamento morfológico quantifica-se a maneira como o elemento de estruturação se encaixa (ou nao se encaixa) dentro dos limites da imagem. Marcando as localizações em que o elemento de estruturação se encaixa dentro dos limites da imagem nós obtemos uma informação estrutural com respeito a imagem e esta informação depende tanto da forma, quanto do tamanho do elemento de estruturação e a natureza da informação é dependente da escolha do elemento de estruturação

Erosão

A caracterização do encaixe de um elemento de estruturação $\mathbf{S}$ em uma imagem $\mathbf{A}$ depende de uma operação básica no espaço euclideano, no caso a *translação* de um conjunto $\mathbf{A}$ por um ponto $\mathbf{x}$, que denotamos por

$$\mathbf{A_X} = \{\mathbf{a} + \mathbf{x}; \mathbf{a} \in \mathbf{A}\}.$$

Uma operação fundamental na morfologia matemática binária é a *erosão*. As definições para a *erosão* são baseadas na operação de *subtração de Minkowski* [34] de geometria integral. Seja $\mathbf{S} \subseteq \mathbf{X}$ e $\mathbf{A} \subseteq \mathbf{X}$, a subtração de Minkowski é definida segundo a equação:

$$\mathbf{A} \diagup \mathbf{S} = \bigcap_{\mathbf{s} \in \check{\mathbf{S}}} \mathbf{A_s}. \tag{2.10}$$

onde $\check{\mathbf{S}}$ é a reflexão de $\mathbf{S}$ com relação a origem, ou seja, $\check{\mathbf{S}} = \{-\mathbf{s} \mid \mathbf{s} \in \mathbf{S}\}$. Sejam $\mathbf{A}$ e $\mathbf{S}$ conjuntos de pontos que representam objetos das imagens binárias $\mathbf{a}$ e $\mathbf{s}$, respectivamente, onde $\mathbf{A} \subset \mathbf{X}$ e $\mathbf{S} \subset \mathbf{X}$, sendo $\mathbf{S}$ o *elemento de estruturação*. A *erosão* de uma imagem $\mathbf{A}$ por

um elemento de estruturação $\mathbf{S}$, denotada por $\mathbf{A} \boxminus \mathbf{S}$, é dada por:

$$\mathbf{A} \boxminus \mathbf{S} = \mathbf{A} \diagup \mathbf{S} = \bigcap_{\mathbf{s} \in \check{\mathbf{S}}} \mathbf{A}_{\mathbf{s}}. \tag{2.11}$$

Uma outra formulação equivalente da *erosão* de um imagem $\mathbf{A}$ por um elemento de estruturação $\mathbf{S}$ é a seguinte

$$\mathbf{A} \boxminus \mathbf{S} = \{\mathbf{x} : \mathbf{S}_{\mathbf{x}} \subset \mathbf{A}\}. \tag{2.12}$$

a qual consiste de todos os pontos para os quais a *translação* de $\mathbf{S}$ por $\mathbf{x}$ encaixa-se completamente dentro dos limites de $\mathbf{A}$.

Se a origem é considerada dentro dos limites do elemento de estruturação, então a *erosão* tem o efeito de encolhimento da imagem.

A Figura 2.1 mostra a erosão de uma imagem binária $\mathbf{A}$ por um elemento de estruturação $\mathbf{S}$.

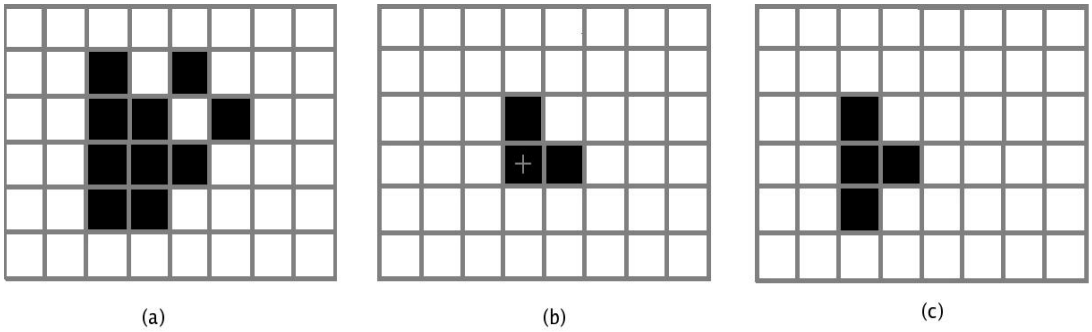

Fig. 2.1: (a) Imagem $\mathbf{A}$, (b) Elemento de estruturação $\mathbf{S}$, (c) Imagem Erodida

Dilatação

A segunda operação fundamental da morfologia matemática binária é a *dilatação*. A *dilatação* corresponde a operação dual da *erosão* no sentido em que é definida via erosão por operação de complementaridade de conjuntos. Em termos matemáticos corresponde à:

$$\mathbf{A} \boxplus \mathbf{S} = (\mathbf{A}^{\mathrm{C}} \boxminus \check{\mathbf{S}})^{\mathrm{C}}.$$

A operação de dilatação é baseada na *soma de Minkowski* que é dada pela equação

$$\mathbf{A} \times \mathbf{S} = \bigcup_{\mathbf{s} \in \mathbf{S}} \mathbf{A}_{\mathbf{s}}. \tag{2.13}$$

A dilatação de uma imagem $\mathbf{A}$ por um elemento de estruturação $\mathbf{S}$ pode ser definida seguindo a formulação de Sternberg [57], denotada por $\mathbf{A} \boxplus \mathbf{S}$, sendo dada por:

$$\mathbf{A} \boxplus \mathbf{S} = \mathbf{A} \times \mathbf{S} = \bigcup_{\mathbf{s} \in \mathbf{S}} \mathbf{A}_{\mathbf{s}}. \tag{2.14}$$

Ou pode-se ainda considerar a formulação de Serra [56]:

$$\mathbf{A} \oplus \mathbf{S} = \mathbf{A} \times \mathbf{S} = \bigcup_{\mathbf{s} \in \check{\mathbf{S}}} \mathbf{A}_{\mathbf{s}}. \tag{2.15}$$

É importante ressaltar uma outra importante formulação equivalente de *dilatação* segundo Sternberg:

$$\mathbf{A} \boxplus \mathbf{S} = \{\mathbf{x} : \check{\mathbf{S}}_{\mathbf{x}} \cap \mathbf{A} \neq \emptyset\}. \tag{2.16}$$

Segundo esta formulação, dilatar a imagem $\mathbf{A}$ por $\mathbf{S}$ resulta no conjunto dado pelos pontos onde o elemento de estruturação refletido $\check{\mathbf{S}}$ transladado por $\mathbf{x}$ e intersectado por $\mathbf{A}$ é diferente de vazio. Neste relatório consideraremos a formulação de Sternberg. A Figura 2.2 mostra a dilatação de uma imagem binária $\mathbf{A}$ por um elemento de estruturação $\mathbf{S}$.

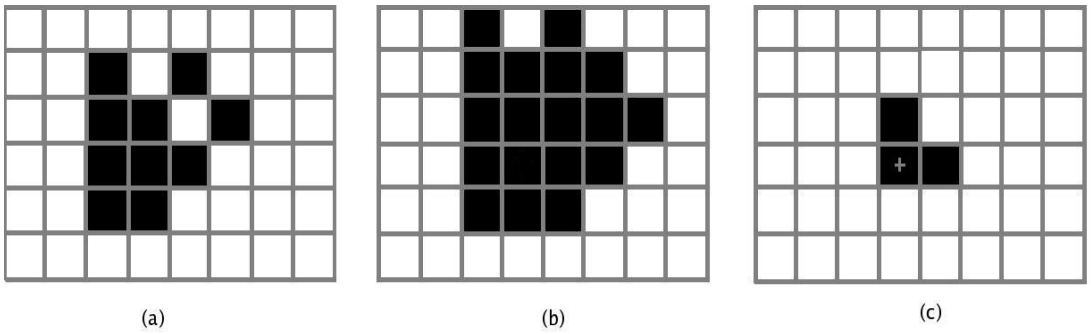

Fig. 2.2: (a) Imagem $\mathbf{A}$, (b) Imagem dilatada por $\mathbf{S}$, (c) Elemento de estruturação $\mathbf{S}$

2.3 Morfologia matemática em tons de cinza

Na morfologia matemática em tons de cinza, uma imagem em tons de cinza $\mathbf{a}$ é uma função de um conjunto $\mathbf{X}$, onde $\mathbf{X} = \mathbb{R}^n$ ou $\mathbf{X} = \mathbb{Z}^n$, para um conjunto de valores $\mathbb{G}$, onde $\mathbb{G}$ pode representar $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ ou $\bar{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\}$. Um *elemento de estruturação* $\mathbf{s}$ é simplesmente uma imagem. Denotamos $\mathbb{G}^{\mathbf{X}}$ o conjunto de todas as imagens que são definidas em $\mathbf{X}$ e assumem valores em $\mathbb{G}$.

Em morfologia matemática, toda imagem em tons de cinza é considerada como uma superfície topográfica, associando cada pixel com uma elevação proporcional à sua intensidade. Esta representação morfológica nos permite aplicar transformações às imagens em tons de cinza. Os operadores morfológicos agirão sobre funções definidas em um espaço $\mathbf{X}$ de dimensão n, onde para o processamento de imagens utiliza-se $n = 1$ ou $n = 2$.

O correspondente à translação, no caso binário, para o caso em tons de cinza é obtido da seguinte maneira: seja $\mathbf{a}$ uma imagem com domínio de definição $\mathbf{X}$ então $\mathbf{a} : \mathbf{X} \to \mathbb{R}$. A translação de $\mathbf{a}$ para a direita por $\mathbf{x}$ é dada por:

$$\mathbf{a_x}(\mathbf{y}) = \mathbf{a}(\mathbf{y} - \mathbf{x});$$

Os operadores morfológicos, na prática, são usados para extrair estruturas relevantes de uma imagem $\mathbf{a}$ considerada. Isto é alcançado investigando a imagem $\mathbf{a}$ com uma outra imagem (ou conjunto), de forma conhecida, chamada de *elemento de estruturação*. Nesta dissertação denotaremos a *erosão* de uma imagem $\mathbf{a}$ por um elemento de estruturação $\mathbf{s}$ por $E(\mathbf{a}, \mathbf{s})$. Similarmente a *dilatação* de $\mathbf{a}$ por $\mathbf{s}$ é denotada por $D(\mathbf{a}, \mathbf{s})$. Denotamos a reflexão de $\mathbf{s}$ em torno da origem por $\check{\mathbf{s}} \in \mathbb{G}^{\mathbf{X}}$ onde

$$\check{\mathbf{s}}(\mathrm{x}) = \mathbf{s}(-\mathrm{x}), \forall \mathrm{x} \in \mathrm{X}. \tag{2.17}$$

Uma erosão ou dilatação particular é especificada por um sub-índice, por exemplo $E_{\mathcal{U}}(\mathbf{a}, \mathbf{s})$ denota a erosão da umbra de $\mathbf{a}$ por $\mathbf{s}$.

Serra e Sternberg desenvolveram abordagens para extender a morfologia matemática binária para a morfologia matemática em tons de cinza, nos anos de 1980. Estas abordagens são chamadas abordagens da umbra (desenvolvida por Sternberg) [57] e threshold (segundo Serra) [56].

Abordagem Threshold

Nesta abordagem consideramos as classes de elementos de estruturação chatos. Mas esta classe é identica ao conjunto de subconjuntos de retas. Consequentemente, nós podemos considerar erosão ou dilatação por um elemento de estruturação chato como erosão ou dilatação de uma imagem por um conjunto.

A formulação Threshold de Serra para as definições de dilatação e erosão é a seguinte, dada uma imagem em tom de cinza $\mathbf{a} \in \mathbb{G}^{\mathbf{X}}$ e um elemento de estruturação $\mathbf{S} \subseteq \mathbf{X}$. Nós definimos a *T-dilatação* $D_T(\mathbf{a}, \mathbf{S})$ e a *T-erosão* $E_T(\mathbf{a}, \mathbf{S})$ como segue:

$$E_T(\mathbf{a}, \mathbf{S})(\mathrm{x}) = \bigwedge_{\mathbf{y} \in \mathrm{S}_{\mathbf{X}}} \mathbf{a}(\mathbf{y}). \tag{2.18}$$

$$D_T(\mathbf{a}, \mathbf{S})(\mathrm{x}) = \bigvee_{\mathbf{y} \in \check{\mathrm{S}}_{\mathbf{X}}} \mathbf{a}(\mathbf{y}). \tag{2.19}$$

onde $\mathbf{S_x}$ é a translação de $\mathbf{S}$ por $\mathbf{x} \in \mathbf{X}$, dada por $\mathbf{S_x} = \{\mathbf{s} + \mathbf{x} : \mathbf{s} \in \mathbf{S}\}$.

Na Figura 2.3, a imagem $\mathbf{a} : \mathbb{R} \to \bar{\mathbb{R}}$, representada com traço contínuo, é definida de um modo tal que $\forall\ \mathrm{x} \in [-6, 9.6]$, temos que $\mathbf{a}$ assume valores reais e $\forall\ \mathrm{x} \notin [-6, 9.6]$, $\mathbf{a}(\mathrm{x}) = -\infty$.

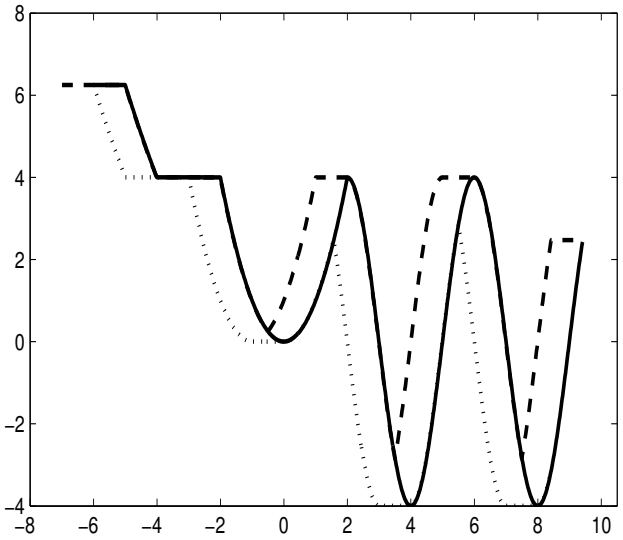

Fig. 2.3: A imagem em tom de cinza $\mathbf{a} : \mathbb{R} \to \bar{\mathbb{R}}$ é representada com traço contínuo; $E_T(\mathbf{a}, \mathbf{S})$ é representado em pontilhado; $D_T(\mathbf{a}, \mathbf{S})$ é representado em tracejado, onde $\mathbf{S} = [0, 1]$.

Abordagem da Umbra

Sternberg extendeu a morfologia matemática binária para a análise de imagens em tons de cinza utilizando a noção de *umbra*. Seja uma imagem $\mathbf{a} : \mathbf{X} \longrightarrow \mathbb{G}$, com domínio de definição $\mathbf{X} = \mathbb{R}^n$. A *umbra* de $\mathbf{a}$, denotada por $\mathcal{U}(\mathbf{a})$, é definida por:

$$\mathcal{U}(\mathbf{a}) = \{(x_1, x_2, \ldots, x_n, x_{n+1}) \in \mathbf{X} \times \mathbb{G}; (x_1, ...x_n) \in \mathbf{X}, \quad \text{e} \quad x_{n+1} \leq \mathbf{a}(x_1, x_2, \ldots, x_n)\}. \tag{2.20}$$

Nós definimos a $\mathcal{U}$-*erosão* e a $\mathcal{U}$-*dilatação* de uma imagem $\mathbf{a} : \mathbf{X} \longrightarrow \mathbb{G}$ por um elemento de estruturação $\mathbf{s} : \mathbf{X} \longrightarrow \mathbb{G}$ como se segue:

$$E_{\mathcal{U}}(\mathbf{a}, \mathbf{s})(\mathrm{x}) = \bigwedge_{\mathbf{y} \in \mathbf{X}} (\mathbf{a}(\mathrm{y}) +' \check{\mathbf{s}}_{\mathbf{y}}(\mathrm{x})). \tag{2.21}$$

$$D_{\mathcal{U}}(\mathbf{a}, \mathbf{s})(\mathbf{x}) = \bigvee_{\mathbf{y} \in \mathbf{X}} (\mathbf{a}(\mathbf{y}) + \mathbf{s}_{\mathbf{y}}(\mathbf{x})). \tag{2.22}$$

Nas equações (2.21) e (2.22), as operações " $+$ " e " $+'$ " são definidas de maneira usual, exceto com respeito a soma de ∞ e $-\infty$, onde estas equações se comportam do seguinte modo:

$$\infty + (-\infty) = (-\infty) + \infty = -\infty. \tag{2.23}$$

$$\infty +' (-\infty) = (-\infty) +' \infty = \infty. \tag{2.24}$$

No exemplo da Figura (2.4(a)) a imagem $\mathbf{a} : \mathbb{R} \to \bar{\mathbb{R}}$, representada com traço contínuo, é definida de um modo tal que $\forall$ x $\in [-6, 9.6]$, temos que $\mathbf{a}$ assume valores reais e $\forall$ x $\notin [-6, 9.6]$, $\mathbf{a}(\mathrm{x}) = -\infty$. Da mesma forma definimos o elemento de estruturação $\mathbf{s}$, Figura 2.4(b), de maneira tal que $\forall$ x$\in [-1, 1]$, $\mathbf{s}$ assume valores reais e $\forall x \notin [-1, 1]$, $\mathbf{s}(\mathrm{x}) = -\infty$.

A idéia por trás da abordagem da umbra consistiu em uma maneira natural de extender a morfologia matemática binária para a morfologia matemática em tons de cinza. Vamos denotar $\mathcal{U}_{\mathbf{X} \times \mathbb{G}}$ o conjunto de todas umbras contidas em $\mathbf{X} \times \mathbb{G}$. Usando a definição de $\mathcal{U}(\mathbf{a})$, o homomorfismo de reticulados $\mathcal{U} : \mathbb{G}^{\mathbf{X}} \longrightarrow \mathcal{U}_{\mathbf{X} \times \mathbb{G}}$ torna-se um isomorfismo de reticulados se $\mathbb{G}$ é discreto. O inverso deste isomorfismo é dado por $\mathcal{T} : \mathcal{U}_{\mathbf{X} \times \mathbb{G}} \longrightarrow \mathbb{G}^{\mathbf{X}}$ onde

$$(\mathcal{T}(U))(x) = \bigvee \{t \in \mathbb{G} : (\mathbf{x}, t) \in U\}, \forall \mathbf{x} \in \mathbf{X}. \tag{2.25}$$

Nós nos referimos a $\mathcal{T}(U)$ como sendo o *topo da umbra* $\mathcal{U}$. No caso em que o conjunto de níveis de cinza $\mathbb{G}$ é contínuo, ou seja $\mathbb{G} = \bar{\mathbb{R}}$, o homomorfismo $\mathcal{U}$ é injetivo mas falha em ser sobrejetivo [66]. Portanto o homomorfismo $\mathcal{T}$, definido na equação (2.25), representa uma

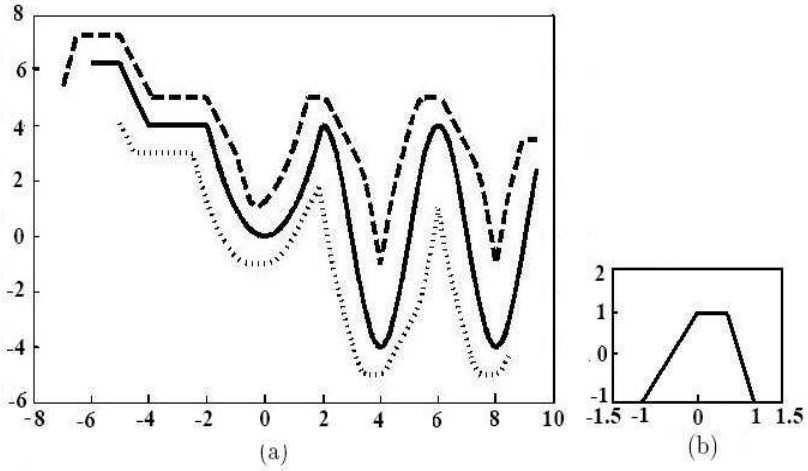

Fig. 2.4: (a) imagem em tom de cinza $\mathbf{a} : \mathbb{R} \to \bar{\mathbb{R}}$ representada com traço contínuo; $E_{\mathcal{U}}(\mathbf{a}, \mathbf{s})$ representada em pontilhado; $D_{\mathcal{U}}(\mathbf{a}, \mathbf{s})$ representada em tracejado; (b) elemento de estruturação $\mathbf{s} : \mathbb{R} \to \bar{\mathbb{R}}$.

inversa à esquerda de $\mathcal{U}$ mas não representa uma inversa à direita de $\mathcal{U}$. Para ambos os casos, tanto discreto como contínuo de imagens em tons de cinza, nós podemos considerar a seguinte formulação da erosão e dilatação da umbra de uma imagem $\mathbf{a}$ por um elemento de estruturação $\mathbf{s}$:

$$D_{\mathcal{U}}(\mathbf{a}, \mathbf{s}) = \mathcal{T}(D_{\mathcal{B}}(\mathcal{U}(\mathbf{a}), \mathcal{U}(\mathbf{s}))), \tag{2.26}$$

$$E_{\mathcal{U}}(\mathbf{a}, \mathbf{s}) = \mathcal{T}(E_{\mathcal{B}}(\mathcal{U}(\mathbf{a}), \mathcal{U}(\mathbf{s}))), \tag{2.27}$$

Sendo que $D_{\mathcal{B}}(\mathcal{U}(\mathbf{a}), \mathcal{U}(\mathbf{s}))$ e $E_{\mathcal{B}}(\mathcal{U}(\mathbf{a}), \mathcal{U}(\mathbf{s}))$ representam, respectivamente, a dilatação e erosão binária de uma imagem $\mathbf{a}$ por um elemento de estruturação $\mathbf{s}$ de acordo com as equações (2.26) e (2.27). Enfatizamos que $V = \mathcal{T}(\mathcal{U}(V))$ para todas umbras V se $\mathbb{G} = \bar{\mathbb{Z}}$. Portanto, podemos obter as seguintes igualdades a partir das equações (2.26) e (2.27).

$$\mathcal{U}(D_{\mathcal{U}}(\mathbf{a}, \mathbf{s})) = D_{\mathcal{B}}(\mathcal{U}(\mathbf{a}), \mathcal{U}(\mathbf{s})), \tag{2.28}$$

$$\mathcal{U}(E_{\mathcal{U}}(\mathbf{a}, \mathbf{s})) = E_{\mathcal{B}}(\mathcal{U}(\mathbf{a}), \mathcal{U}(\mathbf{s})), \tag{2.29}$$

Para o caso contínuo de níveis de cinza, a equação (2.28) não é satisfeita em geral [66].

Podemos estabelecer uma relação entre a abordagem threshold e a abordagem umbra. Seja

um elemento de estruturação binário $\mathbf{S} \subseteq \mathbf{X}$, considere um elemento de estruturação $\mathbf{s} \in \mathbb{G}^{\mathbf{X}}$ definido da seguinte maneira:

$$\mathbf{s}(\mathbf{x}) = \begin{cases} 0, & \text{se} \quad \mathbf{x} \in \mathbf{S} \\ -\infty, & \text{se} \quad \mathbf{x} \notin \mathbf{S} \end{cases}$$

De fato com estas considerações temos as seguintes igualdades:

$$D_{\mathcal{T}}(\mathbf{a}, \mathbf{S}) = D_{\mathcal{U}}(\mathbf{a}, \mathbf{s}), \tag{2.30}$$

$$E_{\mathcal{T}}(\mathbf{a}, \mathbf{S}) = E_{\mathcal{U}}(\mathbf{a}, \mathbf{s}). \tag{2.31}$$

No exemplo da imagem $\mathbf{a} : \mathbb{R} \to \bar{\mathbb{R}}$ da Figura 2.5, considerando $\mathbf{S} = [0, 1]$ podemos utilizar a abordagem das umbras para obter o resultado da *T-erosão* $E_T(\mathbf{a}, \mathbf{S})$ e da *T-dilatação* $D_T(\mathbf{a}, \mathbf{S})$, como pode ser observado nas figuras 2.6 e 2.7 respectivamente.

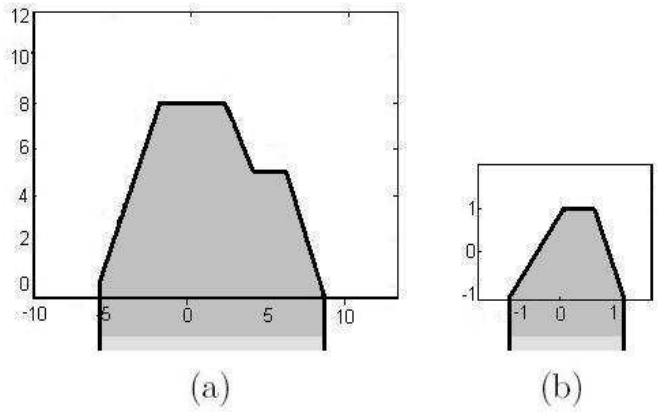

Fig. 2.5: (a)umbra de uma imagem $\mathbf{a} : \mathbb{R} \to \bar{\mathbb{R}}$ em tom de cinza; (b) umbra de um elemento de estruturação $\mathbf{s} : \mathbb{R} \to \bar{\mathbb{R}}$

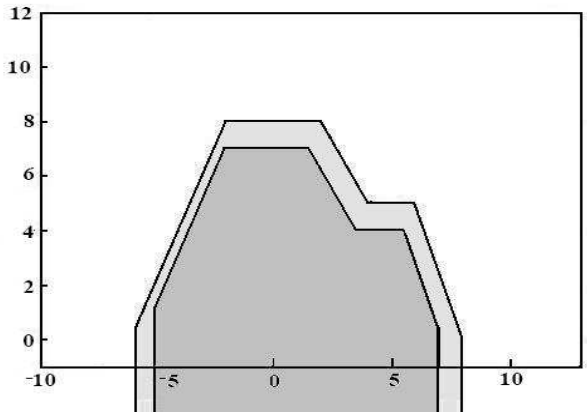

Fig. 2.6: A umbra da imagem $\mathbf{a} : \mathbb{R} \longrightarrow \bar{\mathbb{R}}$ corresponde a região cinza escuro unida com a região cinza claro e a região cinza escuro corresponde a erosão de $\mathcal{U}(\mathbf{a})$ por $\mathcal{U}(\mathbf{s})$.

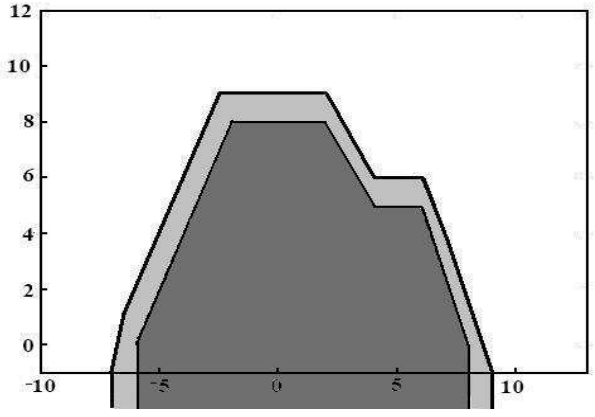

Fig. 2.7: A umbra da imagem $\mathbf{a} : \mathbb{R} \longrightarrow \bar{\mathbb{R}}$ corresponde a região cinza escuro e a região cinza claro unida com a cinza escuro corresponde a dilatação de $\mathcal{U}(\mathbf{a})$ por $\mathcal{U}(\mathbf{s})$

Capítulo 3

Relação da morfologia matemática com outras teorias matemáticas

3.1 Álgebra minimax

A partir de 1950 diferentes pesquisadores descobriram independentemente a estrutura algébrica formada pelo conjunto $\bar{\mathbb{R}}$ munido das operações binárias de máximo, mínimo e adição [7]. A extensão desta estrutura para matrizes foi desenvolvida por Cuninghame-Green [11] no seu livro *Minimax Algebra* (1979), onde são encontrados exemplos de aplicações no campo de pesquisa operacional. A *álgebra minimax* é uma estrutura algébrica de matrizes e vetores que assumem valores em um *bounded lattice ordered group*, ou simplesmente *blog*, e possui características semelhantes às da álgebra linear, tais como a noção de solução de sistemas de equações, independência linear, autovalores, autovetores e matriz inversa. O conjunto $\mathbb{G}$ munido das operações de máximo, mínimo e as adições $+,+'$ constitui um exemplo de *blog*.

As operações " $+$ " e " $+'$ " são definidas de maneira usual em $\mathbb{G}$, exceto quando operam sobre ∞ e $-\infty$, onde estas operações são definidas de acordo com as equações (2.23) e (2.24).

O sistema $(\mathbb{G}, \vee, \wedge, +, +')$ será o *blog* que adotaremos em particular. Existem dois tipos de operações definidas em matrizes tendo valores neste sistema. Sejam $\mathrm{A} = (\mathrm{a}_{ij})$ e $\mathrm{B} = (\mathrm{b}_{ij})$ pertencentes à $\mathbb{G}_{m\times n}$ então o *máximo* $\mathrm{A} \vee \mathrm{B}$ é definido como:

$$\mathrm{A} \vee \mathrm{B} = \mathrm{C}, \quad \text{onde} \quad \mathrm{c}_{ij} = \mathrm{a}_{ij} \vee \mathrm{b}_{ij}. \tag{3.1}$$

Se $\mathbf{A}$ é $m \times p$ e $\mathbf{B}$ é $p \times n$. Definimos o *produto máximo* e o *produto mínimo* da matriz $\mathbf{A}$ pela matriz $\mathbf{B}$ respectivamente por:

$$\mathrm{A} \boxtimes \mathrm{B} = \mathrm{C}, \quad \text{onde} \quad \mathrm{c}_{ij} = \bigvee_{k=1}^{p} (\mathrm{a}_{ik} + \mathrm{b}_{kj}). \tag{3.2}$$

$$\mathrm{A} \boxtimes \mathrm{B} = \mathrm{C}, \quad \text{onde} \quad \mathrm{c}_{ij} = \bigwedge_{k=1}^{p} (\mathrm{a}_{ik} + \mathrm{b}_{kj}). \tag{3.3}$$

Definimos sobre o *blog* $(\mathbb{G}, \vee, \wedge, +, +')$ o automorfismo da conjugação dado por:

$$x^* = \begin{cases} -x, & \text{se} \quad x \in \mathbb{G} \setminus \{-\infty, +\infty\} \\ +\infty, & \text{se} \quad x = -\infty \\ -\infty, & \text{se} \quad x = \infty \end{cases}$$

Outro isomorfismo que vamos considerar diz respeito ao isomorfismo que leva uma matriz $\mathrm{A} \in \mathbb{G}^{m \times n}$ a uma matriz conjugada $\mathrm{A}^* \in \mathbb{G}^{n \times m}$, onde cada entrada a^*_{ij} da matriz A^* é dada por:

$$a^*_{ij} = (a_{ji})^*; \tag{3.4}$$

A bijeção que leva uma matriz A para $(\mathrm{A}^*)^\top = (\mathrm{A}^\top)^*$ representa uma negação. Finalmente definimos uma negação especial $\nu_* : \mathbb{G}^n \longrightarrow \mathbb{G}^n$ tal que :

$$\nu_*(x) = (x^*)^\top, \quad \forall \mathbf{x} \in \mathbb{G}^n. \tag{3.5}$$

As matrizes A e B com entradas em $\mathbb{G}$ satisfazem às seguintes igualdades:

$$(\mathrm{A} \boxbslash \mathrm{B})^* = \mathrm{B}^* \boxtimes \mathrm{A}^* \quad \text{e} \quad (\mathrm{A} \boxtimes \mathrm{B})^* = \mathrm{B}^* \boxtimes \mathrm{A}^*. \tag{3.6}$$

No caso em que B tem tamanho $n \times 1$, ou seja, B se reduz a um vetor $\mathbf{x}$, e A tem tamanho $m \times n$, temos que o *produto máximo* de uma matriz A por um vetor $\mathbf{x}$, visto como um operador, é dado por:

$$\begin{array}{rccc} \delta_A & : & \mathbb{G}^n & \longrightarrow & \mathbb{G}^m \\ & & \mathbf{x} & \longmapsto & \mathrm{A} \boxbslash \mathbf{x} \end{array} \tag{3.7}$$

Este operador representa uma *dilatação* no sentido de Heijimans [24], ou seja, satisfaz a equação (2.4). O *produto mínimo* de uma matriz A por um vetor $\mathbf{x}$, visto como um operador, é dado por:

$$\begin{array}{rccc} \varepsilon_A & : & \mathbb{G}^n & \longrightarrow & \mathbb{G}^m \\ & & \mathbf{x} & \longmapsto & \mathrm{A} \boxtimes \mathbf{x} \end{array} \tag{3.8}$$

Este operador claramente representa uma *erosão* também no sentido de Heijimans [24], ou seja, satisfaz a equação (2.3). Seja $\mathbf{w}$ um vetor fixo de $\mathbb{G}^n$, e $\mathbf{x}^k$ uma coleção arbitrária de elementos de $\mathbb{G}^n$, vamos mostrar que $\bar{\delta}_{\mathbf{w}} = \varepsilon_{\mathbf{w}} \circ \nu_*$ corresponde de fato a uma *anti-dilatação*,

onde $\varepsilon_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^{\top} \boxbslash \mathbf{x}$. Neste sentido, devemos mostrar a seguinte igualdade:

$$\bar{\delta}_{\mathbf{w}}(\bigvee_{k\in I} \mathbf{x}^k) = \bigwedge_{k\in I} \bar{\delta}_{\mathbf{w}}(\mathbf{x}^k). \tag{3.9}$$

Desenvolvendo o primeiro membro da equação (3.9):

$$\bar{\delta}_{\mathbf{w}}(\bigvee_{k\in I} \mathbf{x}^k) \;=\; \mathbf{w}^{\top} \boxbslash \nu_*(\bigvee_{k\in I} \mathbf{x}^k), \tag{3.10}$$

A seguir, desenvolvendo a expressão do segundo membro da equação (3.9), temos que

$$\begin{aligned} \bigwedge_{k\in I} \bar{\delta}_{\mathbf{w}}(\mathbf{x}^k) &= \bigwedge_{k\in I} \{\mathbf{w}^{\top} \boxbslash \nu_*(\mathbf{x}^k)\}, & (3.11)\\ &= \bigwedge_{k\in I} \{(w_1 - x_1^k) \wedge (w_2 - x_2^k) \wedge \ldots \wedge (w_n - x_n^k)\}. & (3.12) \end{aligned}$$

Rearranjando os termos e usando o fato do reticulado completo ser munido da operação de distributividade, temos que:

$$\begin{aligned} \bigwedge_{k\in I} \bar{\delta}_{\mathbf{w}}(\mathbf{x}^k) &= \bigwedge_{j=1} w_j + \bigwedge_{k\in I} -x_j^k, & (3.13)\\ &= \bigwedge_{j=1}^{n} \{w_j - (\bigvee_{k\in I} x_j^k)\}, & (3.14)\\ &= \mathbf{w}^{\top} \boxbslash \nu_*(\bigvee_{k\in I} \mathbf{x}^k), & (3.15)\\ &= \bigvee_{k\in I} \bar{\delta}_{\mathbf{w}}(\mathbf{x}^k). & (3.16) \end{aligned}$$

3.2 Álgebra de imagens

A *álgebra de imagens* [12] é uma teoria preocupada com a transformação e análise de imagens, sobretudo de imagens digitais. No contexto matemático, a *álgebra de imagens* é uma álgebra heterogênea, no sentido de Birkhoff e Lipson (1970) [7], em outras palavras, a *álgebra de imagens* é uma coleção de conjuntos não vazios junto com um conjunto de operandos e operadores. De fato, a álgebra de imagens admite como operandos imagens, conjuntos de pontos, conjunto de valores, os elementos destes conjuntos e *templates*. Um conjunto de pontos, no que diz respeito ao processamento de imagens, corresponde a um espaço topológico que em particular adotamos como sendo $\mathbf{X}$. O *conjunto de valores* é uma álgebra homogênea, isto é, uma álgebra

heterogênea com somente um conjunto de operandos. Adotamos o conjunto de valores como sendo $\mathbb{G}$.

Uma *imagem* é um operando fundamental em *álgebra de imagens*, a qual consiste de uma função definida em um espaço topológico $\mathbf{X}$, assumindo valores em um *conjunto de valores* $\mathbb{G}$. Em geral, representamos uma imagem $\mathbf{a}$ da seguinte forma: $\mathbf{a} : \mathbf{X} \to \mathbb{G}$. O elemento $(\mathbf{x}, \mathbf{a}(\mathbf{x}))$ é chamado de pixel. A primeira coordenada $\mathbf{x}$ de um pixel é chamada ponto de imagem, e a segunda coordenada $\mathbf{a}(\mathbf{x})$ é chamada o valor de pixel de $\mathbf{a}$ em $\mathbf{x}$. O conjunto de valores $\mathbb{G}$ induz operações nas imagens definidas em $\mathbf{X}$ e com valores em $\mathbb{G}$. Definimos as seguintes operações de interesse entre imagens $\mathbf{a}, \mathbf{b} \in \mathbb{G}^{\mathbf{X}}$:

$$\mathbf{a} \vee \mathbf{b} \equiv \mathbf{c} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \vee \mathbf{b}(\mathbf{x}), \forall \mathbf{x} \in \mathbf{X}\}. \tag{3.17}$$

$$\mathbf{a} \wedge \mathbf{b} \equiv \mathbf{c} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \wedge \mathbf{b}(\mathbf{x}), \forall \mathbf{x} \in \mathbf{X}\}. \tag{3.18}$$

Templates

Um *template* é um caso especial do conceito geral de imagem. A noção de um *template*, no contexto da *álgebra de imagens*, unifica e generaliza os conceitos básicos de máscaras, janelas e vizinhanças em uma entidade matemática geral. Mais ainda, *templates* generalizam a noção de elementos de estruturação da morfologia matemática. Seja $\mathbf{X}$ o conjunto de pontos e $\mathbb{G}$ um conjunto de valores. Um template generalizado $\mathbf{t}$ de $\mathbf{Y}$ para $\mathbf{X}$ é uma função $\mathbf{t} : \mathbf{Y} \longrightarrow \mathbb{G}^{\mathbf{X}}$, em outras palavras, para cada $\mathbf{y} \in \mathbf{X}$, $\mathbf{t}(\mathbf{y})$ é uma imagem em $\mathbf{X}$.

Denotamos $\mathbf{t}(\mathbf{y})$ por $\mathbf{t}_{\mathbf{y}}$, por uma conveniência de notação, ou seja,

$$\mathbf{t}_{\mathbf{y}} = \{(\mathbf{x}, \mathbf{t}_{\mathbf{y}}(\mathbf{x}) : \mathbf{x} \in \mathbf{X}, \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \in \mathbb{G}\}. \tag{3.19}$$

O valor de pixel $\mathbf{t}_{\mathbf{y}}(\mathbf{x})$ recebe o nome de peso do *template* no ponto $\mathbf{y}$. O conjunto de todas as templates com valores em $\mathbb{G}$ de $\mathbf{Y}$ para $\mathbf{X}$ é denotado por $(\mathbb{G}^{\mathbf{X}})^{\mathbf{Y}}$. Particularmente, dizemos que uma template é *invariante sob translação* se para todas as triplas $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{X}$ com $(\mathbf{y} + \mathbf{z}) \in \mathbf{X}$ e $(\mathbf{x} + \mathbf{z}) \in \mathbf{X}$, nós tivermos que

$$\mathbf{t}_{\mathbf{y}}(\mathbf{x}) = \mathbf{t}_{\mathbf{y}+\mathbf{z}}(\mathbf{x} + \mathbf{z})$$

Uma template que não é *invariante sob translação* é dita ser *variante*.

Produtos Imagens-Templates

O produto imagem-template nas transformações de imagens que consideraremos, tais que $\mathbf{a} \in \mathbb{G}^{\mathbf{X}}$ e $\mathbf{t} \in (\mathbb{G}^{\mathbf{X}})^{\mathbf{Y}}$, são dados por:

$$\mathbf{a} \boxtimes \mathbf{t} = \{(\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in \mathbf{X}} (\mathbf{a}(\mathbf{x}) + \mathbf{t}_{\mathbf{y}}(\mathbf{x})), \mathbf{y} \in \mathbf{Y}\}, \tag{3.20}$$

$$\mathbf{a} \boxtimes \mathbf{t} = \{(\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x} \in \mathbf{X}} (\mathbf{a}(\mathbf{x}) +' \mathbf{t}_{\mathbf{y}}(\mathbf{x})), \mathbf{y} \in \mathbf{Y}\}, \tag{3.21}$$

$$\mathbf{t} \boxtimes \mathbf{a} = \{(\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in \mathbf{X}} (\mathbf{t}_{\mathbf{x}}(\mathbf{y}) + \mathbf{a}(\mathbf{x})), \mathbf{y} \in \mathbf{Y}\}, \tag{3.22}$$

$$\mathbf{t} \boxtimes \mathbf{a} = \{(\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x} \in \mathbf{X}} (\mathbf{t}_{\mathbf{x}}(\mathbf{y}) +' \mathbf{a}(\mathbf{x})), \mathbf{y} \in \mathbf{Y}\}. \tag{3.23}$$

A morfologia matemática pode ser imersa na álgebra de imagens por meio de um isomorfismo, formulado por Jennifer L. Davidson [12], que faz corresponder os produtos $\boxtimes$,$\boxtimes$, definidos nas equações (3.2) e (3.3), de uma imagem $\mathbf{a}$ por um template $\mathbf{t}$ a uma *dilatação* e *erosão*, respectivamente, de uma imagem por um elemento de estruturação da morfologia matemática. Podemos mostrar que existe uma sub-álgebra da álgebra de imagens que é isomorfa à álgebra minimax. Neste sentido, é possível estabelecer um isomorfismo entre a morfologia matemática e a álgebra minimax [12].

3.3 A imersão da morfologia matemática na álgebra minimax

Nesta seção apresentaremos e discutiremos a relação entre a álgebra minimax e a morfologia matemática. Primeiramente, consideraremos o caso binário. Seja $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\} \subset \mathbf{X}$ e $\mathbf{A}, \mathbf{S} \subset \mathbf{Y}$ tais que $\mathbf{A} \boxplus \mathbf{S} \subset \mathbf{Y}$. Definimos a função $\rho : \mathcal{P}(\mathbf{X}) \to \mathbb{G}^n$ dada por:

$$\rho(\mathbf{A}) = \mathbf{v} = \begin{pmatrix} \mathrm{v}_1 \\ \vdots \\ \mathrm{v}_n \end{pmatrix}, \quad \text{onde} \quad \mathrm{v}_i = \begin{cases} 1, & \text{se} \quad \mathbf{y}_i \in \mathbf{A} \\ 0, & \text{se} \quad \mathbf{y}_i \notin \mathbf{A} \end{cases} \tag{3.24}$$

Esta função mapeia uma imagem $\mathbf{A}$ da morfologia matemática para um vetor $\mathbf{v}$ da álgebra minimax. Uma outra função interessante que definiremos é $\xi : \mathcal{P}(\mathbf{X}) \to \mathbb{G}^{m \times n}$ dada por:

$$\xi(\mathbf{S}) = Q = (\mathrm{q}_{ji}), \quad \text{onde} \quad \mathrm{q}_{ij} = \begin{cases} 0, & \text{se} \quad \mathbf{y}_i \in \check{\mathbf{S}}_{\mathbf{y}_j} \\ -\infty, & \text{se} \quad \mathbf{y}_i \notin \check{\mathbf{S}}_{\mathbf{y}_j} \end{cases} \tag{3.25}$$

Isto é, para cada coluna j fixa da matriz Q, o elemento q_{ij} é igual a zero, se $\mathbf{y}_i \in \check{\mathbf{S}}_{\mathbf{y}_j}$ ou $-\infty$, se $\mathbf{y}_i \notin \check{\mathbf{S}}_{\mathbf{y}_j}$. A partir destas funções obtemos um mapeamento de um elemento de estruturação da morfologia matemática para uma matriz da álgebra minimax e vice versa. Caracterizaremos estas correspondências por meio das equações abaixo:

$$\rho(\mathbf{A} \boxplus \mathbf{S}) = \xi(\mathbf{S}) \boxtimes \rho(\mathbf{A}). \tag{3.26}$$

$$\rho(\mathbf{A} \boxminus \mathbf{S}) = ((\xi(\mathbf{S})))^* \boxtimes \rho(\mathbf{A}). \tag{3.27}$$

Com respeito ao caso tons de cinza, considere $\mathbf{a}, \mathbf{s} \in \mathbb{G}^{\mathbf{X}}$. Suponhamos que $S_{-\infty}(D_{\mathcal{U}}(\mathbf{a}, \mathbf{s})) \subset \mathbf{Y}$, onde consideramos $D_{\mathcal{U}}(\mathbf{a}, \mathbf{s})(\mathbf{y}_j) = \bigvee_{\mathbf{y}_i \in \mathbf{Y}} (\mathbf{a}(\mathbf{y}_i) + \check{\mathbf{s}}_{\mathbf{y}_j}(\mathbf{y}_i))$. Definimos a função $\varrho : \mathbb{G}^{\mathbf{X}} \to \mathbb{G}^n$ dada por:

$$\varrho(\mathbf{a}) = \mathbf{v}, \quad \text{onde} \quad \mathbf{v} = \begin{pmatrix} \mathbf{a}(\mathbf{y}_1) \\ \vdots \\ \mathbf{a}(\mathbf{y}_n) \end{pmatrix}, \quad \forall \mathbf{y}_i \in \mathbf{Y}. \tag{3.28}$$

Definimos $\psi : \mathbb{G}^{\mathbf{X}} \to \mathbb{G}^{n \times n}$ dada por:

$$\psi(\mathbf{s}) = Q = (\mathrm{q}_{ij}), \quad \text{onde} \quad \mathrm{q}_{ij} = \mathbf{s}_{\mathbf{y}_j}(\mathbf{y}_i), \quad \forall \mathbf{y}_i, \mathbf{y}_j \in \mathbf{Y}. \tag{3.29}$$

Isto é, para uma coluna j fixa da matriz Q, temos que o elemento q_{ij} é dado por $\mathbf{s}_{\mathbf{y}_j}(\mathbf{y}_i)$, para todo $\mathbf{y}_i, \mathbf{y}_j \in \mathbf{Y}$. As equações que determinam a correspondência entre a álgebra minimax e a morfologia matemática, para o caso tons de cinza, são dadas por:

$$\varrho(D_{\mathcal{U}}(\mathbf{a}, \mathbf{s})) = \psi(\mathbf{s}) \boxtimes \varrho(\mathbf{a}). \tag{3.30}$$

$$\varrho(E_{\mathcal{U}}(\mathbf{a}, \mathbf{s})) = (\psi(\mathbf{s}))^* \boxtimes \varrho(\mathbf{a}). \tag{3.31}$$

Capítulo 4

Conceitos básicos de redes neurais tradicionais

4.1 Breve introdução sobre Redes Neurais Artificiais

O cérebro de um ser humano possui bilhões de células nervosas as quais chamamos de neurônios. Dizemos que o cérebro é uma rede neural biológica. Um modelo matemático que descreve um neurônio recebe o nome de *neurônio artificial* [21]. Uma rede neural artificial, ou simplesmente uma rede, pode ser definida como uma estrutura de processamento formada por neurônios artificiais, onde estes, por sua vez, são conectados por ligações as quais damos o nome de *pesos sinápticos* [30], que são tipicamente adaptados durante o processo. De fato, o estudo de redes neurais artificiais se desenvolveu inicialmente na tentativa de se entender determinadas funções que cérebro biológico desempenha.

De um modo geral, os modelos de redes neurais artificiais são aplicados em áreas tais como reconhecimento de fala e imagens, data mining, memórias associativas, classificação de padrões e outras. Dentre os benefícios potenciais de redes neurais artificiais, destacamos a habilidade da rede neural aprender a partir de seu ambiente e de melhorar o seu desempenho com a evolução do tempo.

Os modelos de redes neurais artificiais são especificados pela sua topologia, função de agregação dos neurônios e regras de treinamento e aprendizado [22]. Ao longo deste capítulo caracterizaremos estes conceitos.

4.2 Contexto histórico

A introdução de modelos matemáticos para a simulação do comportamento neural ocorreu por volta de 1940 quando o biólogo Warren McCulloch e o matemático Walter Pitts apresentaram o modelo de um neurônio biológico [33] que satisfazia uma lei tudo ou nada, e a partir de certas conexões sinápticas, neurônios operando paralelamente poderiam computar funções booleanas. Em 1949, Donald Hebb publicou o livro *The Organization of Behavior* [23], onde foi introduzido uma primeira formulação de uma regra de aprendizagem que explica como uma rede de neurônios aprende. Alguns anos mais tarde surgiram trabalhos com esse tema, como o de Rosenblatt [52] (1958) que apresentou uma nova abordagem para o problema de reconhecimento de padrões. Nessa mesma época, Widrow e Hoff [72] introduziram o algoritmo do mínimo quadrado médio (LMS-Least Mean-square) para a formulação do *Adaline* (*adaptative linear element*).

O *perceptron* e o *Adaline* [35] apresentavam certas limitações que foram superadas com o desenvolvimento das redes neurais de múltiplas camadas, mas Rosenblatt e Widrow não conseguiam extender seus algoritmos de aprendizagem para estas redes mais complexas. Estas dificuldades e a carência de recursos tecnológicos provocaram um certo declíneo no número de publicações científicas em redes neurais. Mas, na década de 80 esse panorama científico é mudado e ocorre um crescimento considerável no desenvolvimento da pesquisa em redes neurais. Novos conceitos foram introduzidos, tais como mecânica estatística, introduzido por Hopfield [25], possibilitando um maior entendimento de certas redes neurais artificiais recorrentes. Um outro ponto chave para o desenvolvimento das redes neurais artificiais na década de 80 foi o *algoritmo backpropagation* [9, 71] utilizado para o treinamento de perceptron de múltiplas camadas.

4.3 Neurônio Artificial

O modelo simples de um neurônio artificial consiste em um elemento de processamento que recebe um ou mais sinais de entrada e apresenta um sinal de saída. Representamos os atributos de um padrão de entrada $\mathbf{x} \in \mathbb{R}^n$ por $x_1, \ldots, x_n$. Os pesos sinápticos, para um certo neurônio k, são denotados por $w_{k1}, \ldots, w_{kn}$.

A ativação do sistema é dada pela seguinte expressão:

$$h_k(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{n} w_{kj} x_j. \tag{4.1}$$

Após essa soma dos sinais de entrada, ponderados pelos respectivos pesos sinápticos do

neurônio, deve-se comparar esse valor com um limiar e se for atingido um potencial de ação artificial então o sinal é passado adiante, caso contrário não. Esse processo recebe o nome de *função de ativação*. A função de ativação define o valor a ser enviado para outros neurônios. Em geral, denotamos a função de ativação por g, de modo que:

$$y_k = g(h_k(\mathbf{x}, \mathbf{w})). \tag{4.2}$$

Representamos o modelo, caracterizado pela equação (4.2), de acordo com a Figura 4.1.

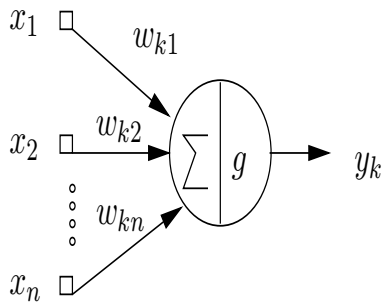

Fig. 4.1: Modelo de um neurônio artificial

Podemos escrever uma equação matricial para caracterizar um conjunto com m neurônios dispostos em paralelo como segue:

$$y = g(W\mathbf{x}), \tag{4.3}$$

onde $\mathbf{x}$ é o vetor coluna que contém os sinais de entrada, $W \in \mathbb{R}^{m\times n}$ representa a matriz de pesos sinápticos, g é uma função vetorial com componentes $g_i : \mathbb{R} \longrightarrow \mathbb{R}$ e $\mathbf{y}$ é o vetor coluna que contém a saída da rede.

Como exemplo de funções de ativação podemos citar a *função de limiar* descrita como:

$$g(x) = \begin{cases} 1, & \text{se} \quad x \geq 0 \\ 0, & \text{se} \quad x < 0 \end{cases} \tag{4.4}$$

Em razão deste tipo de função ser descontínua e, portanto, não derivável, temos que a minimização do erro fica comprometida no processo de treinamento da rede. Dessa maneira, uma alternativa de função de ativação são as funções sigmoidais, que são uma forma bastante utilizada na construção de redes neurais artificiais, caracterizadas pela equação:

$$g(x) = \frac{a}{1 + \mathrm{e}^{-bx+c}} + d. \tag{4.5}$$

A representação gráfica das funções tipo sigmoidal e limiar podem ser observadas na Figura

4.2.

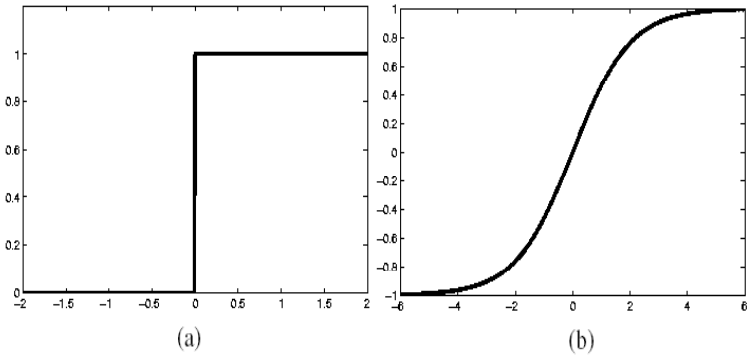

Fig. 4.2: (a) Função Limiar (b) Função tipo sigmoidal com escolhas a=2, b=1, c=0 e d=-1

4.4 Topologia

A *topologia* de uma rede neural refere-se à arquitetura, ou melhor, ao seu esquema de interconexão. A estrutura de uma rede neural é geralmente organizada em camadas de neurônios, que se distinguem nos seguintes tipos:

(1) *Camada de entrada*: camada de neurônios que recebe informações de entrada de fora da rede neural.

(2) *Camada de saída*: camada de neurônios que determina a saída da rede neural.

(3) *Camada escondida*: camada de neurônios cujas iterações se restrigem aos outros neurônios da rede neural.

A arquitetura de uma rede neural está intimamente ligada com o algoritmo de aprendizagem considerado para o ajuste dos parâmetros da rede e, em geral, podemos identificar duas classes de arquiteturas de rede:

(1) *Redes alimentadas adiante*

(i) *com camada única*: rede que não possui camadas de neurônios escondidos e cujas conexões têm somente a direção da camada de saída.

(ii) *com múltiplas camadas*: rede que possui uma ou mais camadas escondidas cujas conexões têm somente a direção da camada de saída.

De fato, o termo "camada única" se refere à *camada de saída* de neurônios. Não se conta a *camada de entrada* porque nesta não é realizado nenhum processamento.

(2) *Redes recorrentes*: Rede que possui conexões entre neurônios da mesma camada ou conexões de neurônios na direção da camada de entrada.

A Figura 4.3 ilustra a topologia de uma rede neural alimentada adiante e de uma rede recorrente.

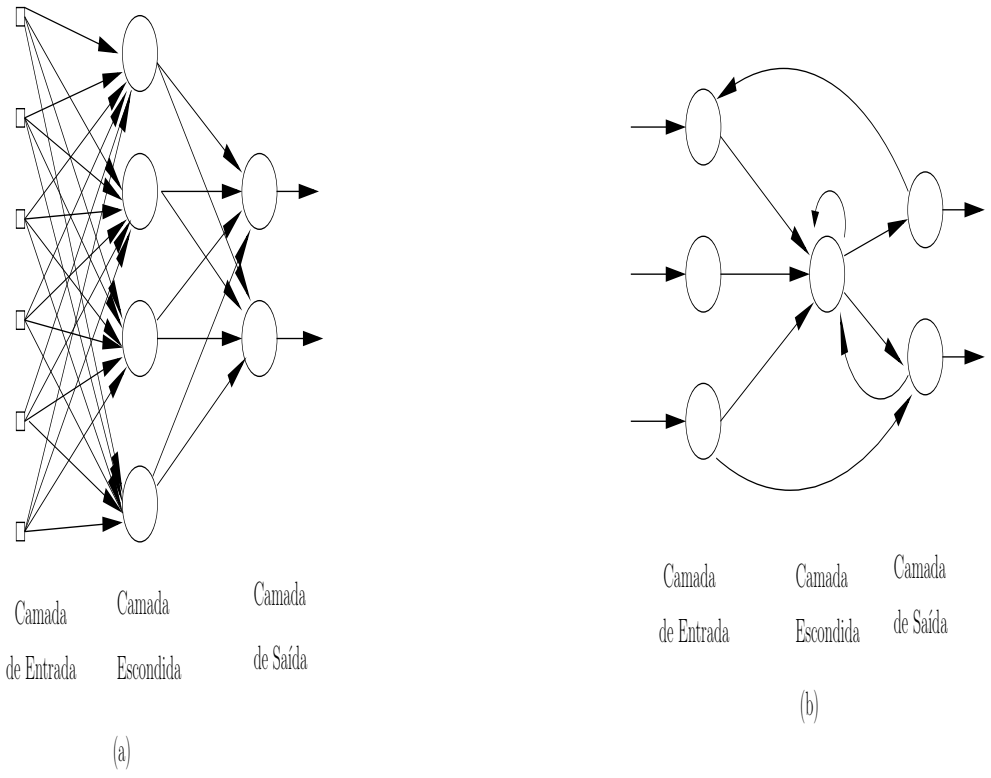

Fig. 4.3: (a) Rede alimentada adiante com uma camada escondida e uma camada de saída (b) Rede recorrente realimentada e com um neurônio oculto

Cada neurônio de uma rede neural artificial realiza uma operação de composição genérica, onde uma entrada no neurônio é primeiro processada por alguma função $h(.,.)$ das entradas e dos pesos sinápticos e então transformada por uma função de ativação $g(.)$. A estrutura do neurônio é definida pela função $h(.,.)$ que no caso de MLPs é simplesmente uma combinação linear. Formalmente podemos descrever formalmente uma rede neural do tipo alimentada adiante geral por meio das equações:

$$\mathbf{y}^l = G(\mathbf{x}^l) = (g(x_1^{(l)}), g(x_2^{(l)}), \ldots, g(x_{N_l}^{(l)})), \quad l = 1, 2, \ldots, L \tag{4.6}$$

$$x_k^{(l)} = h(\mathbf{y}^{l-1}, \mathbf{w}_k^{(l)}) \quad k = 1, 2, \ldots, N_l, \tag{4.7}$$

Sendo que l é a camada, N_l é o número de neurônios na camada l e $\mathbf{w}_k^{(l)} = (w_{k1}^l, w_{k2}^l, \ldots, w_{kN_{l-1}}^l)^\top$, $k = 1, 2, \ldots, N_l$ representa o vetor de pesos associados ao k-ésimo neurônio da camada l. O vetor $\mathbf{w} = (\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(L)})$ representa todos os pesos sinápticos da rede, onde $\mathbf{w}^{(l)} = (\mathbf{w}_1^{(l)}, \mathbf{w}_2^{(l)}, \ldots, \mathbf{w}_{N_L}^{(l)})$.

A estrutura geral de uma rede neural alimentada adiante, e a estrutura de uma camada da rede neural alimentada adiante, em particular, são ilustradas nas Figuras 4.4 e 4.5.

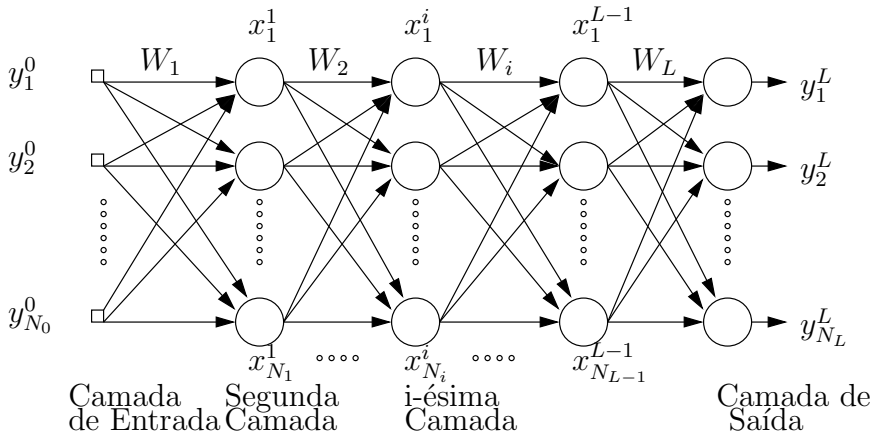

Fig. 4.4: Estrutura Geral de uma rede neural

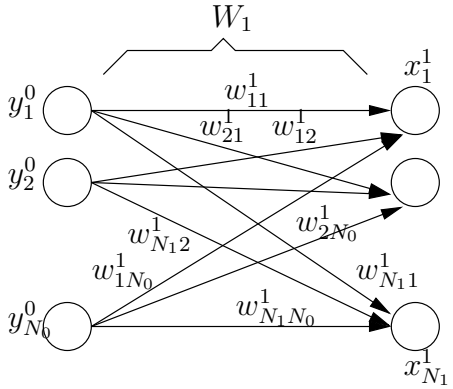

Fig. 4.5: Estrutura de camada de uma rede neural, para $l = 1$.

4.5 Aprendizagem e treinamento

Um atributo muito relevante de uma rede neural é a sua habilidade de aprender interagindo com o ambiente ou com algum recurso de informação. O aprendizado [22] em uma rede neural é normalmente conseguido através de um procedimento adaptativo, conhecido como regra de aprendizado, onde a rede é treinada com um conjunto de padrões, chamado padrões de treina-

mento. Neste procedimento, os pesos sinápticos da rede são incrementalmente ajustados com a finalidade de melhorar uma certa medida de desempenho ao longo do tempo.

Matematicamente, um processo de aprendizagem é baseado na busca de uma solução em um espaço multidimensional de parâmetros (pesos), que gradualmente maximiza ou minimiza uma dada função objetivo. Existem três tipos principais de aprendizado: o *aprendizado supervisionado*, o *aprendizado não supervisionado* e o *aprendizado por reforço* [22].

No *aprendizado supervisionado*, conhecido também como aprendizado com um professor, um conjunto de padrões de entrada $\{\mathbf{x}^p; p = 1, 2, \ldots, P\}$ recebido do ambiente, é associado com um padrão de saída desejado próprio $\{\mathbf{d}^p; p = 1, 2, \ldots, P\}$, conhecido de antemão. De fato, dado um padrão de entrada $\mathbf{x}$, queremos determinar um conjunto de pesos $\mathbf{w}$, que resultará em um particular padrão de saída $\mathbf{y}$ da rede de acordo com a equação 4.2, ou seja, no treinamento da rede, normalmente os pesos são ajustados gradualmente, e em cada passo do processo de aprendizagem eles são atualizados, o que implica na redução do erro entre a saída da rede $\mathbf{y}^k$ e um correspondente padrão de saída desejado $\mathbf{d}^p$. No *aprendizado não-supervisionado* nos é fornecido um conjunto de treinamento, digamos $\{\mathbf{x}^p; p = 1, 2, \ldots, P\}$ de vetores não rotulados em $\mathbb{R}^n$. O objetivo consiste em categorizar ou descobrir similaridades nos padrões de entrada. Com respeito ao aprendizado por reforço, temos que o aprendizado do mapeamento de entrada e saída é realizado através da interação contínua com o ambiente, visando minimizar um índice escalar de desempenho [22].

4.5.1 Regra de aprendizagem back-propagation

O objetivo do treinamento supervisionado em perceptrons de múltiplas camadas (MLPs) é conseguir um conjunto de parâmetros $\mathbf{w}_k^l$, $k = 1, \ldots, N_l$, $l = 1, 2, \ldots, L$, tal que o erro seja minimizado utilizando a técnica de otimização da máxima descida [31], onde o vetor de pesos é ajustado na direção oposta ao crescimento do erro. Em linhas gerais, *backpropagation* [21, 61] diz respeito ao cálculo das derivadas da função erro. Considere o conjunto de treinamento $\{\mathbf{x}^p, \mathbf{d}^p\}$, $p = 1, 2, \ldots, P$, onde $\mathbf{d}^p = (d_1^p, d_2^p, \ldots, d_{N_L}^p)$ representa o padrão de saída desejado da rede para o padrão de entrada apresentado $\mathbf{x}^p$. O erro total é medido em termos dos erros quadráticos E^p gerados pelos padrões de entrada $\mathbf{x}^p$. Ou seja, o erro é escrito como:

$$E = \sum_{p=1}^{P} E^p. \tag{4.8}$$

$$E^p = \frac{1}{2} \sum_{i=1}^{N_L} (d_i^p - y_i^p)^2. \tag{4.9}$$

Os erros E e E^p são funções de todos os pesos da rede. Devido a representação de E como uma soma de erros individuais E^p, temos que o problema de determinar as derivadas de E com respeito aos pesos sinápticos se reduz a determinar as derivadas de E^p com respeito aos pesos.

O algoritmo para calcular as derivadas de E^p com respeito aos pesos realiza os seguintes passos:

(1) Apresente o padrão $\mathbf{x}^p$ à rede e calcule as ativações nos neurônios;

(2) Calcule os sinais erros $e_k^{(L)}$ relacionados às unidades de saída Eq. (4.12).

(3) Use os sinais erros $e_k^{(l+1)}$ das unidades da camada $l+1$ para calcular os sinais erros estimados $e_k^{(l)}$ das unidades da l-ésima camada (procedimento back-propagation) Eq. (4.23).

(4) Avalie as derivadas usando as equações (4.15) e (4.22).

Suponhamos que o padrão $\mathbf{x}^p = (y^p{}_1^{(0)}, \ldots, y^{p(0)}_{N_0})$ já tenha sido apresentado à rede. Denotaremos a sua saída por $\mathbf{y} = (y_1, \ldots, y_{N_L})$ ao invés de $\mathbf{y}^{p(L)} = (y_1^p, \ldots, y_{N_L}^p)$. A partir de agora omitiremos o índice p, deixando claro que os cálculos explicitados a seguir dizem respeito ao padrão de treinamento $\mathbf{x}^p$.

Cada unidade de uma camada escondida ou da camada de saída primeiramente calcula uma combinação linear de suas entradas e de seus pesos:

$$x_j^{(l)} = \sum_i w_{ji}^{(l)} y_i^{(l-1)}. \tag{4.10}$$

A ativação da j-ésima unidade da camada l é obtida por meio da aplicação de uma função de ativação diferenciável g à soma $x_j^{(l)}$:

$$y_j^{(l)} = g(x_j^{(l)}). \tag{4.11}$$

Definimos o sinal erro, associado à camada de saída, por:

$$\mathbf{e}^{(L)} = (\mathbf{d}^p - \mathbf{y}). \tag{4.12}$$

Como as saídas desejadas para as unidades de saída são explicitamente especificadas, aplica-se a regra da cadeia para calcular as derivadas da função E^p com respeito aos pesos da camada de saída. Ou seja,

$$\begin{aligned} \frac{\partial E^p}{\partial w_{kj}^{(L)}} &= \frac{\partial E^p}{\partial e_k^{(L)}} \frac{\partial e_k^{(L)}}{\partial y_k^{(L)}} \frac{\partial y_k^{(L)}}{\partial x_k^{(L)}} \frac{\partial x_k^{(L)}}{\partial w_{kj}^{(L)}}, & (4.13)\\ &= (d_k - y_k^{(L)})(-1) g'(x_k^{(L)}) y_j^{(L-1)}, & (4.14)\\ &= e_k^{(L)}(-1) g'(x_k^{(L)}) y_j^{(L-1)}. & (4.15) \end{aligned}$$

O cálculo das derivadas da função E^p com respeito aos pesos da camada escondida não é tão óbvio quanto o cálculo das derivadas que dizem respeito à camada de saída porque não temos disponível um conjunto de valores de saída desejadas para as unidades escondidas. Neste caso, usando a regra da cadeia temos:

$$\frac{\partial E^p}{\partial w_{kj}^{(l)}} = \frac{\partial E^p}{\partial y_k^{(l)}} \frac{\partial y_k^{(l)}}{\partial x_k^{(l)}} \frac{\partial x_k^{(l)}}{w_{kj}^{(l)}}; \tag{4.16}$$

Na qual,

$$\frac{\partial x_k^{(l)}}{w_{kj}^{(l)}} = y_j^{(l-1)}, \tag{4.17}$$

$$\frac{\partial y_k^{(l)}}{\partial x_k^{(l)}} = g'(x_k^{(l)}), \tag{4.18}$$

e

$$\frac{\partial E^p}{\partial y_j^{(l)}} = \frac{\partial \{\frac{1}{2}\sum_{k=1}^{N_l}(d_k - g(x_k^{(l)}))^2\}}{\partial y_j^l}, \tag{4.19}$$

$$= -\sum_{k=1}^{N_L}(d_k - g(x_k^{(l)}))\frac{\partial g(x_k^{(l)})}{\partial y_j^{(l)}}, \tag{4.20}$$

$$= -\sum_{k=1}^{N_L}(e_k^{(l)})g'(x_k^{(l)})w_{kj}^{(l)}. \tag{4.21}$$

Substituindo as equações (4.17), (4.18) e (4.21) na equação (4.16), nós obtemos:

$$\frac{\partial E^p}{\partial w_{kj}^{(l)}} = \left[\sum_{k=1}^{N_L} e_k^{(l)} g'(x_k^{(l)}) w_{kj}^{(l)}\right] g'(x_k^{(l)}) y_j^{(l-1)}. \tag{4.22}$$

Comparando a equação (4.22) com a equação (4.15), nós podemos estimar o sinal erro $e_k^{(l)}$ associado à k-ésima unidade escondida por meio de uma retropropagação como segue:

$$e_k^{(l)} = \sum_{k=1}^{N_{l+1}} e_k^{(l+1)} g'(x_k^{(l+1)}) w_{kj}^{(l+1)}. \tag{4.23}$$

Para $l < L$ notemos que a equação (4.23) pode ser reescrita como:

$$e_k^{(l)} = \sum_{k=1}^{N_{l+1}} e_k^{(l+1)} g'(x_k^{(l+1)}) \frac{\partial x_k^{(l+1)}}{\partial y_k^{(l)}}. \tag{4.24}$$

O membro direito da equação (4.24) é calculado no passo 3 do algoritmo backpropagation para determinar os sinais erros da l-ésima camada, dado que os sinais erros da camada $l+1$ já estão calculados.

4.5.2 Treinamento de perceptrons de múltiplas camadas

O algoritmo de treinamento para perceptrons de múltiplas camadas utiliza a retropropagação do erro e a técnica de otimização de máxima descida. O gradiente ∇E^p da função erro E^p com respeito aos pesos da rede é um vetor dado por:

$$\nabla E^p = \begin{pmatrix} \nabla_{\mathbf{w}^{(1)}} E^p \\ \nabla_{\mathbf{w}^{(2)}} E^p \\ \vdots \\ \nabla_{\mathbf{w}^{(L)}} E^p \end{pmatrix}. \tag{4.25}$$

Algoritmo de Treinamento

Passo (1) Defina inicialmente $t = 0$ e inicialize randômicamente os pesos $\mathbf{w}(t)$ com componentes em $[0, 1]$.

Passo (2) Para $p = 1, \ldots, P$ faça:

(2.1) Apresente o padão de entrada $\mathbf{x}^p$ à rede e determine a sua saída $\mathbf{y}$ usando as equações (4.6) e (4.7).

(2.2) Realize a fase *backpropagation*, Eq. (4.23), para calcular os gradientes $\nabla_{\mathbf{w}_k^{(l)}} E^p$ usando as equações (4.15) e (4.16).

Passo (3) Atualize os pesos

$\mathbf{w}(t+1) = \mathbf{w}(t) + \triangle \mathbf{w}^(t)$;
onde $\triangle \mathbf{w}(t) = -\mu_0 \sum_{p=1}^{P} \nabla E^p \mid_{\mathbf{w}(t)}$.

Atualize $t = t + 1$.

Passo (4) Repita os passos (2) e (3) até um critério de parada ser alcançado.

A constante positiva μ_0 é definida como parâmetro de taxa de aprendizagem, ou tamanho do passo e tem a finalidade de regular o balanceamento entre estabilidade e velocidade de convergência do processo iterativo. No algoritmo de treinamento acima, os pesos são atualizados quando todo o conjunto de padrões de treinamento foi apresentado à rede. Quando acontece isso dizemos que o algoritmo opera em modo *batch* [21, 22, 61]. Um outro modo alternativo é o modo *estocástico* [21, 22, 61], onde os pesos são atualizados depois que cada padrão de treinamento é apresentado como segue:

Passo (1) Defina inicialmente $t = 0$ e inicialize randômicamente os pesos $\mathbf{w}(t)$ com componentes em $[0, 1]$..

Passo (2) Para $p = 1$ faça

(2.1) Apresente o padão de entrada $\mathbf{x}^p$ à rede e determine a sua saída $\mathbf{y}^p$ usando as equações (4.6) e (4.7).

(2.2) Realize a fase *backpropagation*, Eq. (4.23), para calcular os gradientes $\nabla_{\mathbf{w}_k^{(l)}} E^p$ usando as equações (4.15) e (4.16).

Passo (3) Atualize os pesos

$\mathbf{w}(t+1) = \mathbf{w}(t) + \triangle\mathbf{w}(t)$;
onde $\triangle\mathbf{w}(t) = -\mu_0 \nabla E^p \mid_{\mathbf{w}(t)}$.

E atualize $t = t + 1$. O novo valor de p é dado por $(p+1) mod P$

Passo (4) Repita os passos (2) e (3) até um critério de parada ser alcançado.

4.6 Generalização

Um dos objetivos dos processos de aprendizagem e treinamento é a *generalização* [4],[44] que é definida como a habilidade que uma rede neural treinada tem de interpretar sucessivamente padrões que não lhe foram apresentados previamente e produzir uma resposta apropriada. A propriedade de generalização coloca a rede neural em uma categoria diferente em relação a um sistema tal como uma leitura de tabelas "look up table", onde é necessário armazenar toda a informação requerida para se fazer referência em ocasiões futuras. A tarefa de aprender a partir de exemplos pode ser considerada em muitos casos ser equivalente ao problema de ajuste de curva, onde o objetivo se torna aprender o mapeamento não linear de entrada

e saída quando interpretamos uma rede neural, com n neurônios na camada de entrada e m neurônios na camada de saída, como uma função $F_W : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, onde $W = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n\}$ representa o conjunto de todos os vetores peso. Então, se o conjunto de padrões de entrada $\{\mathbf{x}^p; p = 1, 2, \ldots, P\}$, recebido do ambiente, é associado com um padrão de saída desejado próprio $\{\mathbf{d}^p; p = 1, 2, \ldots, P\}$, ambos conhecidos de antemão, temos que a rede neural é expressa como uma função dos pesos sinápticos. Como todo o conhecimento da rede neural está armazenado em suas sinapses, ou seja, nos pesos atribuídos às conexões entre os neurônios, isto significa que sobre esta perspectiva, o processo de aprendizagem se torna a escolha do melhor ajuste dos parâmetros livres da rede, que corresponde, em termos matemáticos, a resolver o seguinte problema de otimização:

$$\min_W \parallel \mathbf{d}^k - F_W(\mathbf{x}^k) \parallel, \forall k = 1, 2, \ldots, n \tag{4.26}$$

Desta forma, podemos considerar a generalização como o efeito de uma boa regressão não linear sobre os dados de entrada, como pode ser observado na Figura 4.6(a). Uma rede neural que é projetada para generalizar bem, produzirá um mapeamento de entrada e saída correto, mesmo quando a entrada for um pouco diferente dos exemplos usados para treinar a rede.

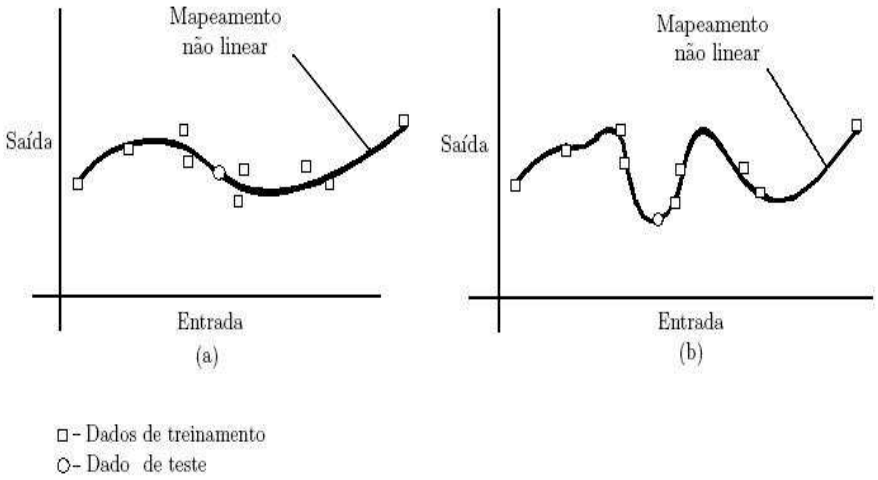

Fig. 4.6: (a) Dados ajustados corretamente (boa generalização) (b) Dados excessivamente ajustados (generalização ruim)

Mas quando a rede aprende um número excessivo de exemplos de entrada e saída, ela pode acabar memorizando os dados de entrada e saída, ou seja, ela encontra uma característica que está presente nos dados de treinamento, um ruído por exemplo, mas não na função subjacente

que deve ser modelada, tal fenômeno é conhecido como *excesso de treinamento* ou *sobreajuste* [45, 42]. Quando a rede é excessivamente treinada ela perde a habilidade de generalizar entre padrões de entrada e saída similares. De um modo geral, consideramos que a generalização está intimamente relacionada com a complexidade do problema e com a complexidade da rede, onde frequentemente o número de parâmetros livres da rede, i.e. o números de pesos, é usado como uma medida da complexidade da rede. Deve-se atender a um compromisso entre a complexidade do problema e a complexidade da rede.

Como exemplo, podemos considerar as seguintes abordagens para tratar a questão da generalização.

(1) A arquitetura da rede é fixa e o que se deve determinar é o tamanho do conjunto de treinamento apropriado, a função de ativação e complexidade computacional para se obter boa generalização.

(2) O tamanho do conjunto de treinamento é fixo, assim como a função de ativação e a natureza do problema a ser resolvido e devemos determinar a arquitetura mais apropriada que, de uma certa forma, está subordinada à escolha do número de camadas, neurônios e pesos (estes estão associados à complexidade da rede) para obter boa generalização.

Um modo razoável e direto de estimar-se a habilidade de generalização de um sistema é medir o erro em um conjunto de dados separado que não foi previamente apresentado a rede neural durante a fase de treinamento. Neste sentido, a *validação* [61], que é uma técnica estatística de predição, fornece um princípio de orientação interessante. O conjunto total de dados é dividido em um conjunto de *treinamento* e um conjunto de *testes*, além disso o conjunto de treinamento é particionado em dois subconjuntos disjuntos:

(i) *subconjunto de estimação*, usado para estimar o modelo de rede neural.

(ii) *subconjunto de validação*, usado para selecionar o modelo de rede neural.

De fato, deve-se validar o modelo com um conjunto de dados diferente daquele usado para estimar os parâmetros. Desta maneira, utiliza-se o *conjunto de treinamento* para avaliar o desempenho de vários modelos candidatos e, assim, escolher aquele que tem o melhor desempenho sobre o *subconjunto de validação*. No entanto, existe a possibilidade de que o modelo assim selecionado, com os valores de parâmetros com melhor desempenho, possa acabar ajustando excessivamente o subconjunto de validação e para evitar isso, o desempenho de generalização do modelo selecionado é medido sobre o *conjunto de teste*, que é diferente do *subconjunto de validação*. O metodo da *validação cruzada* [61, 54] fornecerá uma partição de um conjunto de dados único em subconjuntos distintos que servirão tanto como dados de estimação quanto

dados de validação em diferentes iterações do método de validação geral. A validação cruzada pode ser usada para detectar quando começa o *excesso de treinamento* durante o treinamento supervisionado de uma rede neural. O treinamento é então interrompido antes que o erro no conjunto de validação cruzada for mais alto do que o erro que foi checado na última vez, para evitar *excesso de treinamento*. Enfrenta-se esse problema basicamente usando técnicas, para melhoria da generalização, como *parada antecipada* [42], como pode ser observado na Figura 4.7, *regularização* (nesta técnica soma-se um termo de penalidade à função erro que se pretende minimizar). Como um caso particular de técnica de regularização, pode-se utilizar o *decaimento de pesos*, cujo termo de penalidade é a soma de quadrados de pesos vezes uma certa constante de decaimento. Consideraremos que na prática um conjunto independente que pode ser escolhido como um conjunto de validação não é frequentemente disponível. Seja o conjunto de dados original denotado por R, e digamos que

$$R = R_1 \cup R_2 \cup \ldots \cup R_n, \quad \text{onde} \quad R_i \cap R_j = \emptyset, \forall i \neq j. \tag{4.27}$$

O método de validação cruzada é executado n vezes e na i-ésima iteração definimos o subconjunto de estimação como sendo o conjunto $R - R_i$ e o subconjunto de validação é definido por R_i.

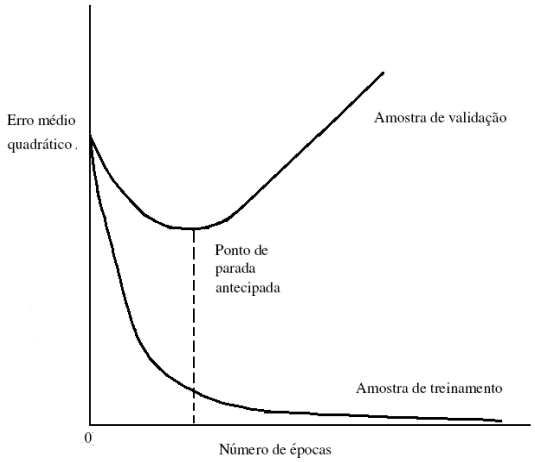

Fig. 4.7: Regra de parada antecipada baseada na validação cruzada

4.7 Introdução aos classificadores de padrões

O objetivo de classificação [45] de padrões é associar padrões de entrada a um número finito, M, de classes. No que se segue, vamos considerar que padrões de entrada consistem de vetores de entrada $\mathbf{x}$, contendo n entradas pertencentes a $\mathbb{G}$, denotados por $x_1, x_2, \ldots, x_n$. As entradas dos vetores representam medidas de características selecionadas para serem úteis na distinção entre classes. Os padrões de entrada podem ser vistos como pontos em um espaço multidimensional definido pelas medidas de características de entrada.

A proposta de um classificador de padrões é particionar este espaço multidimensional em regiões de decisão que indiquem a classe na qual a entrada pertence. A aplicação de um classificador de padrões requer seleção de características que devem ser ajustadas separadamente para cada domínio do problema.

As características devem conter informações necessárias para se distinguir entre classes, não ser seníveis à invariabilidades irrelevantes nas entradas, devem ser limitadas em número para permitir computação eficiente de funções discriminantes e limitar a quantidade do conjunto de treinamento necessária. Dentre os classificadores lineares mais conhecidos, podemos citar o perceptron, que opera sob a premissa de que os padrões a serem classificados sejam linearmente separáveis, e o classificador bayesiano que minimiza a probabilidade de erro de classificação. No contexto das redes neurais artificiais, as superfícies de decisão obtidas pelos classificadores desempenham um papel fundamental no entendimento e desenvolvimento dos algoritmos de aprendizagem. No perceptron morfológico [60], no perceptron morfológico dentrítico [50] e na rede FLNN, evidenciaremos a importância das superfícies de decisão obtidas por estes classificadores morfológicos no desenvolvimento dos seus algoritmos de aprendizagem.

Ao contrário de algoritmos de aprendizagem de redes neurais convencionais, que são conseguidos utilizando uma fase backpropagation para calcular derivadas de uma função Erro com respeito aos pesos e então atualizando os pesos usando alguma regra de aprendizagem, os algoritmos de aprendizagem destas redes neurais morfológicas operam em um número finito de passos e não apresentam problemas de convergência. De fato, as superfícies de decisão produzidas pelos algoritmos de aprendizagem destas redes neurais morfológicas são essencialmente hipercaixas em $\mathbb{G}^n$, ou seja, são superfícies fechadas que delimitam os padrões a serem classificados. O algoritmo de treinamento destas redes neurais morfológicas fornece erro zero de padrões mal-classificados no conjunto de treinamento, no final do treinamento. Todos os padrões do conjunto de treinamento serão corretamente identificados depois que o treinamento pára. Em linhas gerais, podemos dizer que estas redes neurais morfológicas são capazes de resolver uma

dicotomia arbitrária para um conjunto arbitrário de padrões.

Capítulo 5

Redes Neurais Morfológicas

5.1 Introdução ao modelo neuronal morfológico

O conceito de redes neurais morfológicas surgiu à medida em que a álgebra de imagens ia se desenvolvendo. Foi demonstrado que uma sub-álgebra da álgebra de imagens [51] inclui as formulações matemáticas de modelos de redes neurais clássicas. Desde então alguns artigos envolvendo redes neurais morfológicas apareceram. J.L. Davidson implementou redes neurais morfológicas para solucionar o problema de identificação de template e problemas de classificação de alvo [13]. C.P. Suarez Araujo aplicou redes neurais morfológicas para computar invariâncias visuais [58]. Ritter e Sussner forneceram um maior compreendimento e uma base mais rigorosa para computação com redes neurais morfológicas [47, 48]. Entre os diferentes modelos morfológicos que vêm sendo desenvolvidos, estão o perceptron morfológico [48, 60], o perceptron morfológico dentrítico [50], as memórias associativas morfológicas [49, 62, 63, 64, 67, 68], redes neurais morfológicas modulares [1, 2, 14], redes neurais morfológicas de pesos compartilhados [27, 28], redes neurais morfológicas de regularização [16], redes neurais morfológica /posto/lineares híbridas [38]. Redes neurais morfológicas e morfológicas/lineares híbridas possuem muitas aplicações em reconhecimento de padrões [26, 28, 38], processamento de imagens e visão computacional [19, 43], controle e previsão [2, 5, 69].

Na seção 2.2 fizemos uma discussão sobre a teoria de reticulados para caracterizar a morfologia matemática, e definimos *erosão*, *anti-erosão*, *dilatação* e *anti-dilatação* como operadores sobre reticulados completos, no sentido das equações (2.3), (2.4), (2.5) e (2.6), onde estes operadores distribuem sobre o ínfimo e o supremo, respectivamente, de qualquer coleção de elementos do reticulado. As redes neurais morfológicas constituem uma classe de RNAs que executa uma operação elementar da morfologia matemática (*erosão*, *anti-erosão*, *dilatação* ou *anti-dilatação*), em um neurônio ou em uma camada, de um reticulado completo $\mathbb{L}$ para outro reticulado completo $\mathbb{M}$ seguida da aplicação de uma função de ativação. O primeiro modelo

neural morfológico foi proposto por J.L. Davidson e G.X. Ritter [51].

Um ponto central da morfologia matemática é a decomposição de transformações entre reticulados completos em termos das operações elementares. Banon e Barrera propuseram um teorema mostrando, que qualquer transformação $\psi : \mathbb{L} \longrightarrow \mathbb{M}$ pode ser decomposta como o supremo de ínfimos entre *erosões* e *anti-dilatações* [3]. Note que o operador ínfimo pode ser expresso como uma *erosão*, dado que este operador claramente comuta com o ínfimo de uma coleção de elementos de um reticulado $\mathbb{L}$, similarmente o operador supremo pode ser expresso como uma *dilatação*.

Na seção 3.3 apresentamos a imersão da morfologia matemática (para o caso binário e em tons de cinza) na álgebra minimax. Isto se fez necessário porque os cálculos que neurônios de certos tipos de redes neurais morfológicas, que abordaremos, serão baseados em *produtos máximo* e *mínimo* de matrizes por vetores, definidos nas equações (3.4) e (3.5), que assumem valores no *blog* $(\mathbb{G}, \vee, \wedge, +, +')$, onde $\mathbb{G} = \bar{\mathbb{R}}$ ou $\mathbb{G} = \bar{\mathbb{Z}}$. Uma definição importante que colocaremos aqui, diz respeito a definição de um *neurônio artificial generalizado*. Definimos um neurônio artificial generalizado como sendo aquele que possui uma função de agregação que combina as entradas e os pesos sinápticos e em seguida aplica uma função de ativação. A Figura 5.1 ilustra um neurônio artificial generalizado. O neurônio morfológico é um caso

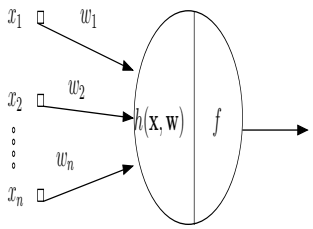

Fig. 5.1: Neurônio artificial generalizado

especial do neurônio artificial generalizado. Pesquisas recentes em neurociência evidenciaram que um neurônio pode de fato executar uma operação de máximo [73]. Vamos definir uma *unidade de processamento* como sendo um neurônio artificial generalizado. No capítulo 4, nós descrevemos que uma rede neural do tipo alimetada adiante fica bem definida pelas equações (4.6) e (4.7). A equação (4.7) diz simplesmente que o k-ésimo neurônio da camada l calcula uma operação $h(\mathbf{y}^{l-1}, \mathbf{w}_k)$ que depende do vetor de entradas e do vetor de pesos associados ao k-ésimo neurônio. Se este neurônio for um neurônio morfológico, esta operação deve ser uma *erosão*, *anti-erosão* ou uma *dilatação*, *anti-dilatação* no sentido da teoria de reticulados. Se o k-ésimo neurônio de uma camada l realiza uma erosão, este neurônio é dito ser um *neurônio erosivo*. Similarmente, definimos um neurônio *dilatativo*, *anti-dilativo* e *anti-erosivo*. Quando uma camada é constituída apenas de neurônios erosivos, dizemos que a camada realiza uma *erosão* e ,portanto, ela é dita ser uma camada *erosiva*. Similarmente definimos uma camada

dilativa, *anti-dilativa* e *anti-erosiva*.

5.2 Perceptron Morfológico (MP)

O *perceptron* [35, 52, 60] é um modelo de rede neural alimentada adiante baseado em uma rede de unidades de decisão binária que modelam as células nervosas do cérebro humano. Seu papel é agir como um classificador ou, simplesmente, reconhecer padrões. O modelo do perceptron convencional é naturalmente descrito pelo par de equações (4.1) e (4.2), com a função limiar g, definida pela equação (4.4), como função de ativação. Dado um vetor de entradas $\mathbf{x} \in \mathbb{G}^n$ e um vetor de pesos sinápticos $\mathbf{w} \in \mathbb{G}^n$, definimos o perceptron morfológico como uma RNM alimentada adiante, tal que em cada neurônio calcula-se uma das quatro operações seguintes:

$$\varepsilon_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^{\top} \boxslash \mathbf{x}; \tag{5.1}$$

$$\delta_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^{\top} \boxbslash \mathbf{x}; \tag{5.2}$$

$$\bar{\varepsilon}_{\mathbf{w}}(\mathbf{x}) = \delta_{\mathbf{w}} \circ \nu_*(\mathbf{x}); \tag{5.3}$$

$$\bar{\delta}_{\mathbf{w}}(\mathbf{x}) = \varepsilon_{\mathbf{w}} \circ \nu_*(\mathbf{x}). \tag{5.4}$$

Em seguida aplica-se uma certa função de ativação g. Com relação à notação utilizada, temos que x_j denota o valor da j-ésima entrada, w_j denota o peso sináptico associado à j-ésima entrada, e f é uma certa função de ativação. Se g corresponder à função limiar, $g : \bar{\mathbb{G}} \longrightarrow \{0, 1\}$ é definida de acordo com a equação (4.4)

A equação

$$y = g(\varepsilon_{\mathbf{w}}(\mathbf{x})) = 0, \tag{5.5}$$

por exemplo, determina uma superfície, de dimensão $(n-1)$, que divide o espaço euclideano n-dimensional em duas regiões. Cada região recebe o nome de *fronteira de decisão* ou *superfície de decisão* e cada uma representa uma classe de padrões associados. Antes de o perceptron agir como um classificador, os valores de seus pesos devem ser determinados. No caso em que a superfície de decisão é conhecida a priori, os pesos podem ser obtidos analiticamente, se isto não acontecer, os pesos são determinados em um estágio de treinamento, onde o perceptron determinará sua própria superfície de decisão, por meio de um algoritmo de aprendizagem. Entender as superfícies de decisão é relevante para desenvolver algoritmos de aprendizagem para a MP.

Nesta seção iremos apresentar um algoritmo de aprendizagem que tem por finalidade resolver exemplos arbitrários de problemas de classificação de duas classes. Denotamos o conjunto de padrões pertencentes à classe 0 por C_0 e o conjunto de padrões pertencentes à classe 1 por C_1. O algoritmo é baseado na eliminação de padrões mal-classificados que, basicamente, faz um hi-

percubo inicial, contendo todos os padrões da classe C_1 que estão atualmente mal-classificados, reduzir-se através da eliminação de padrões exteriores e de regiões menores que contêm eles. O treinamento termina quando todos os padrões exteriores no conjunto de treinamento tiverem sido removidos. Serão fornecidos k-padrões de treinamento em $\mathbb{R}^n$ e o algoritmo deverá encontrar pesos apropriados, tal que cada um dos padrões de treinamento será corretamente classificado pelo perceptron morfológico. O algoritmo também determina quantos neurônios na camada escondida são necessários para resolver o problema dado.

Algumas definições importantes:

(i) Denotemos por $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^P \in \mathbb{R}^n$ os padrões de treinamento fornecidos.

(ii) Define-se os seguintes conjuntos de índices:

$$K(0) = \{j \in \{1, \ldots, P\} : \mathbf{x}^j \in C_0\}. \tag{5.6}$$
$$K(1) = \{j \in \{1, \ldots, P\} : \mathbf{x}^j \in C_1\}. \tag{5.7}$$

(iii) Seja g a função de ativação definida por (4.4). Definimos os seguintes conjuntos de índices que dizem respeito à resposta da rede neural:

$$L(0) = \{j \in \{1 \ldots, P\} : f(g(\mathbf{x}^j)) = 0\}. \tag{5.8}$$
$$L(1) = \{j \in \{1 \ldots, P\} : f(g(\mathbf{x}^j)) = 1\}. \tag{5.9}$$

(iv) Denotamos o conjunto de índices dos padrões pertencentes à classe C_1 que estão mal-classificados por:

$$D = \{j \in L(0) \cap K(1)\}.$$

(v) Seja $Q = \text{box}(\mathbf{p}^\perp, \mathbf{p}^\top)$, onde $\mathbf{p}^\perp, \mathbf{p}^\top \in \mathbb{G}^n$, denotam o *canto inferior* e *superior* respectivamente. Definimos os seguintes conjuntos, ilustrados na Figura 5.2, que representam faixas:

$$C^i = [p_1^\perp, p_1^\top] \times [p_2^\perp, p_2^\top] \times [p_{i-1}^\perp, p_{i-1}^\top] \times [-\infty, \infty] \times [p_{i+1}^\perp, p_{i+1}^\top] \ldots \times [p_n^\perp, p_n^\top]. \tag{5.10}$$

Sendo que os conjuntos C^i são uma modificação no algoritmo original proposto em [60].

Algoritmo de treinamento do perceptron morfológico

Definimos o perceptron morfológico inicial como aquele que calcula a função $g(\varphi(\mathbf{x}))$ para padrões de entrada $\mathbf{x} \in \mathbb{G}^n$, onde $\varphi : \bar{\mathbb{G}}^n \to \bar{\mathbb{G}}$ é constante com $\varphi(\mathbf{x}) = -\infty$, $\forall \mathbf{x} \in \mathbb{G}^n$.

Passo 1 (Encontra uma caixa contendo apenas padrões pertencentes à classe C_1) Inicialmente consideramos que todos os padrões pertencentes à classe C_1 estão atualmente mal-classificados, ou seja, $D = K(1)$.

1 *While* $D \neq \emptyset$, faça

1.1 Construa uma caixa $Q = \text{box}(\mathbf{p}^\perp, \mathbf{p}^\top)$, onde $\mathbf{p}^\perp$ e $\mathbf{p}^\top$, denotam o *canto inferior* e *superior* respectivamente, e são dados por:

$$p_i^\top = \bigvee_{j \in D} x_i^j, \forall i = 1, \ldots, n \tag{5.11}$$

$$p_i^\perp = \bigwedge_{j \in D} x_i^j, \forall i = 1, \ldots, n \tag{5.12}$$

1.2 Se exise um índice i_0 tal que $p_{i_0}^\top = p_{i_0}^\perp$, então redefinimos as coordenadas dos cantos superiores e inferiores como:

$$p_i^\perp = \max\{x_i < p_i^\perp : \mathbf{x} \in C_0 \cap C^i\}. \tag{5.13}$$

$$p_i^\top = \min\{x_i > p_i^\top : \mathbf{x} \in C_0 \cap C^i\}. \tag{5.14}$$

1.3 Caso contrário, inicie o seguinte loop, para todo $j = 1, \ldots, P$ tal que $\mathbf{x}^j \in C_0$:

(*a*) Defina $\mathbf{x} = \mathbf{x}^j$. Determine a maior caixa (*com respeito ao volume*) $Q' \subseteq Q$ que contenha o maior número de padrões da classe C_1 (*este último fato foi uma modificação no algoritmo original* [60]) que estão malclassificados, e tal que $\mathbf{x}$ não pertença ao interior de Q'. Atualize Q da seguinte forma:

$$Q = Q'.$$

(*Caso exista mais que uma caixa Q' que contenha o número máximo de padrões da classe C_1 que estão mal-classificados, escolha aleatóriamente uma dentre essas e atualize $Q = Q'$*).

Fim do loop.(*No fim deste loop, a caixa Q contém somente padrões da classe C_1 que estão mal-classificados.*)

(*b*) Aumente a caixa Q atualizada, obtida no item a, redefinindo os seus pontos de canto $\mathbf{p}^{\perp}$ e $\mathbf{p}^{\top}$ da seguinte forma:

$$p_i^{\perp} = \max\{x_i \leq p_i^{\perp} : \mathbf{x} \in C_0 \cap C^i\}. \tag{5.15}$$
$$p_i^{\top} = \min\{x_i \geq p_i^{\top} : \mathbf{x} \in C_0 \cap C^i\}. \tag{5.16}$$

1.4 Determine o menor hipercubo B que contém todos os padrões da classe C_1 mal-classificados no interior de Q, ou seja

$$p_{b_i}^{\top} = \bigvee_{j \in D} x_i^j, \forall i = 1, \ldots, n \tag{5.17}$$
$$p_{b_i}^{\perp} = \bigwedge_{j \in D} x_i^j, \forall i = 1, \ldots, n \tag{5.18}$$

1.5 Determine um hipercubo intermediário C cujo canto superior e inferior, denotados respectivamente por $\mathbf{c}^{\top}$ e $\mathbf{c}^{\perp}$, são dados pela média aritmética dos cantos de B e de Q, ou seja,

$$c_i{}^{\top} = \frac{p_{b_i}{}^{\top} + p_i^{\top}}{2}. \tag{5.19}$$
$$c_i{}^{\perp} = \frac{p_{b_i}{}^{\perp} + p_i^{\perp}}{2}. \tag{5.20}$$

Passo 2 (Implementando o perceptron morfológico) No passo 1 encontramos os pontos de canto $\mathbf{c}^{\top}$ e $\mathbf{c}^{\perp}$, que correspondem aos pesos sinápticos do nosso modelo neural.

1.6 Neste passo calcularemos a ativação nos neurônios:

$$\varphi(\mathbf{x}) = \varphi(\mathbf{x}) \vee \left[\bigwedge_{i=1}^{n} (x_i - c_i{}^{\top}) \wedge \bigwedge_{i=1}^{n} (c_i{}^{\perp} - x_i) \right].$$

1.7 Aplicamos a função de ativação limiar:

$$y = g(\varphi(\mathbf{x})).$$

1.7 Atualizamos o conjunto de indices D e retornamos ao *while*

fim do while

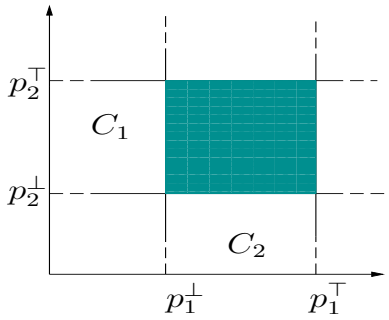

Fig. 5.2: Representação dos Conjuntos C_i, $i = 1, 2$

Vejamos agora como é a arquitetura desta rede neural e a operação elementar da morfologia matemática que cada neurônio realiza. Inicialmente, temos que a configuração do perceptron morfológico é dada pela Figura 5.3, ou seja, a rede é formada por duas camadas. No que diz respeito à primeira camada, temos que o primeiro neurônio realiza uma *erosão*, e o segundo neurônio realiza uma *anti-dilatação*. Aplica-se a função de ativação identidade à ativação de cada neurônio, de modo que as entradas para o neurônio da segunda camada serão $y_1 = \bigwedge_{i=1}^{n}(x_i - {c_i}^\perp)$ e $y_2 = \bigwedge_{i=1}^{n}({c_i}^\top - x_i)$, e os pesos sinápticos associados à essas entradas são definidos como sendo iguais a zero. Esse neurônio calculará simplesmente uma *erosão* dada por:

$$\bigwedge_{i=1}^{2} y_i,$$

e, em seguida, aplicamos a função de ativação limiar g.

Um módulo do perceptron morfológico é uma configuração representada na Figura 5.3. Nas próximas iterações do algoritmo de treinamento, temos que outros pares de vetores de pesos sinápticos são produzidos.

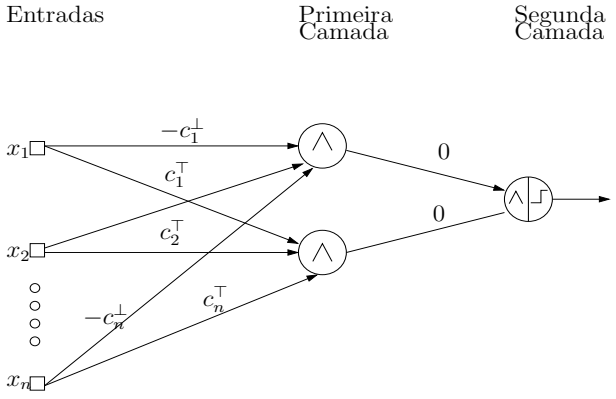

Fig. 5.3: Arquitetura do perceptron morfológico após o primeiro passo do algoritmo

Denotemos por $(\mathbf{c}^k)^\perp$, $(\mathbf{c}^k)^\top$ tais pesos, onde k denota que os vetores pesos se referem à k-ésima iteração. Podemos observar, pela Figura 5.6, que na medida em que o algoritmo vai

evoluindo a cada iteração, módulos são criados. Há o aparecimento de uma terceira camada com um único neurônio que tem o papel de calcular uma *dilatação* , seguida da aplicação de uma função limiar, que fornecerá a resposta da rede. Neste sentido, para K módulos produzidos pelo perceptron morfológico existem $3K + 1$ unidades de processamento. Podemos definir para o perceptron morfológico a noção de épocas como sendo o número de hipercaixas construídas durante o processo de treinamento. A justificativa para esta definição reside no fato que para cada módulo que vai ser construído devemos apresentar todos os padrões da classe C_1 que estão mal-classificados. Fornecemos um exemplo ilustrativo, representado nas figuras 5.4 e 5.5, da superfície de decisão obtida através deste algoritmo de aprendizagem do perceptron morfológico.

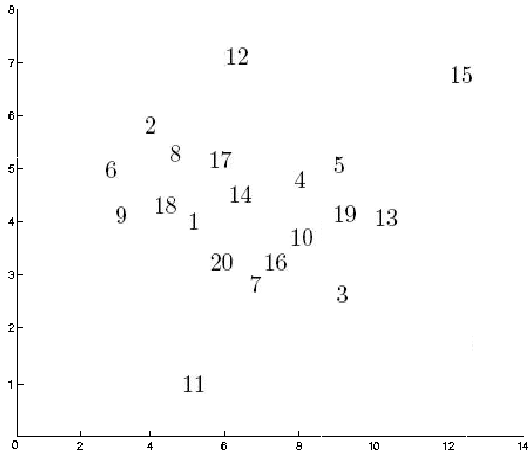

Fig. 5.4: Localização dos padrões de treinamento

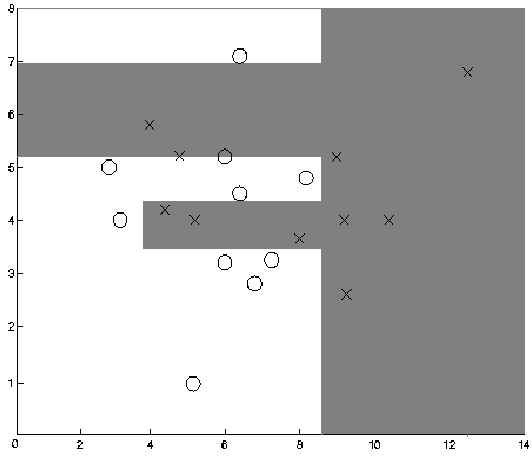

Fig. 5.5: Superfície de decisão obtida pelo perceptron morfológico

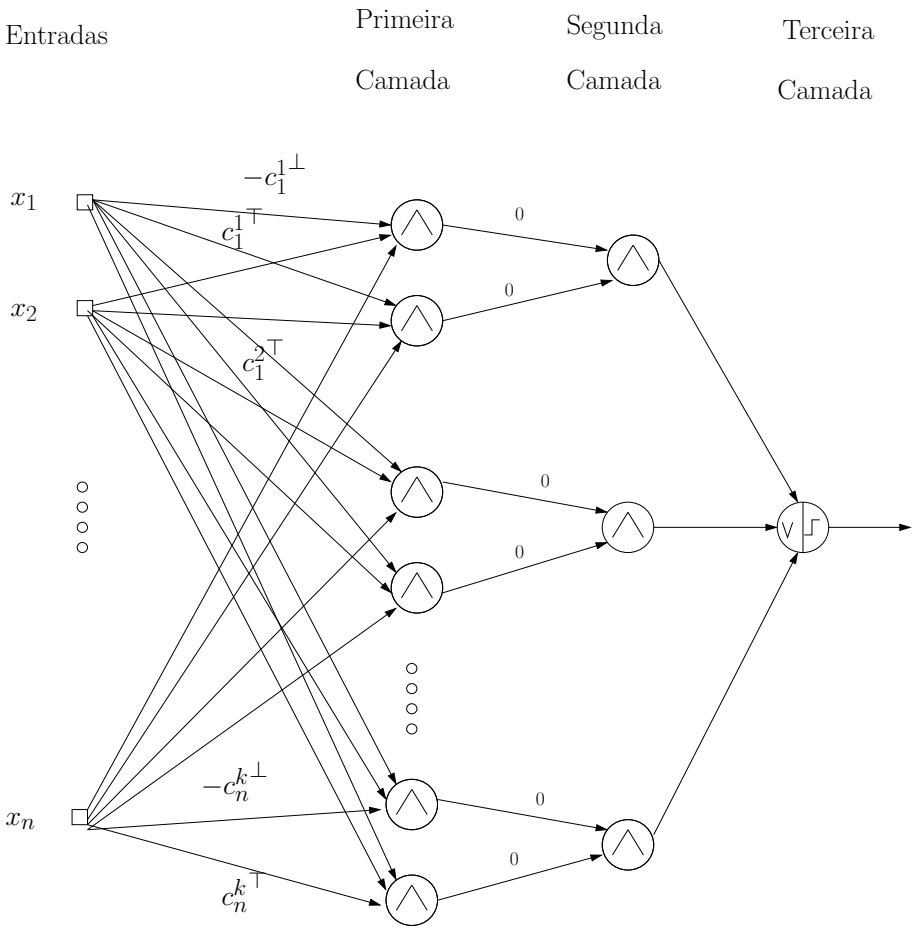

Fig. 5.6: Arquitetura do perceptron morfológico

5.3 Perceptrons Morfológicos com Dentritos (MPD)

Motivação Biológica

Pesquisas recentes em neurocomputação têm dado destaque à importância de estruturas dentríticas em uma célula nervosa. Estas unidades são agora vistas como unidades computacionais primária básicas do neurônio, capazes de realizar operações lógicas. Com respeito à esses novos modelos neurais, Ritter e Urcid desenvolveram um novo paradigma para a computação envolvendo neurônios morfológicos, onde o processo é realizado no dentrito [50], produzindo assim um modelo mais realista de um neurônio em redes neurais artificiais. Unidades computacionais e a topologia do modelo imitam neurônios biológicos do cortéx cerebral.

O neurônio, que é a unidade estrutural do sistema nervoso, é constituído de um corpo celular, chamado *soma*, e alguns processos. Os processos são de dois tipos, denominados *dentritos* e *axônios*, respectivamente. Os dentritos conduzem impulsos de entrada para o corpo da célula.

Em geral, os dentritos apresentam ramificações que criam árvores longas e complicadas. O axônio que normalmente fica no lado oposto da célula em um ponto chamado *tronco axonial*, consiste de uma fibra longa cujos ramos formam a árvore axonial. Eventualmente, em alguns neurônios, o axônio pode ter ramos em intervalos ao longo de seu comprimento, além de sua terminação arborizada. O axônio é o principal ramo fibral do neurônio para a transmissão de sinais a outros neurônios. A Figura 5.7 representa uma célula nervosa.

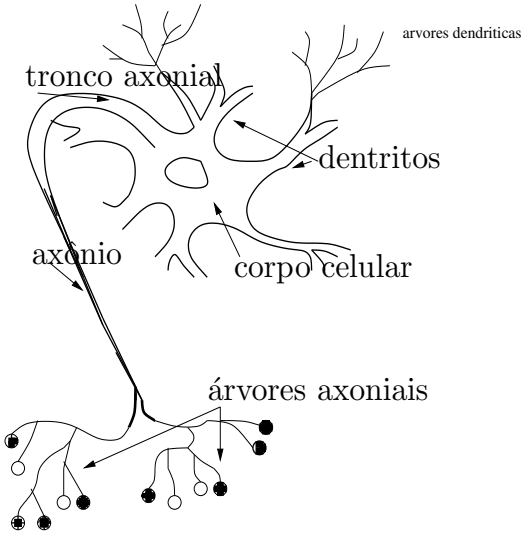

Fig. 5.7: Diagrama de uma célula nervosa

Um impulso percorre um axônio, através do tronco axonial, do neurônio para terminações nervosas. Tais terminações fazem contato com o corpo da célula e muitos dentritos de outras células nervosas. As regiões de contato entre axônio e dentrito recebem o nome de regiões sinápticas porque é o lugar onde as sinápses ocorrem. A *sinápse* é simplesmente uma estrutura especializada que fornece a comunicação entre neurônios. De fato o mecanismo exato de como a sinápse é realizada ainda é desconhecido, embora o que é bem conhecido é que existem dois tipos de sinápses: as *sinápses excitantes*, que tendem a despolarizar a membrana pós sináptica facilitando a ativação do neurônio, e as *sinápses inibitórias* que tentam prevenir a ativação final do neurônio.

O número de sinápses num neurônio, no córtex cerebral, está entre 500 e 200000. A maioria das sinápses ocorrem nos dentritos, sendo estes responsáveis por ocupar uma parcela significante de área e volume do cérebro. Pesquisas recentes revelaram que a informação transmitida é processada nos dentritos. Neste sentido, parece bem razoável considerar a atuação dos dentritos na neurocomputação.

Descrição detalhada do modelo

Neste modelo consideraremos que as informações serão processadas nos dentritos, ou seja, um neurônio morfológico N tem regiões pós-sinápticas dentríticas que recebem entrada de terminalizações axoniais de outros neurônios. Se N recebe entradas de n neurônios $N_1, N_2, \ldots, N_n$, então cada N_i tem ramos axoniais que desembocam em várias regiões sinápticas de N. Estas regiões são distribuídas ao longo de um número finito de dentritos $D_1, D_2, \ldots, D_k$. Considerando um neurônio N com K dentritos e suponhamos que cada dentrito recebe como entrada o vetor $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top$. O peso sináptico correspondente à entrada x_i do dentrito k é denotado por w^l_{ki}, onde $l \in \{0, 1\}$, e o índice l distingue simplesmente as sinápses excitantes das sinápses inibidoras. Ou seja, quando denotamos w^1_{ki}, significa que a entrada x_i produz uma excitação e w^0_{ki} determina uma inibição induzida pela entrada x_i.

A entrada x_i produz um efeito final no k-ésimo dentrito dado por $+(w^1_{ki} + x_i)$ para as sinapses excitantes e $-(w^0_{ki} + x_i)$ para as sinapses inibidoras. Notemos que, quanto maior for o valor de x_i, maior será o efeito de uma sinapse excitante e menor será o efeito de uma sinapse inibidora. Neste modelo, uma entrada é permitida produzir excitação e inibição no dentrito. O que ocorre no fim das contas é que uma competição entre o efeito de todas as conexões sinápticas é realizada e apenas a sinápse de menor valor é considerada como ativação do dentrito em questão.

A computação realizada pelo k-ésimo dentrito para uma entrada $\mathbf{x} \in \mathbb{R}^n$ é fornecida pela seguinte expressão:

$$\tau_k(\mathbf{x}) = p_k \bigwedge_{i=1}^{n} \bigwedge_{l \in L} (-1)^{l+1}(x_i + w^l_{ki}), \tag{5.21}$$

onde $L = \{0, 1\}$ e $p_k = \{-1, 1\}$ denota uma resposta excitante ou inibidora do k-ésimo dentrito. Ressaltamos que $w^1_{ki} = \infty$ e $w^0_{ki} = -\infty$ representam ausência de conexão sináptica excitante ou inibidora, respectivamente.

O valor $\tau_k(\mathbf{x})$ é transmitido para o corpo celular do neurônio e o estado do neurônio N é uma função das entradas recebidas de todos os dendritos. Ou seja,

$$\tau(\mathbf{x}) = \bigwedge_{k=1}^{K} \tau_k(\mathbf{x}), \tag{5.22}$$

na qual K denota o número de dentritos. Ou seja, a ativação do neurônio é considerada o menor valor dentre as excitações de todos os dentritos. O próximo estado do neurônio N, ou seja, a saída, é determinada por uma função de ativação aplicada á ativação do neurônio N, que adotaremos como sendo a função limiar definida na equação (4.4).

Desta maneira, o neurônio será ativado se, e somente se, todos os dentritos fornecerem saídas

positivas, isto é, se $\tau(\mathbf{x}) \geq 0$ para todo $k = 1, \ldots, K$. A saída de um neurônio morfológico com dentritos é dada por:

$$y(\mathbf{x}) = f\left(\bigwedge_{k=1}^{K}\left[p_k \bigwedge_{i=1}^{n}\bigwedge_{l\in L}(-1)^{l+1}(w_{ik}^{l} + x_i)\right]\right). \tag{5.23}$$

Vejamos como contar as unidades computacionais de um MPD. De acordo com a equação (5.21), temos que um k-ésimo dentrito calcula a seguinte operação:

$$\begin{aligned}
\tau_k(\mathbf{x}) &= p_k \bigwedge_{i=1}^{n}\bigwedge_{l\in\{0,1\}}(-1)^{l+1}(x_i + w_i^l);\\
&= p_k \bigwedge_{i=1}^{n}\{(-1)^{0+1}(x_i + w_i^0)\bigwedge(-1)^{1+1}(x_i + w_i^1)\};\\
&= p_k\{\{(-1)\cdot(x_1 + w_1^0)\bigwedge(x_1 + w_1^1)\}\bigwedge\ldots\bigwedge\{(-1)\cdot(x_n + w_n^0)\bigwedge(x_n + w_n^1)\}\};
\end{aligned}$$

Como o reticulado completo é distributivo, rearranjamos os termos e obtemos:

$$\tau_k(\mathbf{x}) = p_k\{\bigwedge_{i=1}^{n}(-1)(x_i + w_i^0)\}\bigwedge\{\bigwedge_{1}^{n}(x_i + w_i^1)\}.$$

Observamos que existem três unidades computacionais no k-ésimo dentrito; uma calcula a expressão $\bigwedge_{i=1}^{n}(-1)(x_i+w_i^0)$, outra calcula a expressão $\bigwedge_{1}^{n}(x_i+w_i^1)$, e a última calcula a expressão $\{\bigwedge_{i=1}^{n}(-1)(x_i+w_i^0)\}\bigwedge\{\bigwedge_{1}^{n}(x_i+w_i^1)\}$. Pela equação (5.23), temos que, por fim, um neurônio realiza uma competição entre as ativações produzidas pelos K dentritos. Neste sentido, teríamos no total $3K + 1$ unidades computacionais. A Figura 5.8 representa o modelo de um percetron morfológico com dentritos.

Algoritmo de treinamento do perceptron morfológico dendrítico

Nesta seção, apresentaremos um procedimento de aprendizagem construtivo para o treinamento de um *perceptron morfológico dentrítico*. Seja $T = \{(\mathbf{x}^\varepsilon, c_\varepsilon) : \varepsilon = 1, \ldots, P\}$ o conjunto de treinamento, onde $\mathbf{x}^n \in \mathbb{R}^n$ e $c_\varepsilon = \{0, 1\}$. Neste sentido, dizemos que $\mathbf{x}^\varepsilon \in C_1$ se $c_\varepsilon = 1$ e $\mathbf{x}^\varepsilon \in C_0$ se $c_\varepsilon = 0$. Pretendemos neste algoritmo encontrar pesos sinápticos apropriados $\mathbf{w}_k^1$ e $\mathbf{w}_k^0$ e p_k, para $k = 1, 2, \ldots, K$ de modo que a saída da rede, descrita pela equação (5.23) cumpra a igualdade $y(\mathbf{x}^\varepsilon) = c_\varepsilon$ para todo $\varepsilon = 1, \ldots, P$. Em outras palavras, estamos interessados em

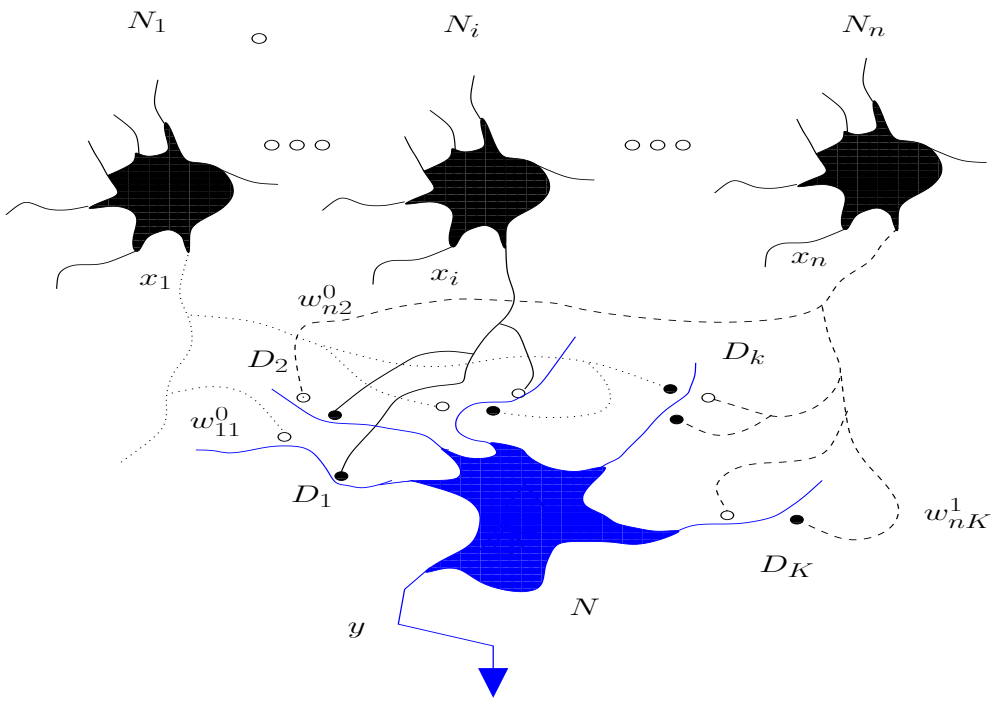

Fig. 5.8: Modelo do perceptron morfológico com dentritos

produzir *hipercaixas* da forma:

$$Q_k = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{c}_k^{\perp} \leq \mathbf{x} \leq \mathbf{c}_k^{\top}\},$$

tais que $\mathbf{x} \in \bigcap_{k=1}^{K} Q_k$, $\forall \mathbf{x}^{\varepsilon} \in C_1$ de maneira que possamos conseguir classificar todos os pontos pertencentes à classe C_1. Os pesos sinápticos estão associados aos cantos inferiores e superiores da hipercaixa.

Passo 1 Inicializamos o *perceptron morfológico dentrítico* com um único dentrito que é representado pela menor hipercaixa que contém todos os padrões da classe C_1.

$$p_1 = 1, \tag{5.24}$$

$$\mathbf{w}_1^1 = -\mathbf{c}_1^{\perp} = -\bigwedge_{\mathbf{x} \in C_1} \mathbf{x}, \tag{5.25}$$

$$\mathbf{w}_1^0 = -\mathbf{c}_1^{\top} = -\bigvee_{\mathbf{x} \in C_1} \mathbf{x}. \tag{5.26}$$

Selecionamos aleatoriamente um padrão $\mathbf{x} \in C_0$ mal classificado e adicionamos um novo dentrito que é representado pelo complementar do interior de uma hipercaixa contendo

$\mathbf{x}$. Assim, os pesos sinápticos do k-ésimo dentrito, representado pela caixa Q_k para $k \geq 2$, são obtidos da seguinte maneira:

(1) Escolha $\mathbf{x}^\gamma \in C_0$ tal que $y(\mathbf{x}^\gamma) = 1$.

(2) Defina o seguinte conjunto $D = \{\mathbf{x}^\varepsilon : \mathbf{x} \in C_1\}$.

(3) while $D \neq \emptyset$

(3.1) Calcule o mínimo da distância de *Chebyshev* do padrão mal-classificado selecionado $\mathbf{x}^\gamma$ à todos os padrões pertencentes a D.

$$\mu = \bigwedge_{i=1}^{n} \left\{ \bigvee_{i=1}^{n} |x_i^\gamma - x_i^\varepsilon| : \mathbf{x}^\varepsilon \in D \right\}.$$

(3.2) Encontre a coordenada i tal que

$$|x_i^\gamma - x_i^\varepsilon| = \mu, \mathbf{x} \in D, \varepsilon \neq \gamma.$$

(3.3) Atribuição dos pesos

$$w_{ik}^1 = -x_i^\varepsilon, \quad \text{se} \quad x_i^\varepsilon < x_i^\gamma. \tag{5.27}$$

$$w_{ik}^0 = -x_i^\varepsilon, \quad \text{se} \quad x_i^\varepsilon > x_i^\gamma. \tag{5.28}$$

(3.4) Atualização do conjunto D

$$D = \{\mathbf{x} \in D : -w_{ik}^1 < x_i^\varepsilon \quad \text{e} \quad x_i^\varepsilon < -w_{ik}^0\}. \tag{5.29}$$

O procedimento de aprendizagem do perceptron morfológico dentrítico demonstra a sua capacidade computacional para solucionar problemas não lineares, produzindo dentritos necessários, sem problemas de convergência ou necessidades de camadas escondidas. Definimos a noção de épocas para o perceptron morfológico com dentritos como sendo o número de dentritos produzidos durante o processo de treinamento. Apresentamos na Figura 5.9 a superfície de decisão obtida por um MPD com respeito ao exemplo da Figura 5.4.

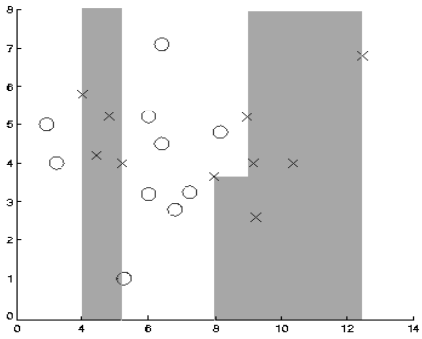

Fig. 5.9: Superfície de decisão obtida pelo perceptron morfológico dentrítico

5.4 Fuzzy Lattice Neural Network-FLNN

A fundamentação teórica da rede neural FLNN constitui uma combinação bem sucedida de conjuntos fuzzy [29] e teoria de reticulados [6] com teoria de ressonância adaptativa (ART) [10]. Dois paradigmas de aprendizagem são implementados pela rede FLNN, sendo um esquema para clusterização e outro para classificação. No processo de aprendizagem, conjuntos de elementos de reticulados são especificados por conjuntos de intervalos de reticulados.

5.4.1 Reticulado dos intervalos generalizados

Seja $\mathbb{L}$ um reticulado completo e $\boldsymbol{O}_{\mathbb{L}}$ e $\boldsymbol{I}_{\mathbb{L}}$, o menor e o maior elemento de $\mathbb{L}$, respectivamente. O reticulado completo $\mathbb{PL}$, chamado de reticulado dos intervalos generalizados de $\mathbb{L}$, é dado por

$$\mathbb{PL} = \{[a, b] : a, b \in \mathbb{L}\}.$$

Com a relação de ordem parcial dada por $[a, b] \leq [c, d] \Leftrightarrow c \leq a$ e $b \leq d$. Seja x^i uma coleção de elementos de $\mathbb{PL}$. Então o *ínfimo* e o *supremo* de elementos deste reticulado $\mathbb{PL}$ são naturalmente dados, respectivamente, por:

$$\wedge_{i\in I}\mathrm{x}^i = \wedge_{i\in I}[a^i, b^i] = [\vee_{i\in I}a^i, \wedge_{i\in I}b^i]. \tag{5.30}$$

$$\vee_{i\in I}\mathrm{x}^i = \vee_{i\in I}[a^i, b^i] = [\wedge_{i\in I}a^i, \vee_{i\in I}b^i]. \tag{5.31}$$

Um elemento de $\mathbb{PL}$ é chamado de intervalo generalizado. Definimos o reticulado completo $\mathbb{V}_{\mathbb{L}}$ como o conjunto de todos os intervalos fechados de elementos de $\mathbb{L}$ aumentado pelo elemento

mínimo $[\boldsymbol{I}_{\mathbb{L}}, \boldsymbol{O}_{\mathbb{L}}]$ de $\mathbb{PL}$:

$$\mathbb{V}_{\mathbb{L}} = \{[a,b] : a,b \in \mathbb{L} \quad \text{e} \quad a \leq b\} \cup \{[\boldsymbol{I}_{\mathbb{L}}, \boldsymbol{O}_{\mathbb{L}}]\}.$$

Podemos definir as operações de *ínfimo* e *supremo* no reticulado completo $\mathbb{V}_{\mathbb{L}}^N = (\mathbb{V}_{\mathbb{L}})^N$ da seguinte maneira: Seja $\mathbf{x}^i = (\mathrm{x}_1^i, \mathrm{x}_2^i, \ldots, \mathrm{x}_N^i)$ uma coleção de elementos de $\mathbb{V}_{\mathbb{L}}^N$. Então o *ínfimo* e o *supremo* de $\mathbf{x}^i$ em $\mathbb{V}_{\mathbb{L}}^N$ são dados, respectivamente, por:

$$\wedge_{i\in I}\mathbf{x}^i = (\wedge_{i\in I}\mathrm{x}_1^i, \wedge_{i\in I}\mathrm{x}_2^i, \ldots, \wedge_{i\in I}\mathrm{x}_N^i). \tag{5.32}$$

$$\vee_{i\in I}\mathbf{x}^i = (\vee_{i\in I}\mathrm{x}_1^i, \vee_{i\in I}\mathrm{x}_2^i, \ldots, \vee_{i\in I}\mathrm{x}_N^i). \tag{5.33}$$

5.4.2 Conjuntos Fuzzy

Um conjunto nebuloso $\mathbf{a}$ é definido por meio de uma função, chamada de *função de pertinência*, de um conjunto clássico $\mathbf{X}$ para o intervalo $[0,1]$:

$$\mathbf{a}(\mathbf{x}) : \mathbf{X} \longrightarrow [0,1] \tag{5.34}$$

De fato esta função caracteriza o grau de pertinência de $\mathbf{x}$ em $\mathbf{a}$. Todo conjunto nebuloso fica unicamente determinado pela sua função de pertinência $\mathbf{a}(\mathbf{x})$. A classe dos conjuntos nebulosos é dada por:

$$\mathcal{F}(\mathbf{X}) = \{\mathbf{a} : \mathbf{X} \longrightarrow [0,1]\}$$

Um conjunto clássico $\mathbf{A} \in \mathcal{P}(\mathbf{X})$ se relaciona com um conjunto nebuloso $\mathbf{a} \in \mathcal{F}(\mathbf{X})$ da seguinte forma:

$$\mathbf{a}(\mathbf{x}) = \begin{cases} 1, & \text{se} \quad \mathbf{x} \in \mathbf{A} \\ 0, & \text{se} \quad \mathbf{x} \notin \mathbf{A} \end{cases}$$

5.4.3 Reticulados Fuzzy

A noção de *reticulado fuzzy* foi introduzida para extender a *relação de ordem parcial* a todos pares $(\mathbf{x}, \mathbf{y}) \in \mathbb{L} \times \mathbb{L}$ de um reticulado convencional $\mathbb{L}$. Um *reticulado fuzzy* é um par $(\mathbb{L}, p)$, onde $\mathbb{L}$ é um reticulado convencional e $p : \mathbf{X} \longrightarrow [0,1]$ é uma função de ordem parcial nebulosa no universo de discurso $\mathbf{X} = \{(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathbb{L}\}$, tal que:

$$p(\mathbf{x}, \mathbf{y}) = 1 \Leftrightarrow \mathbf{x} \leq \mathbf{y} \quad \text{em} \quad \mathbb{L}$$

Uma função $h : \mathbb{L} \longrightarrow \mathbb{R}$ é dita ser uma função de *avaliação positiva* se satisfaz as seguintes propriedades:

$(P1)$ $h(O) = 0$, onde O é o menor elemento em $\mathbb{L}$

$(P2)$ $\mathbf{u} < \mathbf{w} \Rightarrow h(\mathbf{u}) < h(\mathbf{w}), \quad \mathbf{u}, \mathbf{w} \in \mathbb{L}.$

$(P3)$ $\mathbf{u} \leq \mathbf{w} \Rightarrow h(\mathbf{x} \vee \mathbf{w}) - h(\mathbf{x} \vee \mathbf{u}) \leq h(\mathbf{w}) - h(\mathbf{u}), \quad \mathbf{x}, \mathbf{u}, \mathbf{w} \in \mathbb{L}.$

Uma *função de avaliação positiva* não necessariamente existe em um reticulado $\mathbb{L}$. No entanto, quando esta função existe pode-se mostrar que $p(\mathbf{x}, \mathbf{y}) = h(\mathbf{y})/h(\mathbf{x} \vee \mathbf{y}), \quad \mathbf{x}, \mathbf{y} \in \mathbb{L}$, sendo que $h(.)$ é uma *função de avaliação positiva*. Tal função define uma ordem parcial nebulosa em $\mathbb{L}$, pois $p(\mathbf{x}, \mathbf{y}) = h(\mathbf{y})/h(\mathbf{x} \vee \mathbf{y}) = 1 \Leftrightarrow h(\mathbf{y}) = h(\mathbf{x} \vee \mathbf{y}) \Leftrightarrow \mathbf{y} = \mathbf{x} \vee \mathbf{y} \Leftrightarrow \mathbf{x} \leq \mathbf{y}$ [40]. Desta maneira, $(\mathbb{L}, p(\mathbf{x}, \mathbf{y}))$ é um reticulado fuzzy. Neste caso, a função p satisfaz as seguintes propriedades:

$(C1)$ $p(\mathbf{u}, O) = 0, \quad \mathbf{u} \neq O;$

$(C2)$ $p(\mathbf{u}, \mathbf{u}) = 1, \quad \forall \mathbf{u} \in \mathbb{L};$

$(C3)$ $\mathbf{u} \leq \mathbf{w} \Rightarrow p(\mathbf{x}, \mathbf{u}) \leq p(\mathbf{x}, \mathbf{w}), \quad \mathbf{x}, \mathbf{u}, \mathbf{w} \in \mathbb{L}.$

A função $p(\mathbf{x}, \mathbf{y})$ será usada como função de agregação dos neurônios da rede neural FLNN. De fato para calcularmos a função de ordem parcial nebulosa $p(\mathbf{x}, \mathbf{u})$ de um itervalo de reticulados $\mathbf{x}$ em outro intervalo de reticulados $\mathbf{u}$, primeiro mapeamos $\mathbf{x}, \mathbf{u} \in {\mathbb{V}_{\mathbb{L}}}^N \subseteq {\mathbb{PL}}^N$ a seus isomorfos $\mathbf{x}', \mathbf{u}' \in \mathbb{L}^N$.

O seguinte lema garante a existência de uma *função de avaliação positiva* no produto de reticulados fuzzy desde que todos os reticulados fuzzy constituintes possuam uma função de avaliação positiva.

Lema 1 *Sejam $\mathbb{L} = \mathbb{L}_1 \times \mathbb{L}_2 \times \ldots \times \mathbb{L}_N$ o produto de N reticulados completos, cada qual com as suas respectivas funções de avaliação positiva $h_1, h_2, \ldots, h_n$. Logo a função $h((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)) = h_1(\boldsymbol{x}_1) + \ldots + h_N(\boldsymbol{x}_N)$ define uma função de avaliação positiva no produto de reticulados $\mathbb{L}$ [40].*

Uma função de avaliação positiva qualquer em $\mathbb{V}_{\mathbb{L}}$ define uma função de ordem parcial em $\mathbb{V}_{\mathbb{L}}$. Então, para se obter uma função de ordem parcial nebulosa em $\mathbb{V}_{\mathbb{L}}$ é preciso primeiramente estabelecer um isomorfismo dual entre $\mathbb{PL}$ e $\mathbb{L}^2$, sendo que para construir este isomorfismo é suficiente que exista um automorfismo dual $\theta : \mathbb{L}^2 \longrightarrow \mathbb{L}^2$. Consideremos o isomorfismo:

$$\begin{array}{rccc} \varphi_\theta & : & \mathbb{PL} & \longrightarrow & \mathbb{L}^2 \\ & & [a, b] & \longmapsto & (\theta(a), b) \end{array} \tag{5.35}$$

Assim existe um isomorfismo entre $\mathbb{PL}$ e $\mathbb{L}^2$. Uma função de ordem parcial nebulosa pode ser definida no reticulado $\mathbb{V}_{\mathbb{L}}$ da seguinte maneira:

1 Uma função de avaliação positiva h no reticulado $\mathbb{L}$ pode definir uma função de avaliação positiva no reticulado $\mathbb{L}^2$ de acordo com o lema 1.

2 Uma função de avaliação positiva h em $\mathbb{L}^2$ define uma função de ordem parcial nebulosa $p_h(\mathbf{x}, \mathbf{y}) = \frac{h(\mathbf{y})}{h(\mathbf{x} \vee \mathbf{y})}$ em $\mathbb{L}^2$.

3 Um automorfismo dual $\theta(.)$ em $\mathbb{L}$ pode definir um isomorfismo φ_θ entre $\mathbb{PL}$ e $\mathbb{L}^2$.

4 Uma função de ordem parcial nebulosa entre dois intervalos $[a, b], [c, d] \in \mathbb{V}_\mathbb{L}$ é dada por:

$$p([a, b], [c, d]) = p_h(\varphi_\theta[a, b], \varphi_\theta[c, d]), \tag{5.36}$$

$$= p_h((\theta(a), b), (\theta(c), d)), \tag{5.37}$$

onde $(\theta(a), b), (\theta(c), d) \in \mathbb{L}^2$ e p_h é a função de ordem parcial nebulosa em $\mathbb{L}^2$ dada por:

$$p_h(\mathbf{x}, \mathbf{y}) = \frac{h(\mathbf{y})}{h(\mathbf{x} \vee \mathbf{y})} \tag{5.38}$$

Então sempre podemos associar um elemento $\mathbf{x} \in \mathbb{L}^2$ com um intervalo $[a, b]$ de $\mathbb{V}_\mathbb{L}$ pelo fato da existência deste automorfismo θ e do isomorfismo φ_θ entre $\mathbb{PL}$ e $\mathbb{L}$. A função de ordem parcial nebulosa $p : \mathbb{V}_\mathbb{L} \longrightarrow [0, 1]$ pode representar uma *anti-erosão* do reticulado completo $\mathbb{V}_\mathbb{L}$ para o reticulado completo $[0, 1]$ ou pode representar também uma *anti-dilatação* [59] do reticulado completo $\mathbb{V}_\mathbb{L}$ para o reticulado completo $[0, 1]$.

Uma ordem parcial nebulosa p em ${\mathbb{V}_\mathbb{L}}^N$ é obtida por meio do isomorfismo ${\varphi_\theta}^N : (\mathbb{PL})^N \longrightarrow (\mathbb{L}^2)^N$. Ou seja, para mapear um elemento $\mathbf{a} \in {\mathbb{V}_\mathbb{L}}^N$ para $(\mathbb{L}^2)^N$, basta aplicar ${\varphi_\theta}^N(\mathbf{a}) = {\varphi_\theta}^N(\mathbf{a}_1, \ldots, \mathbf{a}_N) = (\varphi_\theta(\mathbf{a}_1, \ldots, \varphi_\theta(\mathbf{a}_N))$, onde $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$.

5.4.4 Arquitetura da rede neural FLNN

A rede neural FLNN opera de modo semelhante a rede neural ART [10], ou seja, a rede FLNN consiste de uma arquitetura recorrente em duas camadas: a camada de entrada e a camada de categoria, de acordo com a Figura 5.10. A camada de entrada possui N neurônios artificiais que são usados para armazenar e comparar dados de entrada. A camada de categoria possui L neurônios artificiais, que definem M classes. Cada neurônio da camada de categoria pode definir no máximo uma classe, em outras palavras, $L \geq M$.

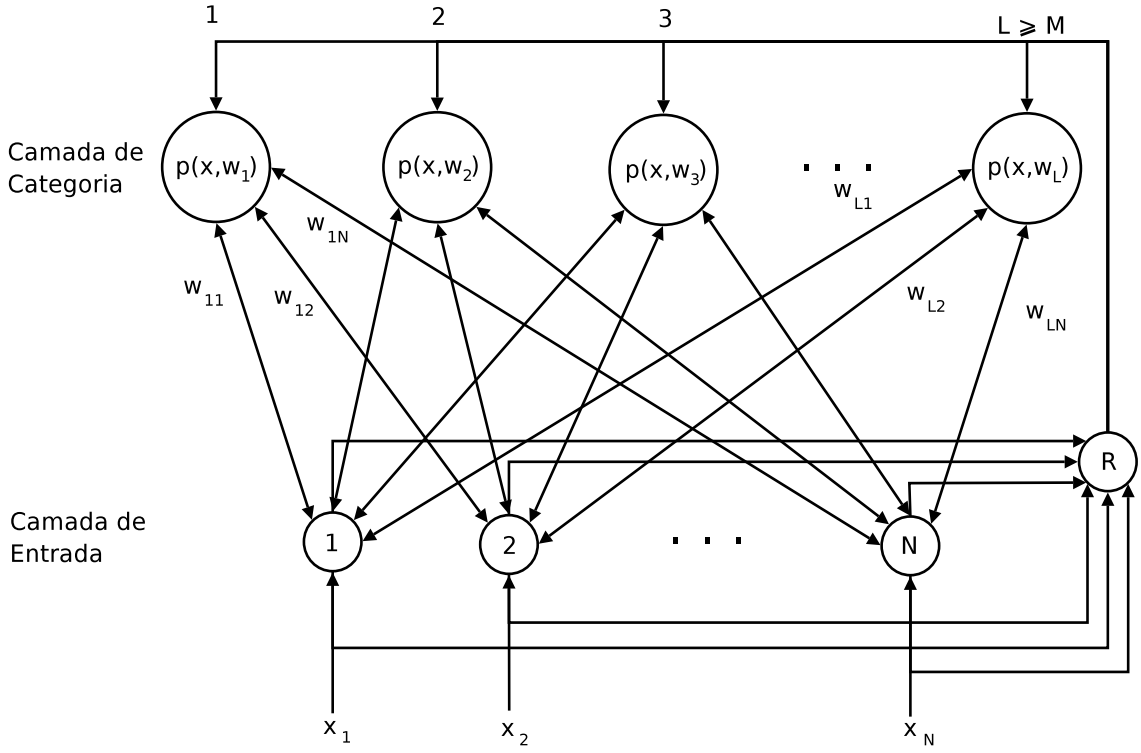

Fig. 5.10: Arquitetura da rede neural FLNN

As duas camadas são completamente conectadas pelo vetor de pesos sinápticos $\mathbf{w}$, que tem a finalidade de filtrar os dados tanto de baixo para cima (pesos "bottom-up"), como de cima para baixo (pesos "top-down"). Também existe um nó utilizado para desabilitar as classes que opera de modo semelhante à rede neural ART. A entrada da rede FLNN deve ser uma hipercaixa $\mathbf{x}$ pertencente a $(\mathbb{V}_{\mathbb{L}})^N$, ao invés de vetores de $\mathbb{R}^N$, ou seja, cada componente x_j, para $j = \{1, \ldots, N\}$, representa um intervalo em $\mathbb{V}_{\mathbb{L}}$. O tamanho de uma hipercaixa arbitrária $\mathbf{x} = ([a_1, b_1], \ldots, [a_N, b_N])$, denotado por $Z(\mathbf{x})$, é definido por $Z(\mathbf{x}) = \prod_{i=1}^{N}(h(b_i) - h(a_i))$, onde $h_1, \ldots, h_N$ são funções de avaliação positiva definidas em $\mathbb{L}$. A função de agregação empregada pelos neurônios da camada superior da rede FLNN, é uma função de ordem parcial nebulosa, denotada por p, que calcula o valor de pertinência da entrada $\mathbf{x}$ no intervalo de pesos aprendidos $\mathbf{w}_i$, os quais ligam os neurônios da camada de entrada ao i-ésimo neurônio da camada de categoria, onde $i = \{1, \ldots, N\}$. A rede neural FLNN lida com certas famílias de intervalos de reticulados, denotadas por $\{\mathbf{w}_i\}$ onde $\mathbf{w}_i \in {\mathbb{V}_{\mathbb{L}}}^N$ e $i \in I$, I finito. O objetivo da FLNN é identificar conjuntos c de elementos de reticulados, chamados de classe que podem ser representados por uma união de um número finito de intervalos de $(\mathbb{V}_{\mathbb{L}})^N$, isto é:

$$c = \cup_i \mathbf{w}_i.$$

A aplicabilidade da rede neural FLNN depende fundamentalmente da existência de uma função de ordem parcial nebulosa em $(\mathbb{V}_{\mathbb{L}})^N$. Se $\mathbf{x} \in (\mathbb{V}_{\mathbb{L}})^N$ e $c = \cup_i \mathbf{w}_i$ então o grau de inclusão de $\mathbf{x}$ na classe c é definido como $p(\mathbf{x}, c) := \max(p(\mathbf{x}, \mathbf{w}_i))$.

A tomada de decisão na rede neural FLNN é conseguida através do grau de verdade da proposição "$\mathbf{x} \leq c$" como expressado por $p(\mathbf{x}, c)$, onde $\mathbf{x}$ é um intervalo de reticulado de entrada que excita o sistema e c representa a classe aprendida guardada na memória do sistema. Quando $p(\mathbf{x}, c) = 1$ nós escreveremos $\mathbf{x} \leq c$ e diremos que $\mathbf{x}$ está dentro da classe c.

5.4.5 Algoritmo de aprendizagem supervisionado

Em particular um conjunto de treinamento é empregado explicitamente, consistindo de P pares $(\mathbf{x}_i, g_i)$ $i \in \{1, \ldots, P\}$, onde $\mathbf{x}_i \in \mathbb{L}$ é um padrão de entrada e g_i é um índice para a categoria de $\mathbf{x}_i$, com valores no conjunto $\{1, \ldots, k\}$ sendo que k é o número de categorias.

A arquitetura da rede FLNN [18] apresentada na Figura 5.9 precisa acomodar a informação que diz respeito ao índice g_i de uma categoria. Isto é conseguido fazendo-se a seguinte consideração: primeiro permite-se que o ínice g_i seja guardado na camada de categoria, e em seguida adiciona-se um nó na camada de entrada para acomodar o índice de categoria g_i. Com esta nova acomodação, o novo nó da camada de entrada deve se interconectar a todos os nós da camada de categoria.

O algoritmo de aprendizagem da rede FLNN [18, 40, 70] basicamente tem a finalidade de agrupar todas as hipercaixas de entrada que possuam a mesma categoria g_j. Essas hipercaixas que agrupam os elementos são adicionadas na camada de categoria. A seguir, apresentamos o algoritmo de aprendizagem supervisionado propriamente dito:

```
    1 Algoritmo
1 Para i = 1,...N faça
2     x = x_i.
3    Para j = 1,...,N faça
4         se g_i = g_j então
5           x' = x ∨ x_j.
6           Para k = 1,...,N, faça
7                 Se g_k =≠ g_i e p(x_k, x') < 1, então
8                 x = x'.
9              fim-para
10          fim-se
11     fim-para
```

12 armazena$(\mathbf{x}, g_i)$
13 fim-para.

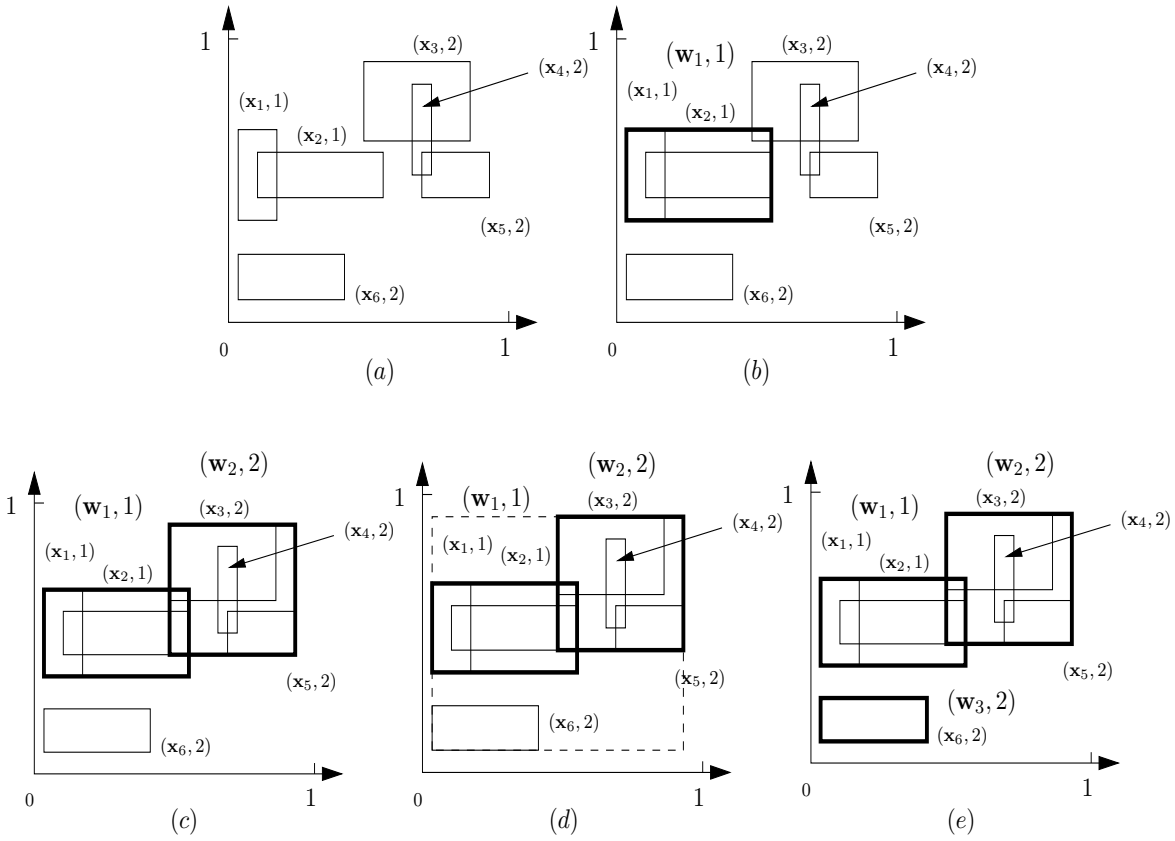

Fig. 5.11: Exemplo do algoritmo supervisionado para 6 hipercaixas de entrada

No exemplo da Figura 5.11, em (a) são apresentadas as hipercaixas de entrada pertencentes a $(\mathbb{V}_{\mathbb{L}})^2$, onde $\mathbb{L} = [0, 1]$ e seus respectivos índices. Na parte (b) agrupam-se as hipercaixas $(\mathbf{x}_1, 1)$ e $(\mathbf{x}_2, 1)$, e o resultado final é representado na camada de categoria por $(\mathbf{w}_1, 1)$. Na parte (c) agrupam-se as caixas $(\mathbf{x}_3, 2)$,$(\mathbf{x}_4, 2)$ e $(\mathbf{x}_5, 2)$ e representa-se este agrupamento por $(\mathbf{w}_2, 2)$ na camada de categoria. Por fim, não foi possível a expansão com a hipercaixa $(\mathbf{x}_6, 2)$ porque $(\mathbf{x}_1, 1)$ e $(\mathbf{x}_2, 1)$, que pertencem à outra classe, ficariam completamente incluídas no resultado, conforme pode ser visto em (d). Então representa-se $(\mathbf{x}_6, 2)$ na camada de categoria por $(\mathbf{w}_3, 2)$. A função de ordem parcial nebulosa considerada é $p(\mathbf{x}, \mathbf{y}) = h(\mathbf{y})/h(\mathbf{x} \vee \mathbf{y})$, onde $\mathbf{x}, \mathbf{y} \in (\mathbb{V}_{\mathbb{L}})^2$ e $h : [0, 1] \longrightarrow [0, 1]$ é uma função de avaliação positiva dada por $h(x) = x$ e $\theta : [0, 1] \longrightarrow [0, 1]$ tal que $\theta(x) = 1 - x$ é a função de automorfismo dual considerada. Podemos definir o número de épocas de uma rede FLNN como sendo três vezes o número de hipercaixas construídas durante o treinamento, visto que, segundo o algoritmo de treinamento, são necessários três loops para uma execução do algoritmo. De fato, quando uma hipercaixa é construída, deve-se apresentar novamente todos os padrões para se determinar a nova hipercaixa que representará uma nova

categoria. Temos que os próprios neurônios da camada de categoria da rede FLNN representam as unidades de processamento da rede, uma vez que cada caixa determina os pesos que serão combinados pela função de agregação nebulosa p do neurônio da FLNN.

5.5 Rede Neural Linear/Posto/Morfológica híbrida

O modelo de rede neural Linear/Posto/Morfológica Híbrida (MRL) [38] pertence à classe das redes neurais alimentadas adiante, e incorpora propriedades dos perceptrons de múltiplas camadas (MLPs) [61] e das redes neurais posto morfológicas (MRNNs). Em outras palavras, cada neurônio realiza uma operação que consiste em uma combinação híbrida de operações lineares e não-lineares seguida da aplicação de uma função de ativação. Em termos matemáticos, descrevemos a rede MRL por meio das equações (4.6) e (4.7) tais que

$$x_k^{(l)} = \lambda_k^{(l)}\alpha_k^{(l)} + (1-\lambda_k^{(l)})\beta_k^{(l)}, \tag{5.39}$$

$$\alpha_k^{(l)} = \mathcal{R}_{r_k^{(l)}}(\mathbf{y}^{(l-1)} + \mathbf{a}_k^{(l)}), \tag{5.40}$$

$$\beta_k^{(l)} = {\mathbf{y}^{(l-1)}}^\top \mathbf{b}_k^{(l)} + \theta_k^{(l)}, \tag{5.41}$$

onde $r_k^{(l)} \in \mathbb{Z}$, $\lambda_k^{(l)}, \theta_k^{(l)} \in \mathbb{R}; \mathbf{a}_k^{(l)}, \mathbf{b}_k^{(l)} \in \mathbb{R}^{N_{l-1}}$. O símbolo $\mathcal{R}_r$ representa a função posto [39], de ordem r, que determina o r-ésimo elemento da sequência obtida colocando-se as componentes do vetor $\mathbf{t} = (t_1, \ldots, t_n) \in \mathbb{R}^n$ dado em ordem decrescente, ou seja, considere a sequência em ordem decrescente:

$$t_{(1)} \geq t_{(2)} \geq \ldots \geq t_{(n)},$$

Tomando o r-ésimo elemento desta sequência, definimos a r-ésima função posto de $\mathbf{t}$ por

$$\mathcal{R}_r(\mathbf{t}) = t_{(r)}, \quad \text{onde} \quad r = 1, 2, \ldots, k. \tag{5.42}$$

O vetor peso associado a cada neurônio é definido por $\mathbf{w}_k^{(l)} = (\mathbf{a}_k^{(l)}, \rho_k^{(l)}, \mathbf{b}_k^l, \theta_k^{(l)}, \lambda_k^{(l)})$ em $\mathbb{R}^{2N_{l-1}+3}$. Usa-se uma variável real $\rho_k^{(l)}$ ao invés de uma variável inteira $r_k^{(l)}$, devido ao fato de se precisar avaliar *derivadas posto* [39] durante o algoritmo de treinamento da rede MRL. A relação entre $\rho_k^{(l)}$ e $r_k^{(l)}$ é dada por:

$$r_k^{(l)} = \left\lfloor N_{l-1} - \frac{N_{l-1}-1}{1+exp(-\rho_k^{(l)})} + 0.5 \right\rfloor, \tag{5.43}$$

onde $\lfloor \cdot \rfloor$ denota a função maior inteiro. Um módulo da rede MRL é representado na Figura 5.12. Notemos que o cálculo realizado na equação (5.39) não consegue ser realizado por um único neurônio. Isto se deve ao fato da variável $r_k^{(l)}$ sempre variar. Neste sentido, em um

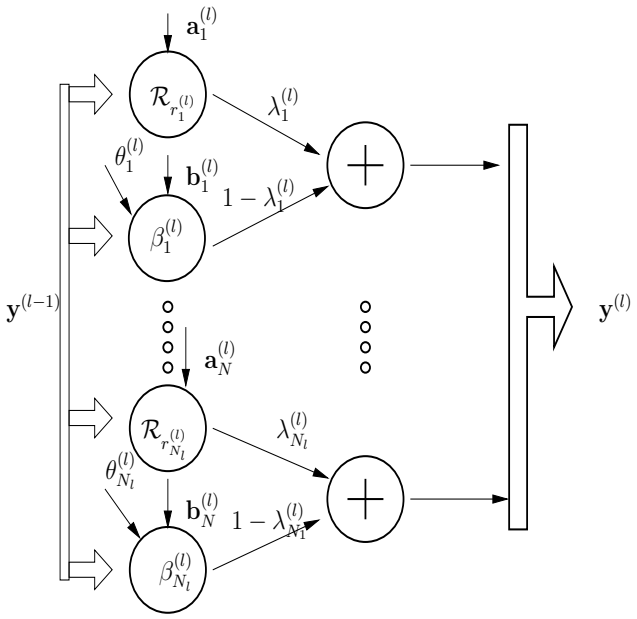

Fig. 5.12: Estrutura da l-ésima camada de uma rede MRL

neurônio é responsável pela soma vetorial $\mathbf{y}^{(l-1)} + \mathbf{a}_k^{(l)}$, e outro é responsável por tomar a $r_k^{(l)}$ posição do vetor resultante da soma. Neste sentido, em um módulo da rede MRL, representado na Figura 5.12, temos que existem quatro unidades computacionais ao invés de três (*impressão equivocada que a Figura* 5.12 *pode passar*). Dois casos especiais da rede MRL são obtidos quando a função de ativação g, empregada pela rede MRL de acordo com as equações (4.6) e (4.7), é a identidade, ou seja, $g(x_k^{(l)}) = x_k^{(l)}$, e quando a função g é uma não linearidade do tipo logística, ou seja, $g(x_k^{(l)}) = (1 + exp(-\eta x_k^{(l)}))^{-1}$, $\eta \geq 1$, de acordo com [38]. Primeiramente, antes de definirmos a derivada da função posto, iremos definir a função impulso. A função impulso é dada por:

$$Q(\mathbf{v}) = [q(v_1), \ldots, q(v_n)], \quad \text{tal que} \quad \mathbf{v} = (v_1, \ldots, v_n), \tag{5.44}$$

onde

$$q(v) = \begin{cases} 1, & \text{se} \quad v = 0 \\ 0, & \text{caso contrário} \end{cases} \tag{5.45}$$

Definição 1 *Dado um vetor* $\boldsymbol{t} = (t_1, t_2, \ldots, t_n) \in \mathbb{R}^n$*, e um posto* $r \in \{1, 2, \ldots, n\}$*, o r-ésimo vetor indicador posto* $\boldsymbol{c}$ *de* $\boldsymbol{t}$ *é definido por*

$$\boldsymbol{c}(\boldsymbol{t}, r) = \frac{Q(z\mathbf{1} - \boldsymbol{t})}{Q(z\mathbf{1} - \boldsymbol{t}) \cdot \mathbf{1}^\top}, \qquad \textit{onde} \quad z = \mathcal{R}_r(\boldsymbol{t}) \quad e \quad \mathbf{1} = (1, 1, \ldots, 1).$$

O *vetor indicador posto* tem o papel de marcar as localizações em $\mathbf{t}$ onde $z = \mathbb{R}_r(\mathbf{t})$ ocorrem. Para evitar mudanças abruptas e conseguir robustez numérica, a função $q(v)$ é, normalmente, substituída por funções impulso suaves [39] $q_\sigma(v)$, $\sigma \geqslant 0$ como, por exemplo, $exp[-\frac{1}{2}(v/\sigma)^2]$ ou $sech^2(v/\sigma)$.

Proposição 2 *Seja* $\boldsymbol{t} \in \mathbb{R}^n$, $r \in \{1, 2, \ldots, n\}$ *e* $\boldsymbol{c} = \boldsymbol{c}(\boldsymbol{t}, r)$*.Então,*

(*a*) $\boldsymbol{c} \cdot \mathbf{1}^\top = 1$

(*b*) $\boldsymbol{c} \cdot \boldsymbol{t}^\top = \mathcal{R}_r(\boldsymbol{t})$

(*c*) *Se* r *é fixo, então* $\boldsymbol{c}$ *é uma função constante por partes de* $\boldsymbol{t}$ *com exatamente* $2^n - 1$ *possíveis diferentes valores. Para todos os pontos* $\boldsymbol{t}_0 \in \mathbb{R}^n$ *com componentes diferentes, existe uma vizinhança em torno deles,* $\|\boldsymbol{t} - \boldsymbol{t}_0\|_\infty < \frac{1}{2} min_{i \neq j} |t_{0i} - t_{0j}|$*, tal que dentro dela* $\boldsymbol{c}$ *é constante, ou seja,* $\boldsymbol{c}(\boldsymbol{t}, r) = \boldsymbol{c}(\boldsymbol{t}_0, r)$*.*

De acordo com a definição e a proposição [39] apresentadas acima, se considerarmos que, para r fixo, $\mathbf{c} = \mathbf{c}(\mathbf{t}, r)$ seja constante em uma certa vizinhança de t_0, então a função posto $\mathcal{R}_r(\mathbf{t})$ é diferenciável em $\mathbf{t}_0$ e sua derivada é dada por

$$\nabla \mathcal{R}_r(\mathbf{t}) \mid_{\mathbf{t}=\mathbf{t}_0} = \mathbf{c}(\mathbf{t}_0, r). \tag{5.46}$$

Nos pontos em cuja vizinhança $\mathbf{c}$ não seja constante, a *função posto* não é diferenciável. De fato, se $\mathbf{t}$ satisfaz as hipóteses da proposição $2(c)$, temos que $\mathbf{t}$ e $\mathbf{t}_0$ apresentam a mesma ordem decrescente de suas componentes, consequentemente, como o posto r é fixo, o *vetor indicador posto* $\mathbf{c}$ de $\mathbf{t}$ e $\mathbf{t}_0$ serão os mesmos, pois o *vetor indicador posto* simplesmente marca as localizações em $\mathbf{t}$ onde $z = \mathcal{R}_r(\mathbf{t})$ ocorrem. Em outras palavras, teremos que $\mathbf{c}(\mathbf{t}, r) = \mathbf{c}(\mathbf{t}_0, r)$. Neste sentido, considerando $\mathbf{c}(\mathbf{t}, r)$ constante em uma vizinhança de $\mathbf{t}_0$, e usando a representação de produto interno para a função posto $\mathcal{R}_r(\mathbf{t})$,(*Proposição* $2(c)$), para todo $\mathbf{t}$ dentro da vizinhança, temos que

$$\nabla \mathcal{R}_r(\mathbf{t}) \mid_{\mathbf{t}=\mathbf{t}_0} = \nabla(\mathbf{c}(\mathbf{t}_0, r) \cdot \mathbf{t}^\top) \mid_{\mathbf{t}=\mathbf{t}_0} = \mathbf{c}(\mathbf{t}_0, r). \tag{5.47}$$

5.5.1 Algoritmo de Treinamento da rede MRL

O algoritmo de treinamento [38, 70] da rede MRL, baseado no critério de mínimos quadrados [31], utiliza idéias do algoritmo *back-propagation* [61], usado no treinamento das MLPs, e técnicas robustas para superar a não-diferenciabilidade das funções posto [39]. Assim como nas redes MLPs a fase de backpropagation objetivava o cálculo de derivadas de E^p com respeito aos pesos, nas redes MRL o algoritmo backpropagation também tem este papel. A função posto

é uma função vetorial não diferenciável. Portanto, iremos adaptar uma versão vetorial para a fase *backpropagation*, discutida no capítulo 4. A equação vetorial equivalente à equação (4.16) é dada por:

$$\nabla_{\mathbf{w}_k^{(l)}} E^p = \frac{\partial E^p}{\partial y_k^{(l)}} \frac{\partial y_k^{(l)}}{\partial x_k^{(l)}} \nabla_{\mathbf{w}_k^{(l)}} x_k^{(l)}. \tag{5.48}$$

Vamos então desenvolver os 3 fatores que aparecem no produto da equação (5.48).

(1) Primeiramente, vamos desenvolver o termo $\frac{\partial E^p}{\partial y_k^{(l)}}$ Com respeito à última camada, $l = L$, o termo $\frac{\partial E^p}{\partial y_k^{(l)}}$ pode ser diretamente calculado, pois notemos que:

$$\nabla_{\mathbf{y}^{(L)}} E^p = \frac{\partial E^p}{\partial e_k^{(L)}} \nabla_{\mathbf{y}^{(L)}} e_k^{(L)} = \mathbf{e}^{(L)} = \mathbf{d}^p - \mathbf{y}^{(L)}. \tag{5.49}$$

Ou seja,

$$\frac{\partial E^p}{\partial y_k^{(L)}} = e_k^{(L)} = (d_k^p - y_k^{(L)}). \tag{5.50}$$

O problema é quando o termo $\frac{\partial E^p}{\partial y_k^{(l)}}$ se refere às camadas escondidas, ou seja, quando $l < L$. Neste caso, usamos a regra da cadeia:

$$\nabla_{\mathbf{y}^{(l)}} E^p = \sum_{k=1}^{N_{l+1}} \frac{\partial E^p}{\partial y_k^{(l+1)}} \nabla_{\mathbf{y}^{(l)}} y_k^{(l+1)}. \tag{5.51}$$

De fato, não podemos calcular $\mathbf{e}^{(l)}$ diretamente para todo $l < L$, e um modo eficiente de superar esta dificuldade é definir $\mathbf{e}^{(l)}$ por meio de uma retropopagação:

$$\mathbf{e}^{(l)} = \sum_{k=1}^{N_{l+1}} e_k^{(l+1)} \nabla_{\mathbf{y}^{(l)}} y_k^{(l+1)}. \tag{5.52}$$

Comparando as equações (5.51) e (5.52), temos que

$$\frac{\partial E^p}{\partial y_k^{(l+1)}} = e_k^{(l+1)}. \tag{5.53}$$

O termo $\nabla_{\mathbf{y}^{(l)}} y_k^{(l+1)}$, que aparece nas equações (5.51) e (5.52), é obtido por meio das equações (4.6) e (4.7). Ou seja, temos que:

$$\nabla_{\mathbf{y}^{(l)}} y_k^{(l+1)} = g'(x_k^{(l+1)}) \nabla_{\mathbf{y}^{(l)}} x_k^{(l+1)}. \tag{5.54}$$

Desta forma,

$$\nabla_{\mathbf{y}^{(l)}} E^p = \mathbf{e}^{(l)} = \sum_{k=1}^{N_{l+1}} e_k^{(l+1)} g'(x_k^{(l+1)}) \nabla_{\mathbf{y}^{(l)}} x_k^{(l+1)}. \tag{5.55}$$

Esta equação é simplesmente uma versão vetorial da equação (4.24) que diz respeito à retropropagação dos erros. Voltando à equação (5.48), temos que o termo $\frac{\partial E^p}{\partial y_k^{(l)}}$ é igual a k-ésima componente do vetor $\mathbf{e}^{(l)}$ definido na equação (5.52), onde o gradiente $\nabla_{\mathbf{y}^{(l)}} x_k^{(l+1)}$ que aparece na equação (5.52) é calculado de acordo com a equação 5.47 para a parte não linear (posto) da função $x_k^{(l)}$ (Eq. (5.41)). Ou seja, usando as equações (5.41), (5.42) e (5.43) nós temos que:

$$\nabla_{\mathbf{y}^{(l-1)}} x_k^{(l)} = \lambda_k^{(l)} \nabla_{\mathbf{y}^{(l-1)}} \alpha_k^{(l)} + (1 - \lambda_k^{(l)}) \mathbf{b}_k^{(l)}. \tag{5.56}$$

(2) O termo $\frac{\partial y_k^{(l)}}{\partial x_k^{(l)}}$ da equação (5.48) é obtido claramente a partir das equações (4.6) e (4.7):

$$\frac{\partial y_k^{(l)}}{\partial x_k^{(l)}} = g'(x_k^{(l)}). \tag{5.57}$$

(3) Definimos o termo $\nabla_{\mathbf{w}_k^{(l)}} x_k^{(l)}$ que aparece na equação (5.48) como:

$$\nabla_{\mathbf{w}_k^{(l)}} x_k^{(l)} = \begin{pmatrix} \nabla_{\mathbf{a}_k^{(l)}} x_k^{(l)} \\ \frac{\partial x_k^{(l)}}{\partial \rho_k^{(l)}} \\ \nabla_{\mathbf{b}_k^{(l)}} x_k^{(l)} \\ \frac{\partial x_k^{(l)}}{\partial \theta_k^{(l)}} \\ \frac{\partial x_k^{(l)}}{\partial \lambda_k^{(l)}} \end{pmatrix}, \tag{5.58}$$

onde,

$$\nabla_{\mathbf{a}_k^{(l)}} x_k^{(l)} = \lambda_k^{(l)} \nabla_{\mathbf{a}_k^{(l)}} \alpha_k^{(l)} = \lambda_k^{(l)} \cdot \frac{Q(\alpha_k^{(l)} \mathbf{1} - \mathbf{y}^{(l-1)} - \mathbf{a}_k^{(l)})}{Q(\alpha_k^{(l)} \mathbf{1} - \mathbf{y}^{(l-1)} - \mathbf{a}_k^{l}) \mathbf{1}^\top}. \tag{5.59}$$

Com respeito ao termo $\frac{\partial x_k^{(l)}}{\partial \rho_k^{(l)}}$, temos que uma escolha possível para ele é dada por:

$$\frac{\partial x_k^{(l)}}{\partial \rho_k^{(l)}} = \lambda_k^{(l)} \cdot \left\{ 1 - \frac{1}{N_{l-1}} Q(\alpha_k^{(l)} \mathbf{1} - \mathbf{y}^{(l-1)} - \mathbf{a}_k^{(l)}) \mathbf{1}^\top \right\}. \tag{5.60}$$

E os outros termos que aparecem no vetor $\nabla_{\mathbf{w}_k^{(l)}} x_k^{(l)}$ são obtidos por diferenciação usual:

$$\nabla_{\mathbf{b}_k^{(l)}} x_k^{(l)} = (1 - \lambda_k^{(l)})\mathbf{y}^{(l-1)}. \tag{5.61}$$

$$\frac{\partial x_k^{(l)}}{\partial \theta_k^{(l)}} = 1 - \lambda_k^{(l)}. \tag{5.62}$$

$$\frac{\partial x_k^{(l)}}{\partial \lambda_k^{(l)}} = \alpha_k^{(l)} - \beta_k^{(l)}. \tag{5.63}$$

Tendo calculado os três fatores que aparecem na equação (5.48), temos que o gradiente $\nabla_{\mathbf{w}_k^{(l)}} E^p$ está bem definido para $l = 1, 2, \ldots, L$ e, consequentemente, o gradiente ∇E^p está bem definido. Sendo assim, podemos introduzir o algoritmo de treinamento, visto que foi possível calcular ∇E^p.

Algoritmo

Passo (1) Defina inicialmente $t = 0$ e inicialize os pesos $\mathbf{w}(0) = (\mathbf{w}^{(l)}(0), \mathbf{w}^{(2)}(0), \ldots, \mathbf{w}^{(L)}(0))$, com entradas em $[-1, 1]$.

Passo (2) Para $p = 1, \ldots, P$ faça:

(2.1) Apresente o padrão de entrada $\mathbf{x}^p$ à rede e determine a sua saída $\mathbf{y}^p$ usando as equações (4.6),(4.7) e as equações (5.41) a (5.43).

(2.2) Realize a fase *backpropagation* discutida na seção 4.5.1 para calcular os gradientes $\nabla_{\mathbf{w}_k^{(l)}} E^p$ (Eq (5.48)).

Passo (3) Atualize os pesos

$\mathbf{w}(t+1) = \mathbf{w}(t) + \triangle\mathbf{w}(t)$
onde $\triangle\mathbf{w}(t) = -\mu_0 \sum_{p=1}^{P} \nabla_{\mathbf{w}} E^p \mid_{\mathbf{w}(t)}$

E atualize $t = t + 1$.

Passo (4) Repita os passos (2) e (3) até um critério de parada ser alcançado.

5.5.2 Fronteiras de decisão da rede MRL

De acordo com as equações (5.39),(5.40) e (5.41), temos que cada nó de uma rede MRLNN fornece uma representação de um conjunto de hiperplanos. Os vetores normais destes hiperplanos dependerão do parâmetro λ e dos coeficientes $\mathbf{b}$. Se $\lambda = 1$, os hiperplanos serão paralelos a algum subconjunto de direções coordenadas canônicas. Vejamos o caso de uma rede MRLNN com um único neurônio, em $\mathbb{R}^2$, com um posto r fixo. Temos que a sua saída é dada por:

$$y = \lambda \cdot \mathcal{R}_r\left(\begin{pmatrix} x_1 + a_1 \\ x_2 + a_2 \end{pmatrix}\right) + (1-\lambda)\cdot(x_1 b_1 + x_2 b_2 + \theta_1) \tag{5.64}$$

Notemos que a fronteira, $y = 0$ fornece duas retas, pois

$$\mathcal{R}_r\left(\begin{pmatrix} x_1 + a_1 \\ x_2 + a_2 \end{pmatrix}\right) = x_1 + a_1 \quad \text{ou} \quad \mathcal{R}_r\left(\begin{pmatrix} x_1 + a_1 \\ x_2 + a_2 \end{pmatrix}\right) = x_2 + a_2 \tag{5.65}$$

5.6 Redes Neurais Morfológicas Modulares (MMNN)

A rede neural morfológica modular (MMNN), introduzida por R Sousa et al. [14], é uma rede neural do tipo alimentada adiante cuja arquitetura é baseada nas *decomposições de Banon e Barrera* [3], que no fundo dizem respeito à decomposição de um operador $\psi : \mathbb{L} \longrightarrow \mathbb{M}$ em termos de um supremo de ínfimos de erosões e anti-dilatações ou, similarmente, de um ínfimo de supremos de dilatações e anti-erosões. O termo *decomposição de Matheron* diz respeito à decomposição de um operador ψ que é crescente e invariante com respeito à translações em termos de um supremo de erosões ou, similarmente, em termos de um ínfimo de dilatações, onde estes operadores são definidos sobre reticulados da forma $\mathcal{P}(\mathbb{R}^d)$. Os exemplos de problemas onde estas decomposições podem ser aplicadas são detecção de bordas, restauração de imagens e reconhecimento de padrões. A MMNN pode ser usada para desenvolver operadores invariantes por translação binários ou em tons de cinza. Ao desenvolver os operadores em tons de cinza, os valores de sinais são normalizados no conjunto $[0, 1]$. A arquitetura geral da MMNN é capaz de aprender tanto operadores invariantes por translação em tons de cinza quanto binários. Para o treinamento da MMNN, são usadas idéias do algoritmo backpropagation (BP) [61] e a metodologia proposta por Pessoa e Maragos [39] para superar o problema da não-diferenciabilidade das funções posto, bem como algoritmos genéticos adaptativos (AG). A estrutura da rede MMNN pode ser vista como um caso especial da rede MRL, mas com arquiteturas e regras de treinamento específicos.

A Figura 5.13 representa as possíveis arquiteturas da rede MMNN de acordo com a decomposição de mapeamentos crescentes invariantes à translação em termos de um supremo de

erosões e em termos de um ínfimo de dilatações. A Figura 5.14 diz respeito as possíveis arquiteturas da MMNN segundo o teorema de Banon e Barrera. Ambas estruturas representam uma decomposição por alguns módulos independentes.

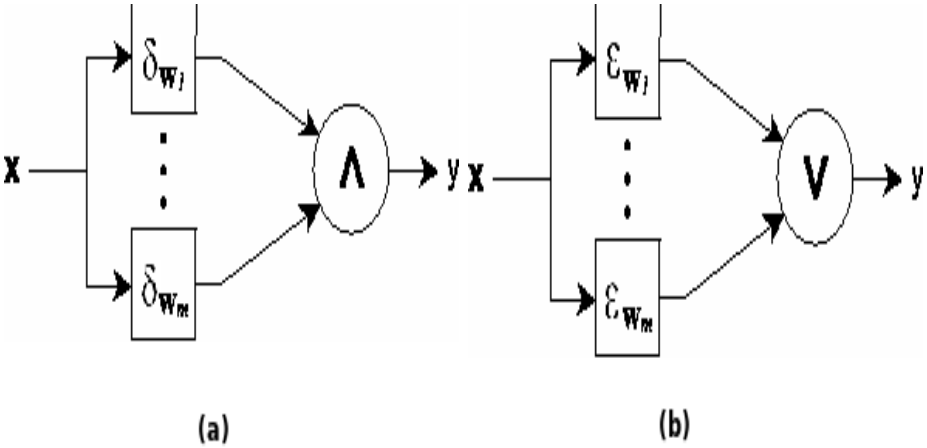

Fig. 5.13: (a) Estrutura da rede MMNN usando dilatações morfológicas, (b) Estrutura da rede MMNN usando erosões morfológicas

Caracterizamos a arquitetura ilustrada na Figura 5.14 de acordo com a decomposição por erosões de Matheron:

$$\varepsilon_k = v_k = \wedge_{i=1}^{N}(x_i - a_{ki}), \tag{5.66}$$

$$y = \vee_{i=1}^{N}\{v_i\}, \quad \text{onde} \quad \mathbf{v} = (v_1, v_2, \ldots, v_N), \tag{5.67}$$

onde $\mathbf{x}$ é o sinal de entrada para a MMNN, $\mathbf{a}_k \in \mathbb{R}^N$ representam os elementos de estruturação das erosões v_k, $k = 1, 2, \ldots, N$.

A matriz de pesos sinápticos, A, da MMNN é definida por

$$A = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N), \tag{5.68}$$

De maneira dual, a arquitetura para a decomposição de Matheron [32] por dilatações, de acordo com a Figura 5.14(b), é definida substituindo erosões por dilatações e o símbolo $\wedge$ por $\vee$.

As equações a seguir representam a arquitetura da Figura 5.15(a), de acordo com a repre-

sentação de Banon e Barrera através sup-geradores.

$$\begin{aligned}
\varepsilon_k &= u_{k1} = \wedge_{i=1}^{N}(x_i - a_{ki}), && (5.69)\\
\delta_k^a &= u_{k2} = 1 - \vee_{i=1}^{N}(x_i + b_{ki}), && (5.70)\\
v_k &= \wedge_{i=1}^{N} u_{ki} && (5.71)\\
\mathbf{u}_k &= (u_{k1}, u_{k2}), \quad k = 1, 2, \ldots, N && (5.72)\\
y &= \vee_{i=1}^{N} v_i, && (5.73)
\end{aligned}$$

onde, $\mathbf{v} = (v_1, v_2, \ldots, v_N)$.

Usando a equação (3.5), notamos que:

$$\begin{aligned}
\delta_{\mathbf{w}}^a(\mathbf{x}) &= \nu_*(\mathbf{w} \boxbslash \mathbf{x}) = [(\mathbf{w} \boxbslash \mathbf{x})^*]^\top && (5.74)\\
= (\mathbf{w} \boxbslash \mathbf{x})^* &= \mathbf{x}^* \boxslash \mathbf{w}^* = (\mathbf{w}^*)^\top \boxbslash (\mathbf{x}^*)^\top && (5.75)\\
= \delta_{(\mathbf{w}^*)^\top}((\mathbf{x}^*)^\top) &= \delta_{(\mathbf{w}^*)^\top}(\nu_*(\mathbf{x})) = \bar{\delta}_{(\mathbf{w}^*)^\top}(\mathbf{x}) && (5.76)
\end{aligned}$$

Isso mostra, essencialmente, que as MMNNs pertencem à classe de perceptrons morfológicos. Similarmente, é possível mostrar que $\varepsilon_{\mathbf{w}}^a = \bar{\varepsilon}_{(\mathbf{w}^*)^\top}(\mathbf{x})$.

Definimos as matrizes de pesos sinápticos da rede MMNN por

$$\begin{aligned}
A &= (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N), && (5.77)\\
B &= (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N), && (5.78)
\end{aligned}$$

onde $\mathbf{a}_k, \mathbf{b}_k \in \mathbb{R}^N$ representam os elementos de estruturação e são considerados parâmetros livres da rede neural MMNN.

De maneira dual, a arquitetura para a decomposição de Banon e Barrera através de inf-geradores, Figura 5.14(b), é definida substituindo dilatações por erosões, anti-dilatações por anti-erosões, o símbolo $\wedge$ por $\vee$ nas unidades sub-integradoras, e o símbolo $\vee$ por $\wedge$ na unidade integradora geral.

5.7 Algoritmo de treinamento da rede neural MMNN para a decomposição de Matheron

O algoritmo de treinamento da rede MMNN [1, 2, 14], ilustrada na Figura 5.14, admite duas abordagens, uma que utiliza idéias do algoritmo backpropagation [61] e a metodologia proposta por Pessoa e Maragos [39] para estimar derivadas de funções posto, e outra que considera o método de treinamento via algoritmos genéticos adaptativos (AG) [17] para determinar os pesos

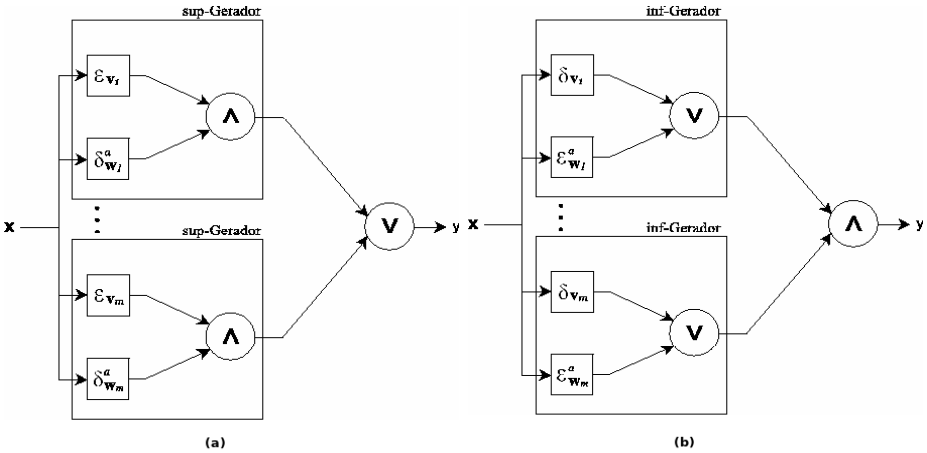

Fig. 5.14: (a) Arquitetura usada para a decomposição de Banon e Barrera por sup-geradores, (b) Arquitetura usada para a decomposição de Banon e Barrera por inf-geradores.

e, possivelmente, a arquitetura e número de módulos da rede MMNN.

Dado um conjunto de treinamento $\{(\mathbf{x}^p, d^p), \quad p = 1, 2, \ldots, P\}$, temos que a função objetivo a ser minimizada neste algoritmo de treinamento é a função E (Eq. (4.8)), que é escrita em função da soma dos erros individuais E^p dado pela equação (4.9). O procedimento de aprendizagem é dado simplesmente pela equação

$$A(t+1) = A(t) - \mu \sum_{p=1}^{P} \nabla E^p \mid_{A(t)} \tag{5.79}$$

onde μ é a taxa de aprendizagem e ∇E^p é dado por

$$\nabla E^p = \begin{pmatrix} \nabla_{\mathbf{a}_1} E^p \\ \vdots \\ \nabla_{\mathbf{a}_N} E^p \end{pmatrix} \tag{5.80}$$

Aplicando a regra da cadeia, temos que o gradiente da função E^p com respeito aos pesos $\mathbf{a}_k$ é dado por

$$\nabla_{\mathbf{a}_k} E^p = -e \frac{\partial y}{\partial v_k} \nabla_{\mathbf{a}_k} v_k, \quad k = 1, 2, \ldots, N. \tag{5.81}$$

Para se estimar a derivada parcial $\frac{\partial y}{\partial v_k}$ que aparece na equação 5.81, utilizamos os vetores indicadores posto $\mathbf{c}$, e funções suaves Q_σ de acordo com a definição 1 e a equação (5.44) respectivamente. Em outras palavras, a derivada parcial $\frac{\partial y}{\partial v_k}$ corresponde a k-ésima posição do

vetor indicador posto $\mathbf{c} = \nabla_{\mathbf{v}} y^{(L)}$ dado por:

$$\nabla_{\mathbf{v}} y = \frac{Q_\sigma(y \cdot \mathbf{1} - \mathbf{v})}{Q_\sigma(y \cdot \mathbf{1} - \mathbf{v}) \cdot \mathbf{1}^\top}, \tag{5.82}$$

Com respeito ao termo $\nabla_{\mathbf{a}_k} v_k$ temos que

$$\nabla_{\mathbf{a}_k} v_k = -\frac{Q_\sigma(v_k \cdot \mathbf{1} - \mathbf{x} + \mathbf{a}_k)}{Q_\sigma(v_k \cdot \mathbf{1} - \mathbf{x} + \mathbf{a}_k) \cdot \mathbf{1}^\top}, \tag{5.83}$$

Algoritmo usando back-propagation

Passo (1) Defina inicialmente $t = 0$ e inicialize matriz de pesos sinápticos $A(0) = (\mathbf{a}_1(0)\mathbf{a}_2(0)\ldots\mathbf{a}_N(0))$, com entradas pertencentes a $[0, 1]$.

Passo (2) Para $p = 1, \ldots, P$ faça:

(2.1) Apresente o padrão de entrada $\mathbf{x}^p$ à rede e determine a sua saída $\mathbf{y}^p$ usando as equações (4.6),(4.7) e as equações (5.66) a (5.67).

(2.2) Calcule os os gradientes $\nabla_{\mathbf{a}_k} E^p, \quad k = 1, 2, \ldots, N$ (Eq (5.81)).

Passo (3) Atualize a matriz de pesos sinápticos

$$A(t+1) = A(t) - \mu \sum_{p=1}^{P} \nabla E^p \mid_{A(t)}$$

E atualize $t = t + 1$.

Passo (4) Repita os passos (2) e (3) até um critério de parada ser alcançado.

5.8 Algoritmo de treinamento da rede neural MMNN para a decomposição de Banon e Barrera

De modo análogo ao caso do treinamento da rede MMNN segundo a decomposição de Matheron, temos que as equações de treinamento, que dizem respeito à rede MMNN ilustrada na Figura 5.14(a), são dadas por:

$$A(t+1) = A(t) - \mu \sum_{p=1}^{P} \nabla E^p \mid_{A(t)} \tag{5.84}$$

$$B(t+1) = B(t) - \mu \sum_{p=1}^{P} \nabla E^p \mid_{B(t)} \tag{5.85}$$

Os gradientes que aparecem nas equações (5.84) e (5.85) são obtidos por meio da regra da cadeia e são dados por:

$$\nabla_{\mathbf{a}_k} E^p = -e \frac{\partial y}{\partial v_k} \frac{\partial v_k}{\partial u_{k1}} \nabla_{\mathbf{a}_k} u_{k1}, \tag{5.86}$$

$$\nabla_{\mathbf{b}_k} E^p = -e \frac{\partial y}{\partial v_k} \frac{\partial v_k}{\partial u_{k2}} \nabla_{\mathbf{b}_k} u_{k2}, \tag{5.87}$$

para $k = 1, 21, \ldots, N$. Novamente, estimando as derivadas parciais que aparecem em (5.86) e (5.87), por meio da metodologia de Pessoa e Maragos [38], temos que as derivadas parciais são dadas por:

$$\nabla_{\mathbf{v}} y = \frac{Q_\sigma(y \cdot \mathbf{1} - \mathbf{v})}{Q_\sigma(y \cdot \mathbf{1} - \mathbf{v}) \cdot \mathbf{1}^\top}, \tag{5.88}$$

$$\nabla_{\mathbf{u}_k} v_k = \frac{Q_\sigma(v_k \cdot \mathbf{1} - \mathbf{u}_k)}{Q_\sigma(v_k \cdot \mathbf{1} - \mathbf{u}_k) \cdot \mathbf{1}^\top} = (\frac{\partial v_k}{\partial u_{k1}}, \frac{\partial v_k}{\partial u_{k2}}), \tag{5.89}$$

$$\nabla_{\mathbf{a}_k} u_{k1} = -\frac{Q_\sigma(u_{k1} \cdot \mathbf{1} - \mathbf{x} + \mathbf{a}_k)}{Q_\sigma(u_{k1} \cdot \mathbf{1} - \mathbf{x} + \mathbf{a}_k) \cdot \mathbf{1}^\top}, \tag{5.90}$$

$$\nabla_{\mathbf{b}_k} u_{k2} = -\frac{Q_\sigma((1 - u_{k2}) \cdot \mathbf{1} - \mathbf{x} - \mathbf{b}_k)}{Q_\sigma((1 - u_{k2}) \cdot \mathbf{1} - \mathbf{x} - \mathbf{b}_k) \cdot \mathbf{1}^\top}, \tag{5.91}$$

Algoritmo usando back-propagation

Passo (1) Defina inicialmente $t = 0$ e inicialize as matrizes de pesos sinápticos $A(0) = (\mathbf{a}_1(0)\mathbf{a}_2(0)\ldots\mathbf{a}_N(0))$, $B(0) = (\mathbf{b}_1(0)\mathbf{b}_2(0)\ldots\mathbf{b}_N(0))$ com entradas pertencentes a $[0, 1]$.

Passo (2) Para $p = 1, \ldots, P$ faça:

(2.1) Apresente o padrão de entrada $\mathbf{x}^p$ à rede e determine a sua saída $\mathbf{y}^p$ usando as equações (4.6),(4.7) e as equações (5.69) a (5.73).

(2.2) Calcule os os gradientes $\nabla_{\mathbf{a}_k} E^p, \nabla_{\mathbf{b}_k} E^p, \quad k = 1, 2, \ldots, N$ (Eqs (5.86) e (5.87)).

Passo (3) Atualize as matrize de pesos sinápticos

$$A(t+1) = A(t) - \mu \sum_{p=1}^{P} \nabla E^p \mid_{A(t)}$$
$$B(t+1) = B(t) - \mu \sum_{p=1}^{P} \nabla E^p \mid_{B(t)} \text{ E atualize } t = t+1.$$

Passo (4) Repita os passos (2) e (3) até um critério de parada ser alcançado.

As unidades de processamento da rede MMNN segundo a decomposição de Banon e Barrera são contadas de maneira semelhante ao perceptron morfológico, uma vez que a arquitetura dessas redes são iguais e pelo fato da rede MMNN pertencer à classe dos perceptrons morfológicos.

5.8.1 Breve introdução aos algoritmos genéticos adaptativos AG

Assim como acontece no meio ambiente, em um AG existe um grupo de indivíduos que competem entre si para garantir a própria sobrevivência e para assegurar que suas características sejam passadas adiantes pelos seus descendentes. Com essa analogia, cada indivíduo do AG é de fato uma solução candidata para o problema em questão, que por sua vez, faz o papel do próprio meio ambiente, já que estabelece os critérios que permitem avaliar se um indivíduo/solução é adaptado ou não.

As melhores soluções contam com uma maior probabilidade de sobreviver e através de operadores genéticos, gerar outras soluções (ou descendentes) de boa qualidade. Da combinação de bons indivíduos podem surgir indivíduos ainda melhores que tenham herdado as boas características daqueles que lhe deram origem. Cada iteração do processo é chamado geração. Após várias gerações existe uma grande probabilidade de que a população tenha aumentado a sua qualidade média, isto é, seja composta por indivíduos mais bem adaptados ao meio ambiente em questão - o problema. Neste ponto, as soluções candidatas tendem a ser mais parecidas.

É importante observar que, assim como na natureza, o processo de transmissão e recombinação de material genético não ocorre no nível de organismos (fenótipos), e sim no nível cromossômico (genótipos). Nos algoritmos genéticos as operações não são realizadas diretamente sobre as soluções candidatas, mas sobre uma codificação das mesmas. Torna-se necessário, portanto, definir uma maneira de descrever as soluções potenciais através de estruturas de dados apropriadas que são de fato manipuladas pelo algoritmo.

No AG clássico, a codificação das soluções é feita através de um vetor de bits. Esta representação é conhecida como representação binária. Nesse caso, uma população seria um conjunto de vetores compostos pelos símbolos 0 e 1.

O primeiro passo na definição de um AG é a escolha de uma maneira de representar as potenciais soluções, em outras palavras, estabelecer um *genótipo* que descreva os *fenótipos* do problema. Tal processo é conhecido como codificação. O próximo passo seria definir um modo

de avaliar os indivíduos com respeito ao seu grau de adequação ao meio. O papel de avaliar o genótipo de um indivíduo e lhe fornecer uma medida numérica de aptidão é atribuido à *função de fitness*. No caso de problemas de otimização, a função de fitness está intimamente ligada à *função objetivo*. Em outros casos, é necessário determinar um outro critério para distinguir as boas soluções das demais.

A *codificação* juntamente com a *função de fitness* e critério de seleção e recombinação constituem etapas relevantes do processo de definição de um AG que são intrinsecamente dependentes do problema. A Figura 5.15 apresenta o funcionamento de um AG simples.

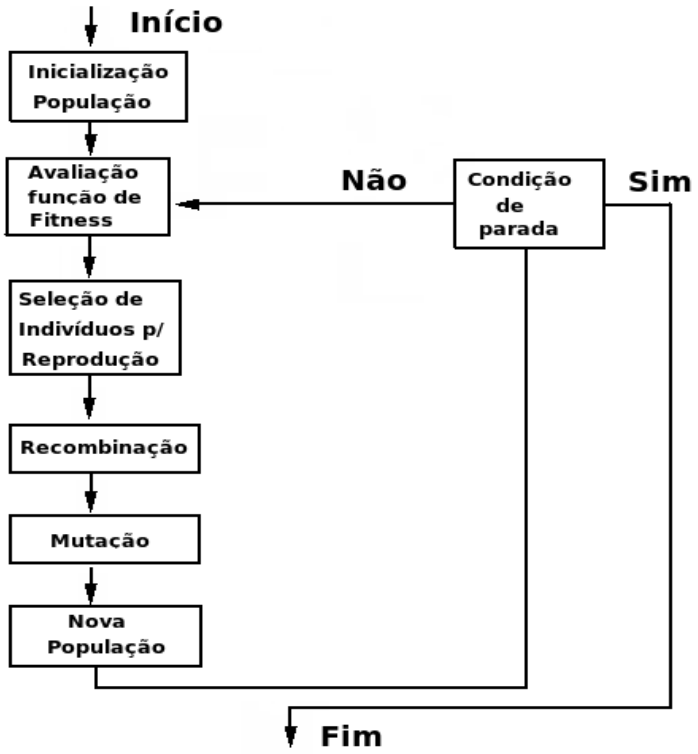

Fig. 5.15: Algoritmo genético simples

Dados esses passos, inicializa-se o processo iterativo descrito na Figura 5.15. Primeiramente deve-se inicializar a população de indivíduos. Em geral essa inicialização é feita de maneira aleatória. Em seguida deve-se selecionar n indivíduos para serem recombinados. De fato a seleção não é feita de maneira determinística, ou seja, qualquer indivíduo pode ser escolhido, contando cada um com uma probabilidade proporcional ao seu valor de *função de fitness*. Aplica-se então o operador de recombinação (*crossover*) que combina as unidades dos códigos de dois ou mais indivíduos para gerar descendentes. No caso binário, por exemplo, a troca ocorreria no nível dos *bits*. Apesar de serem diferentes dos pais, os filhos guardam nítidas semelhanças com ambos, assim como ocorre na natureza.

Os indivíduos gerados pela *recombinação* devem então ser submetidos à *mutação*. A mutação diferentemente do crossover, é um operador unário sendo aplicado sobre um único indivíduo provocando uma modificação, em geral pequena, em seu genótipo. Esse processo garante o surgimento de características novas, inexistentes, até então em uma população finita. Com respeito à codificação binária, a mutação pode ser implementada selecionando uma posição no vetor e invertendo o valor daquele bit de 0 para 1 e vice-versa.

As novas soluções candidatas geradas a partir da recombinação e da mutação constituem a próxima geração. Se o processo for sucessivamente repetido, existe uma boa chance de que a solução ótima seja alcançada, embora em alguns casos temos que uma solução razoavelmente boa seja satisfatória. Uma vez encontrada a solução, o processo deve ser interrompido de acordo com um critério de parada que em geral é feito sobre um número máximo de gerações (iterações).

Treinamento da rede MMNN por metodologia de AG

A metodologia que treina a rede MMNN por AG, tanto para a arquitetura que se refere à decomposição de Matheron por erosões e dilatações, quanto para a arquitetura que se refere à decomposição de Banon e Barrera por módulos sup-geradores e inf-geradores, tem a finalidade de determinar os pesos, a arquitetura e o número de módulos da MMNN. A Figura 5.16 representa um elemento da população do AG. Os termos $se_{(i)}, \quad i = 1, 2, \ldots, N$, denotam os pesos, os termos *arch* e *mod* se referem à arquitetura e o número de módulos da rede MMNN, respectivamente.

A *função de fitness*, que determina o desempenho de um indivíduo com respeito ao erro da resposta da rede com a saída desejada, adotada é o erro quadrático médio. A Figura 5.15 ilustra o algoritmo de treinamento da rede MMNN para o método proposto.

No algoritmo evolutivo de treinamento da rede MMNN consideramos uma população inicial de 100 indivíduos, a *probabilidade de recombinação* e *de mutação* são 0.9 e 0.1 respectivamente. O critério de parada é feito sobre o número de iterações do algoritmo genético simples.

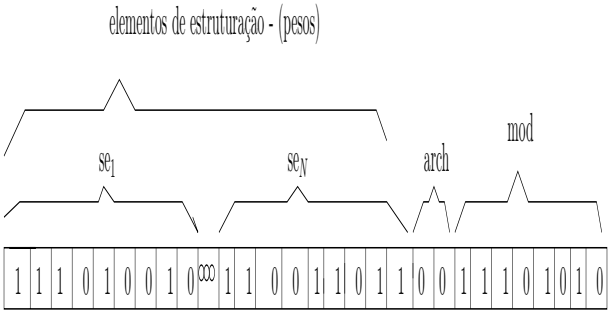

Fig. 5.16: código do cromossomo

Capítulo 6

Resultados Experimentais em Problemas de Classificação

Neste capítulo comparamos o desempenho em termos de erro de classificação (das redes neurais morfológicas e do perceptron de múltiplas camadas, discutidos no Capítulo 4 e 5) e do esforço computacional nos seguintes problemas de classificação que estão disponíveis na internet:

(1) O problema sintético que pode ser encontrado na base de dados disponível na homepage de Brian Ripley [37].

(2) O problema de diagnóstico de diabetes em uma população de Indios Pima, que pode ser encontrado na base de dados da Universidade da Califórnia, Irvine (UCI) [36].

(3) O problema de segmentação de imagens que pode ser encontrado na base de dados da Universidade da Califórnia, Irvine (UCI) [36].

6.1 Problema Sintético

O problema sintético apresentado em [45], contém um conjunto de dados que possui duas classes e dois *atributos*, ou seja, duas variáveis de entrada. Denotamos o primeiro atributo por X e o segundo por Y. A separação dos dados feita por Ripley [45] considera na fase de treinamento 125 padrões em cada classe, totalizando 250 padrões no total, e na fase de teste considera 500 padrões em cada classe, totalizando 1000 padrões para as duas classes. Em nossos experimentos, nós consideramos o conjunto original de treinamento, que possui 250 padrões, para a fase de treinamento e dividimos o conjunto de teste original, que contém 1000 padrões no total, em dois novos conjuntos: um conjunto de validação e um conjunto de teste, onde 500 padrões são separados aleatóriamente para a fase de validação e 500 padrões são separados aleatóriamente

para a fase de teste. Representaremos os padrões pertencentes à classe C_1 por (x) e os padrões pertencentes à classe C_0 por bolas abertas (∘), como ilustra a Figura 6.1.

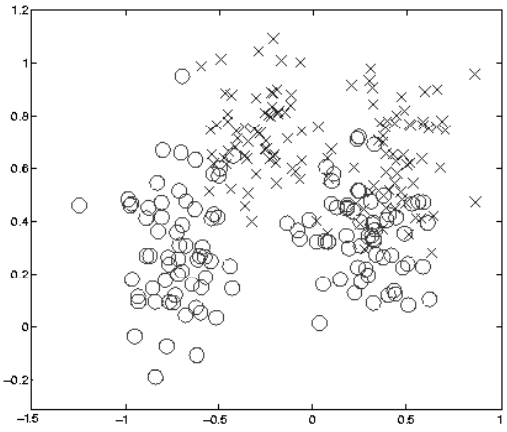

Fig. 6.1: Dados de treinamento do problema sintético

Apresentamos a superfície de decisão obtida pelo *perceptron morfológico*, pelo *perceptron morfológico dentrítico* e pelo perceptron linear nas Figuras 6.2, 6.3 e 6.4, respectivamente.

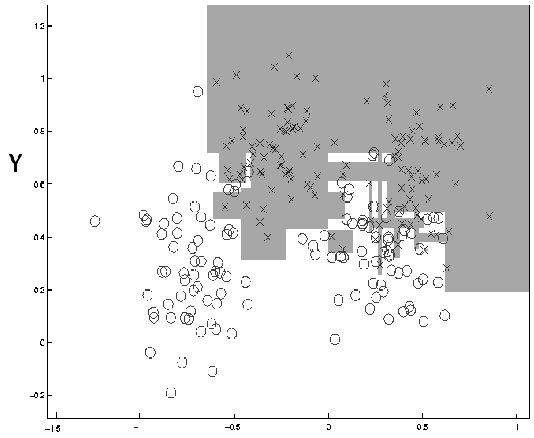

Fig. 6.2: Superfície de decisão obtida pelo perceptron morfológico

A região de ativação do *perceptron morfológico* foi obtida com 17 módulos, e a região de ativação do *perceptron morfológico dentrítico* foi obtida com 16 dentritos.

6.1.1 Análise dos resultados obtidos

O treinamento do perceptron morfológico utilizando o algoritmo supervisionado de Sussner [60], no experimento do problema sintético produziu automaticamente 17 módulos, fornecendo

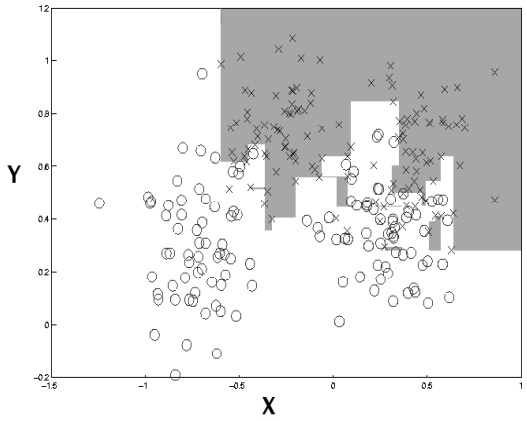

Fig. 6.3: Superfície de decisão obtida pelo perceptron morfológico dentrítico

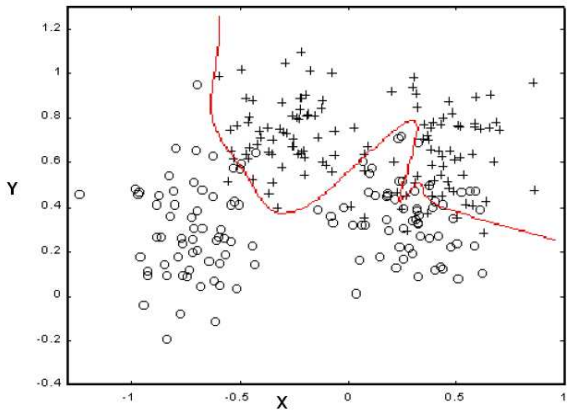

Fig. 6.4: Superfície de decisão obtida pelo perceptron linear retirada de [53]

34 e 17 neurônios na primeira e na segunda camada escondida, respectivamente. De um modo similar, o treinamento do perceptron morfológico com dentritos, usando o algoritmo construtivo de Ritter, produziu 16 dentritos. O processo de treinamento da rede PM e da rede PMD finaliza em um número finito de módulos e dentritos, respectivamente. Ambos algoritmos conseguem classificar corretamente todos os padrões na fase de treinamento, pela própria definição dos algoritmos. Com respeito ao perceptron morfológico, observamos na Figura 6.5 que o erro de validação se estabiliza a uma taxa de 12.12%, à partir da criação do nono módulo, e a taxa de erro no teste oscila muito pouco entre o nono e o décimo terceiro módulo. Portanto, neste caso, realizamos o treinamento até a taxa de erro no treinamento ir a zero, produzindo uma taxa de erro no teste de 10%, como pode ser visto na Tabela 6.1. No entanto, com respeito ao perceptron morfológico com dentritos aplicamos a técnica de *parada antecipada* durante o treinamento supervisionado para monitorar o erro de validação. O treinamento é suspenso

quando o erro de validação começa a crescer. De fato o erro de validação fornece uma medida da habilidade de generalização da rede. Percebemos que a taxa no erro de validação começa a subir com 7 dentritos produzidos, de acordo com a Figura 6.6. Então paramos o treinamento neste momento e conseguimos uma taxa de erro no teste de 12.8% (Tabela 6.1), ao passo que se continuássemos o treinamento, até que o erro no treinamento fosse a zero, teríamos um erro no teste de 17.7% (Tabela. 6.1).

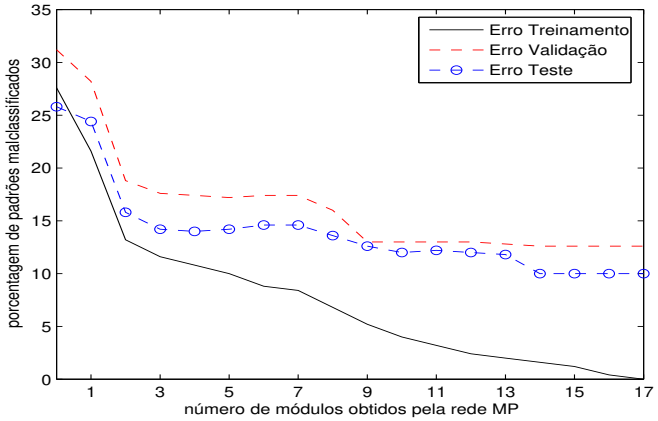

Fig. 6.5: Porcentagem de padrões mal-classificados pelo número de módulos da rede PM treinada com o algoritmo construtivo de Sussner no problema sintético

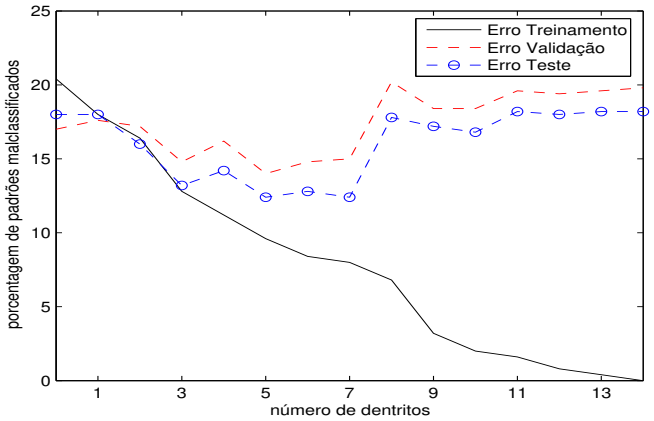

Fig. 6.6: Porcentagem de padrões mal-classificados pelo número de dentritos da rede PMD treinada com o algoritmo construtivo de Ritter e Urcid no problema sintético

Notemos que a rede PM apresentou uma taxa de erro inferior a rede PMD, como mostra a Tabela 6.1, a razão disso pode estar associada ao fato das hipercaixas produzidas pela rede PM não apresentarem pontos da classe C_1 na fronteira, ou isolados, fato que já pode ocorrer com

a rede PMD, comprometendo a capacidade de generalização da rede PMD. Nós consideramos também, neste experimento do problema sintético, uma rede MRL com uma camada escondida consistindo de 10 módulos, que emprega uma função de ativação g do tipo logística, sendo treinada com o algoritmo back-propagation, segundo metodologia de Pessoa e Maragos [38], (seção 5.51). Neste caso o tamanho do passo é igual a $\mu = 0.01$ e o parâmetro de suavidade é $\sigma = 0.05$. Estes parâmetros da rede e o número de módulos na camada escondida foram obtidos empiricamente. Notemos, no gráfico da Figura 6.7, que o erro de validação se estabiliza a partir da época 150 e as pequenas oscilações que podem ser observadas são consequência da descontinuidade da variável $r_k^{(l)}$. A rede MRL apresentou um erro de classificação no conjunto de teste do problema sintético de 23.7%, como pode ser observado na Tabela 6.1.

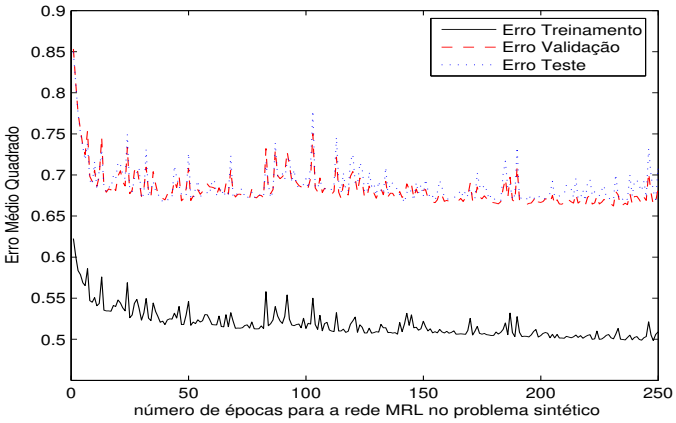

Fig. 6.7: Erro médio quadrado por época do algoritmo backpropagation aplicado para o treinamento da rede MRL no problema sintético.

Além de treinarmos a rede MMNN, de modo semelhante à rede MRL, nós também utilizamos um algoritmo genético (AG), considerando uma população inicial de 50 elementos (pesos sinápticos), um número máximo de 100 gerações, probabilidade de recombinação e mutação iguais a 0.9 e 0.1, respectivamente. As Figuras 6.8 e 6.9 representam o comportamento da rede MMNN para as duas abordagens. A rede MMNN foi a rede que apresentou o pior desempenho em comparação com os outros modelos, conforme pode ser observado na Tabela 6.1.

Com respeito a rede FLNN, a mesma produziu 47 neurônios durante a fase de treinamento. Cada neurônio representa uma caixa, e eles são criados à medida em que o algoritmo de treinamento evolui, de acordo com a Figura 6.10. O algoritmo da rede FLNN finaliza em um número finito de iterações e todos os dados são classificados em suas respectivas classes, não existindo a possibilidade de classificação com erros. De fato treinamos a rede FLNN até que o erro no treinamento fosse a zero. Ressaltamos que os neurônios da rede FLNN se diferenciam entre neurônios *anti-dilativos* e *anti-erosivos* dependendo do tipo de operação realizada pela função

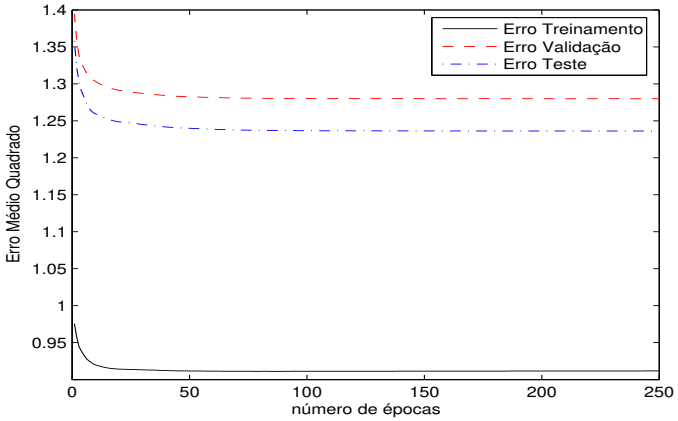

Fig. 6.8: Erro médio quadrado por época do algoritmo backpropagation aplicado para o treinamento da rede MMNN no problema sintético.

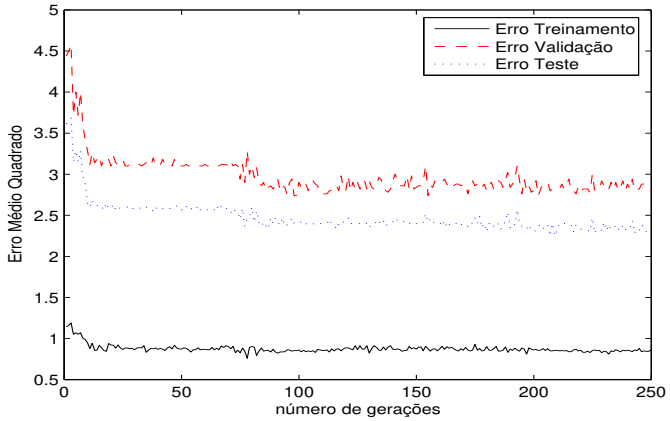

Fig. 6.9: Erro médio quadrado por época da rede MMNN treinada por AG no problema sintético.

de ordem parcial nebulosa p. A rede FLNN demonstrou um erro no teste no problema sintético inferior a todos os modelos implementados, conforme a Tabela 6.1. A Figura 6.10 apresenta a evolução nas taxas de erro no teste e validação pelo número de neurônios produzidos pela rede FLNN no problema sintético. Implementamos também uma rede MLP com 10 neurônios escondidos, treinada com técnica de descida de gradiente com momento e passo adaptado (taxa de aprendizagem $\eta = 10^{-4}$, parâmetros de acréscimo e decréscimo 1.05 e 0.6 respectivamente, fator de momento $\alpha = 0.9$). Tomamos como referência o trabalho de Alexandre Rubesam [53], que modelou várias MLP's para o problema. Aplicamos a técnica de parada antecipada, ou seja, interrompemos o treinamento da rede MLP quando o erro de validação começou a subir. A Figura 6.11 ilustra o comportamento da rede MLP no problema sintético. A Tabela 6.2 se

refere ao esforço computacional exigido no treinamento dos modelos no problema sintético.

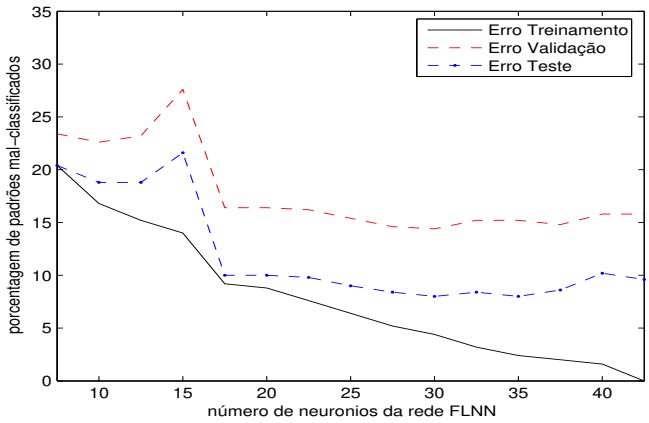

Fig. 6.10: Porcentagem de padrões malclassificados pelo número de neurônios produzidos pela rede FLNN no problema sintético.

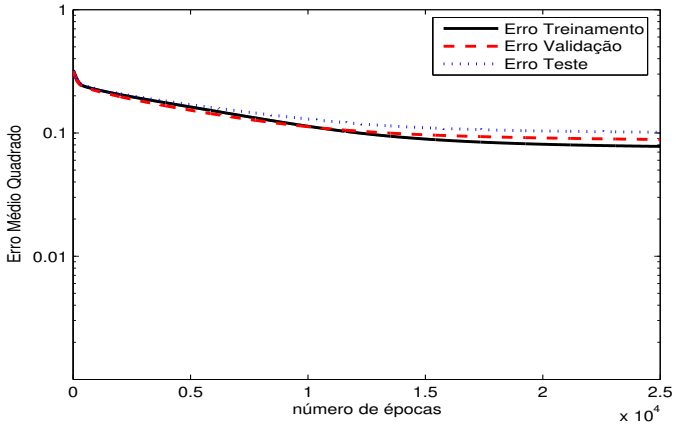

Fig. 6.11: Erro médio quadrado pelo número de épocas da rede MLP treinada pelo algoritmo backpropagation no problema sintético.

Tab. 6.1: Porcentagem de padrões malclassificados para o treinamento e teste, respectivamente no problema sintético

Rede Neural	Treinamento. (%)	Teste. (%)
PM	0	10
PMD	7.1	12.8
MRL	20	23.7
MMNN(BP)	24.2	27
MMNN(AG)	23.1	30
FLNN	0	9.6
MLP	10	11

Percebemos na Tabela 6.1 que os resultados obtidos pelas redes PM, PMD e FLNN foram similares à MLP tradicional. No entanto, o algoritmo backpropagation converge em um número de épocas substancialmente superior aos modelos morfológicos, como pode ser observado na Tabela 6.2, pois o algoritmo backpropagation é baseado no processo iterativo de minimização da função Erro, definida na equação (4.8). Além do que, o número de unidades escondidas da rede MLP deve ser especificado antes do treinamento. Os modelos morfológicos PM, PMD e FLNN atualizam suas arquiteturas durante o processo de aprendizagem e requerem essencialmente um número finito de passos, evitando um processo iterativo longo. Os modelos MRL e MMNN apresentaram um resultado inferior, com respeito à classificação, comparado com os outros modelos. De fato estes modelos operam, em um certo sentido técnico, como a MLP tradicional, ou seja, utilizam um algoritmo backpropagation para minimizar a função Erro e precisam que a sua arquitetura seja especificada antes do treinamento. No entanto, estas redes preservam a propriedade de que suas unidades de processamento calculam operações morfológicas que são operações de máximo ou mínimo e adição, não envolvendo multiplicação. Sendo assim os modelos morfológicos fornecem cálculos mais rápidos do que a rede MLP tradicional.

Tab. 6.2: Esforço Computacional no Treinamento no prob. sintético

Rede Neural	número de épocas	número de unid. process.
PM	17	52
PMD	16	49
MRL	250	41
MMNN(BP)	250	31
MMNN(AG)	150	31
FLNN	141	47
MLP	25000	11

6.2 Problema de diabetes nos índios Pima

A *Diabetes mellitus* é um dos desafios mais sérios na área da saúde enfrentado por nativos americanos nos Estados Unidos hoje. Dentre mais de 500 organizações tribais americanas nativas, os índios Pima, que agora vivem no sudeste do Arizona, têm sido extensivamente estudados devido à alta ocorrência de *diabetes do tipo* 2 nestes indivíduos.

O problema de diabetes nos índios Pima diz respeito à classificação de mulheres como portadoras ou não de diabetes. Tais mulheres pertencem à uma população de 768 mulheres com mais de 21 anos de idade, descendentes de índios Pima, residindo nas proximidades de Phoenix, no Arizona, EUA. Cada mulher foi testada de acordo com o conjunto de dados que foi originalmente publicado pelo *Instituto Nacional de Diabetes e Doênças Digestivas*. As variáveis medidas foram as seguintes:

1 Número de vezes que a mulher ficou grávida;

2 Concentração de plasma glucose em um teste oral de tolerância à glucose;

3 Pressão diastólica do sangue (mm Hg);

4 Grossura da pele do tríceps;

5 Insulina no soro ($\frac{\mu U}{ml}$);

6 Indice de massa corpórea ($\frac{Kg}{m^2}$);

7 Função de pedigree de diabetes;

8 Idade em anos.

De 768 observações, 376 estavam incompletas, principalmente na variável 5. Usamos todas as observações completas, excluindo a variável 5, de modo que restaram 532 observações. Os dados, disponíveis em [36], estão organizados da seguinte maneira: o conjunto de treinamento consiste de 200 observações e o conjunto de teste consiste de 332 observações. Nos nossos experimentos, nós utilizamos o mesmo conjunto de treinamento, disponível em [36], na fase de treinamento e dividimos o conjunto de teste original em um conjunto de validação de tamnho 166 e um conjunto de teste de tamanho 166, ambos gerados aleatóriamente. A classe 1 representa a ocorrência de diabetes nos índios e a classe 2 representa a não-ocorrência de diabetes nos índios.

6.2.1 Análise dos resultados obtidos

O treinamento do perceptron morfológico utilizando o algoritmo supervisionado de Sussner [60], no experimento do problema de diabetes nos índios Pima produziu automaticamente 25 módulos, fornecendo 50 e 25 neurônios na primeira e na segunda camada escondida respectivamente. De um modo similar, o treinamento do perceptron morfológico com dentritos, usando o algoritmo construtivo de Ritter e Urcid, produziu 26 dentritos. Com respeito ao perceptron morfológico com dentritos, nós aplicamos a técnica de parada antecipada, parando o treinamento no momento em que o décimo quarto dentrito é produzido, fornecendo desta maneira uma taxa de erro no teste de 27.4%, como pode ser visto na Figura 6.12. Já para o perceptron morfológico, realizamos o treinamento até que a taxa de erro no treinamento fosse a zero, produzindo desta maneira uma taxa de 22.1% no erro de teste (Figura 6.13). Desta forma, as taxas de erro no teste apresentadas pela rede PMD e pela rede PM são 27.4% e 21.2%, respectivamente, como pode ser observado na Tabela 6.3.

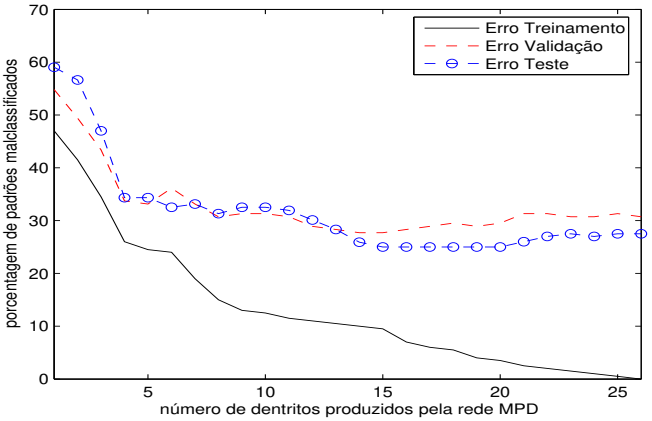

Fig. 6.12: Porcentagem de padrões mal-classificados pelo número de dentritos da rede PMD treinada com o algoritmo construtivo de Ritter no problema de diabetes nos Indios Pima

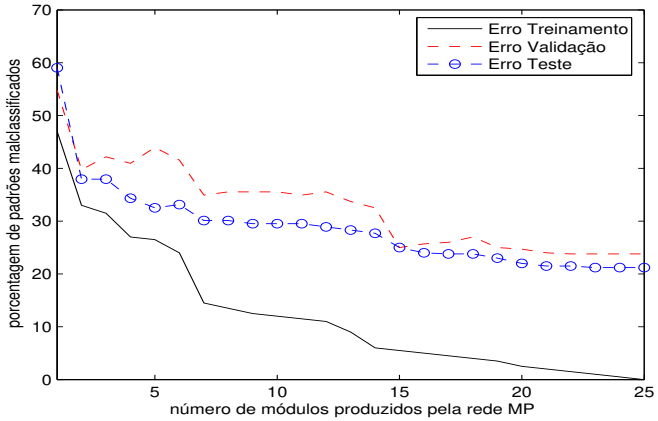

Fig. 6.13: Porcentagem de padrões mal-classificados pelo número de módulos da rede MP treinada com o algoritmo construtivo de Sussner no problema de diabetes nos índios Pima

Nós consideramos também, neste experimento do problema de diabetes dos índios Pima, uma rede MRL com uma camada escondida consistindo de 10 módulos, que emprega uma função de ativação do tipo logística e é treinada com algoritmo back-propagation, segundo metodologia de Pessoa e Maragos [38], (seção 5.51), usando um tamanho de passo $\mu = 0.01$ e um parâmetro de suavidade $\sigma = 0.05$. Devemos ressaltar que estes parâmetros foram obtidos empiricamente. A rede MRL apresentou uma taxa de erro de 34.7% no conjunto de teste, como pode ser observado na Tabela 6.3 e o comportamento da rede MRL neste problema é apresentado na Figura 6.14.

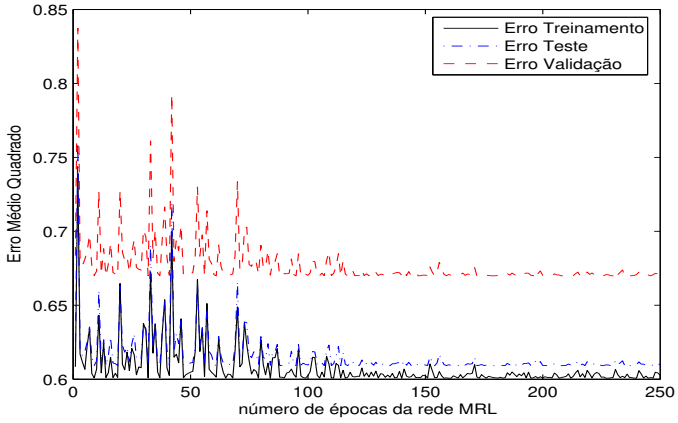

Fig. 6.14: Erro médio quadrado por época do algoritmo backpropagation aplicado para o treinamento da rede MRL no problema de diabetes nos índios Pima.

A rede FLNN produziu 74 neurônios durante a fase de treinamento. Treinamos a rede FLNN até que o erro de treinamento fosse a zero e obtivemos uma taxa de 14.8% de padrões mal-classificados no conjunto de teste como pode ser observado na Tabela 6.3 e na Figura 6.15. Treinamos a rede MMNN, de modo semelhante à rede MRL, e utilizamos também um algoritmo genético (AG), considerando uma população inicial de 50 elementos (pesos sinápticos), um número máximo de 100 gerações, probabilidade de recombinação e mutação iguais a 0.9 e 0.1, respectivamente. A rede MMNN apresentou taxas de erro de 37.5% e 30.7% para a abordagem de backpropagation e algoritmo genético respectivamente (Tabela 6.3). As Figuras 6.16 e 6.17 apresentam o comportamento da rede MMNN, segundo as duas abordagens consideradas, com respeito a evolução da taxa de erro nos conjuntos de teste e de validação.

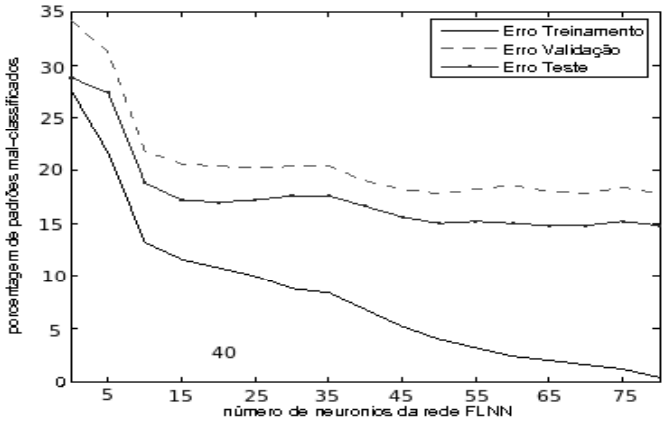

Fig. 6.15: porcentagem de padrões mal-classificados pelo número de neurnios produzidos pela rede FLNN no problema de diabetes nos índios Pima.

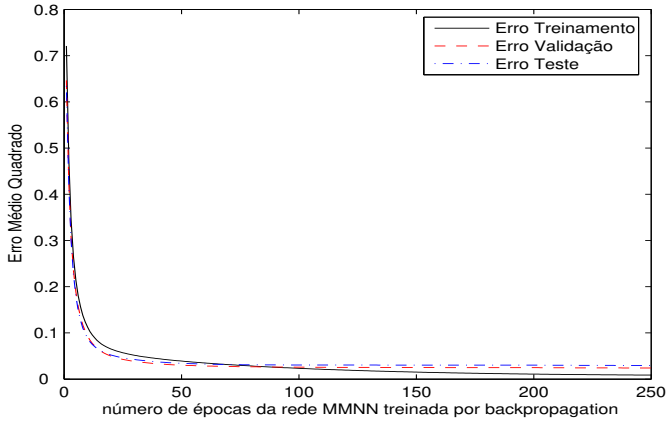

Fig. 6.16: Erro médio quadrado por época do algoritmo backpropagation aplicado para o treinamento da rede MMNN no problema de diabetes nos índios Pima.

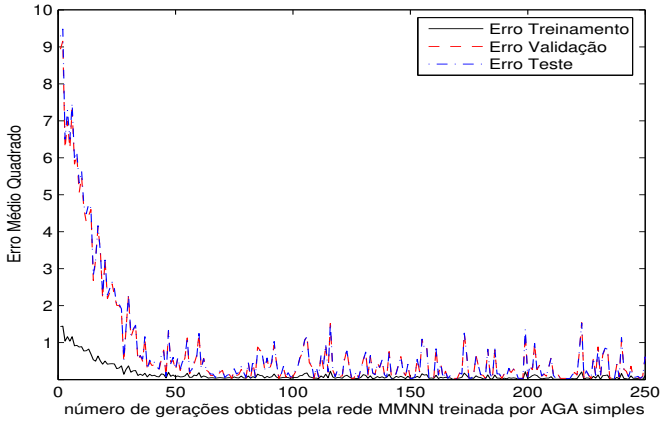

Fig. 6.17: Erro médio quadrado por época da rede MMNN treinada por AG no problema de diabetes nos índios Pima.

Implementamos também uma rede MLP com 10 neurônios escondidos, treinada com técnica de descida de gradiente com momento e passo adaptado (taxa de aprendizagem $\eta = 10^{-4}$, parâmetros de acréscimo e decréscimo 1 e 0.5, ajustados empiricamente, respectivamente, fator de momento $\alpha = 0.9$). Neste experimento, o erro de validação se estabilizava a partir de um certo número de épocas. Neste sentido, fixamos uma precisão de 0.15 para o erro médio quadrático e treinamos a rede MLP até que a MLP atinjisse este erro, como podemos observar pela Figura 6.18. A rede MLP apresentou uma taxa de erro de 20.18% neste problema (Tabela 6.3).

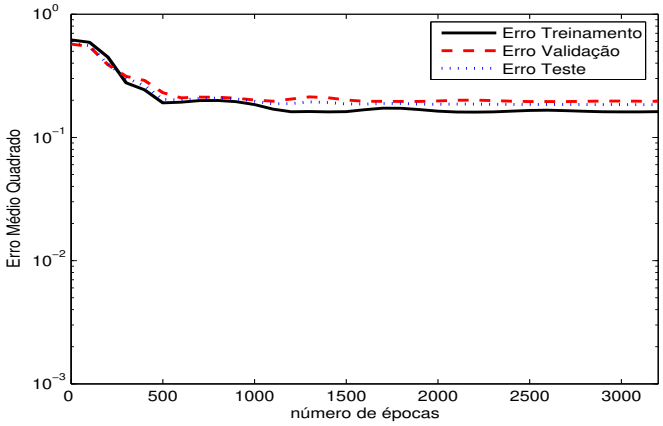

Fig. 6.18: Erro médio quadrado pelo número de épocas da rede MLP treinada pelo algoritmo backpropagation no problema de diabetes nos índios Pima.

A tabela 6.4 diz respeito ao esforço computacional exigido no treinamento dos modelos discutidos no problema de diabetes nos índios Pima.

Tab. 6.3: Porcentagem de padrões malclassificados no problema de diabetes nos índios Pima para o Treinamento e Teste.

Rede Neural	Treinamento (%)	Teste (%)
PM	0	21.2
PMD	0	27.4
MRL	31.2	34.7
MMNN(BP)	31.9	37.5
MMNN(AG)	23.1	30.7
FLNN	0	14.8
MLP	22.4	20.18

Podemos observar na Tabela 6.3 que os modelos morfológicos PM, PMD e FLNN apresentaram resultados similares à MLP e os modelos MRL e MMNN apresentaram o pior desempenho comparado com todos os modelos. Neste problema de diabetes dos índios Pima podemos observar, a partir da Tabela 6.4, que a rede MLP requeriu um número de épocas consideravelmente menor do que no problema sintético para convergir.

Tab. 6.4: Esforço Computacional no Treinamento para o problema de diabetes nos índios Pima

Rede Neural	número de épocas	número unid. process.
PM	25	76
PMD	26	79
MRL	250	41
MMNN(BP)	250	31
MMNN(AG)	150	31
FLNN	234	78
MLP	3500	11

6.3 Problema de Segmentação de Imagens

A classificação de bloco de imagem é um passo importante na segmentação baseada em bloco de imagem. Consiste de três passos principais:

- Dividir toda a imagem em regiões (ou blocos) $r \times r$ pixel;
- Extrair as características dos blocos de imagens;
- Aplicar o algoritmo para classificar os blocos.

O conjunto de dados em que aplicamos nossos algoritmos foram retirados do conjunto de dados de segmentação de imagens da University of California Irvine [36], cujos exemplos foram obtidos randomicamente de um conjunto de dados de sete imagens. O conjunto de dados original contém 210 exemplos para o treinamento e 2100 exemplos para a fase de testes, sendo que cada exemplo diz respeito a um bloco de imagem de 3×3 pixel contendo 19 atributos contínuos. Originalmente, o problema refere-se à classificação de padrão em uma das sete classes (brickface, sky, foliage, cement, window, path e grass). Nos nossos experimentos nós nos concentraremos em determinar se um padrão particular pertence à classe brickface ou não.

Com respeito à rede neural MLP, nós consideramos o conjunto de treinamento original e particionamos randomicamente o conjunto de teste em tamanhos iguais: $n_T = 1050$ e $n_V = 1050$ para teste e validação, respectivamente. Já para os outros modelos ajustados, nós dividimos randomicamente o conjunto de dados total ($treinamento + teste = 2310$) em três conjuntos de tamanhos iguais para treinamento, validação e teste. Considerando esta partição dos conjuntos de treinamento, validação e teste para a rede MP e para a rede PMD, obtemos taxas de erro no teste de 13% e 14% para a rede PM e para a rede PMD, respectivamente, como podemos observar pelas Figuras 6.19 e 6.20, respectivamente. De fato, acreditamos que estes modelos aprenderiam mais se o conjunto de treinamento fosse maior. Neste sentido, fizemos um novo experimento com uma nova proposta, considerando o conjunto de treinamento de tamanho $n_{train} = 210 + 1050$ e o conjunto de teste de tamanho $n_T = 1050$.

Fazendo esta consideração, obtivemos uma taxa de erro no teste de 6.5% para a rede MP e de 6.1% para a rede PMD, de acordo com a Tabela 6.5. As Figuras 6.21 e 6.22 ilustram esta situação. Nos modelos que discutiremos a seguir, consideramos a proposta que diz respeito à divisão em tamanhos iguais dos conjuntos de treinamento, validação e teste.

Nós consideramos também, neste experimento do problema de segmentação, uma rede MRL com uma camada escondida consistindo de 18 módulos, obtidos empiricamente, que emprega

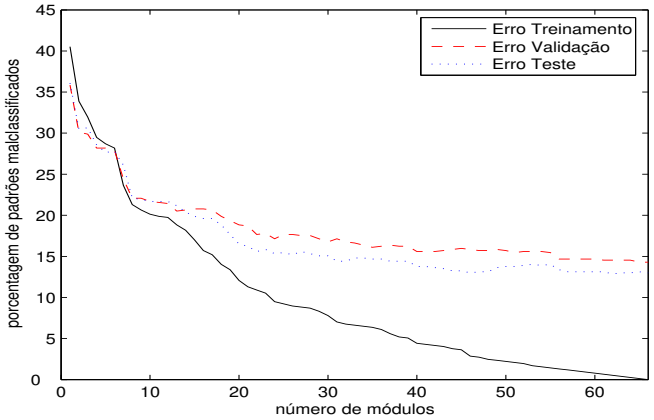

Fig. 6.19: Porcentagem de padrões malclassificados pelo número de módulos produzidos pela rede MP

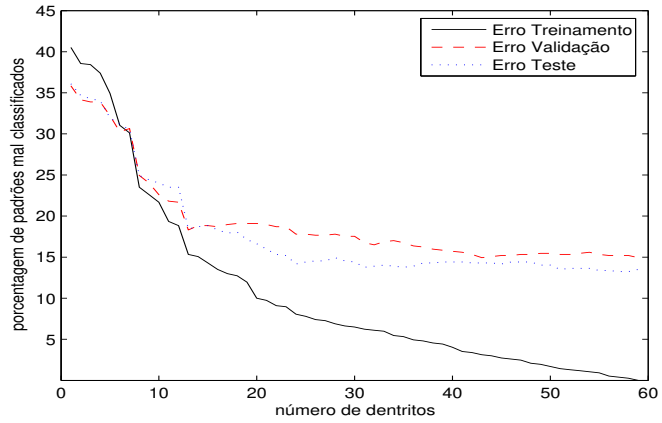

Fig. 6.20: Porcentagem de padrões malclassificados pelo número de dentritos produzidos pela rede PMD

uma função de ativação g do tipo logística e é treinada com algoritmo back-propagation, segundo metodologia de Pessoa e Maragos [38], descrita na (seção 5.51), usando um tamanho de passo $\mu = 0.01$ e um parâmetro de suavidade $\sigma = 0.05$. Notamos a partir da Figura 6.23 que o erro de validação obtido pela rede MRL se estabiliza a partir de 50 épocas. A rede MRL obteve uma taxa de erro no conjunto de teste de 10.5%, como pode ser observado na Tabela 6.5. Treinamos a rede MMNN, de modo semelhante à rede MRL, e utilizamos também um algoritmo genético (AG), considerando uma população inicial de 50 elementos (pesos sinápticos), um número máximo de 100 gerações, probabilidade de recombinação e mutação iguais a 0.9 e 0.1 respectivamente (Figura 6.24 e 6.25 respectivamente). A rede MMNN apresentou taxas de erro no conjunto de teste de 11.18% e 12.2% segundo a abordagem backpropagation e via algoritmos genéticos, respectivamente, de acordo com a Tabela 6.5.

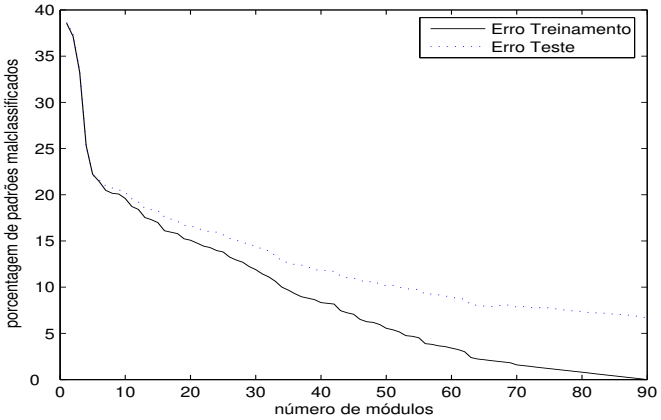

Fig. 6.21: Porcentagem de padrões malclassificados pelo número de módulos produzidos pela rede MP

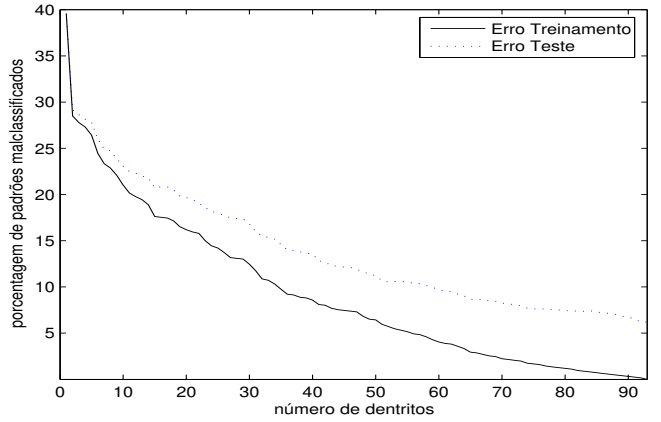

Fig. 6.22: Porcentagem de padrões malclassificados pelo número de dentritos produzidos pela rede PMD

A rede FLNN produziu 90 neurônios durante a fase de treinamento. Treinamos a rede FLNN até que o erro de treinamento fosse a zero e obtivemos uma taxa de 1.9% de padrões malclassificados no conjunto de teste, conforme pode ser visto na Tabela 6.5. A Figura 6.26 apresenta a evolução das taxas de erros de teste e validação obtidas pela rede FLNN neste experimento. Implementamos também uma rede MLP com 5 neurônios na primeira e na segunda camada escondida, treinada com técnica de descida de gradiente com momento e passo adaptado (taxa de aprendizagem $\eta = 10^{-4}$, parâmetros de acréscimo e decréscimo 1.05 e 0.6, respectivamente, fator de momento $\alpha = 0.9$). O número de neurônios e camadas, bem como os parâmetros de aprendizagem, foram determinados empiricamente. Neste experimento, o erro de validação se estabilizava a partir de um certo número de épocas. Neste sentido, fixamos uma precisão de 0.15 para o erro médio quadrático e treinamos a rede MLP até que a MLP atinja

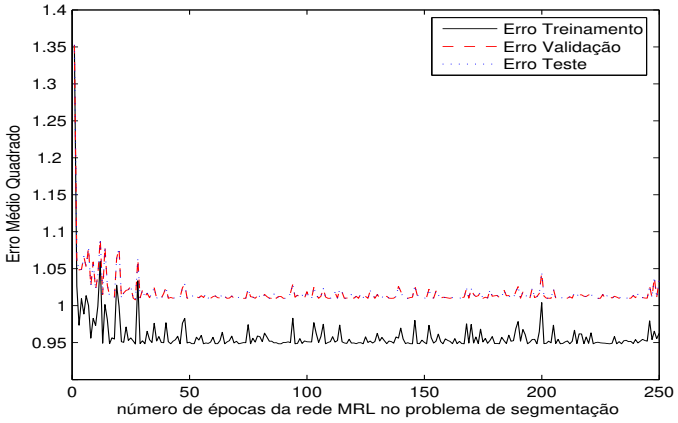

Fig. 6.23: Erro médio quadrado por época do algoritmo backpropagation aplicado para o treinamento da rede MRL no problema de segmentação

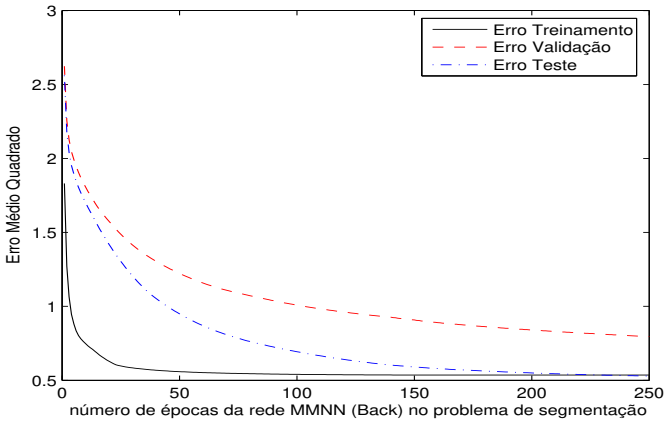

Fig. 6.24: Erro médio quadrado por época do algoritmo backpropagation aplicado para o treinamento da rede MMNN no problema de segmentação.

este erro, como podemos observar pela Figura 6.27. A MLP considerada apresentou uma taxa de erro no conjunto de teste de 5.6%, de acordo com a Tabela 6.5. A Tabela 6.6 diz respeito ao esforço computacional exigido no treinamento dos modelos no problema de segmentação de imagens.

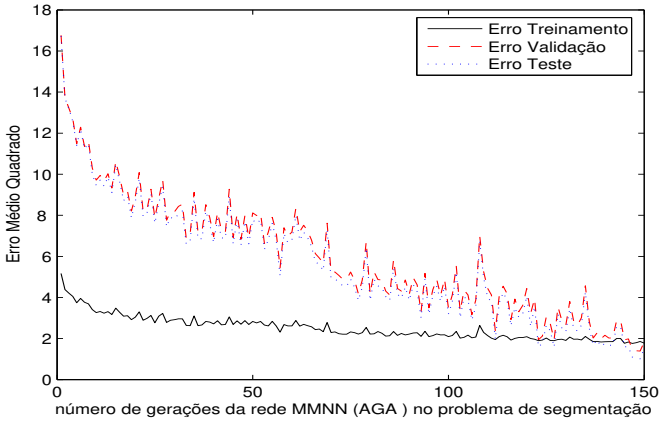

Fig. 6.25: Erro médio quadrado por época da rede MMNN treinada por AG no problema de segmentação.

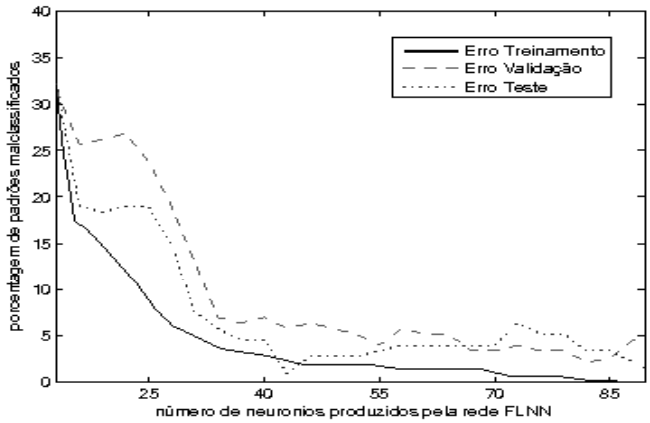

Fig. 6.26: Porcentagem de padrões malclassificados pelo número de produzidos pela rede FLNN.

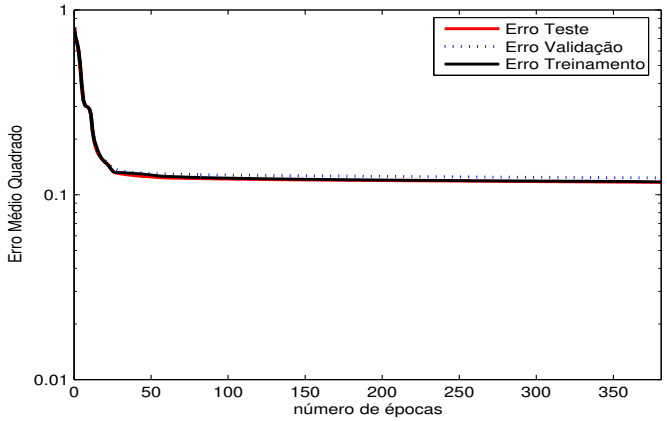

Fig. 6.27: Erro médio quadrado pelo número de épocas da rede MLP treinada pelo algoritmo back-propagation no problema de segmentação.

Tab. 6.5: Porcentagem de padrões malclassificados no problema de segmentação de imagens para o Treinamento e Teste.

Rede Neural	Treinamento (%)	Teste (%)
PM	0	6.5
PMD	0	6.1
MRL	10	10.5
MMNN(BP)	12	11.18
MMNN(SAG)	11	12.2
FLNN	0	1.2
MLP	4.6	5.6

Tab. 6.6: Esforço Computacional no Treinamento para o problema de segmentação de imagens

Rede Neural	número de épocas	número de unid. process.
PM	90	271
PMD	100	301
MRL	250	73
MMNN(BP)	250	55
MMNN(AG)	150	55
FLNN	270	90
MLP	370	11

6.4 Experimentos utilizando validação cruzada

Em geral, quando não existem exemplos suficientes para serem separados em conjuntos de treinamento, validação e teste utiliza-se o procedimento de validação cruzada [61, 54] para se estimar o erro de generalização [42]. A validação cruzada K-fold usa parte do conjunto de dados disponível para ajustar o modelo e uma parte diferente para validá-lo. Particiona-se aleatóriamente o conjunto de dados em K partições de tamanhos iguais. Para a k-ésima partição, nós ajustamos a rede às outras $K-1$ partes do conjunto de dados. Calculamos o erro de validação do modelo ajustado sobre a k-ésima partição. Realizamos este procedimento para $k = 1, 2, \ldots, K$ e calculamos a média dos erros de validação de cada partição que foi disponibilizada para validação, definindo esta medida como sendo o erro de validação. Escolhas típicas de K são 5 ou 10. O caso em que K é igual ao número de todos os dados disponíveis é chamado *leave-one-out*. A validação cruzada valida o desempenho do modelo de rede neural. Das K redes que foram obtidas do procedimento de validação cruzada K-fold, nós selecionamos aquela que produziu o menor erro de validação e implementamos uma fase de teste para esta rede. Se eventualmente existirem redes que produziram o mesmo valor de erro de validação, nós implementamos uma fase de teste para estas redes e calculamos o erro de teste como sendo a média dos erros obtidos no conjunto de testes destas redes.

6.4.1 Resultados utilizando validação cruzada no problema sintético

O conjunto de dados do problema sintético, disponível em [45], fornece 250 padrões para uma fase de treinamento e 1000 padrões para uma fase de teste. Para todos os modelos de redes neurais que consideramos nesta dissertação, nós aplicamos a validação cruzada 25-fold para o problema sintético. Ou seja, o conjunto de treinamento, de tamanho $n_T = 250$, é dividido em 25 partes, cada qual contendo 10 padrões distintos e aplica-se o procedimento de validação cruzada 25-fold. Além disso, para os modelos de redes neurais: MRL, MMNN(BP), MMNN(AG) e MLP, nós consideramos os mesmos parâmetros de definição das redes que foram considerados e

discutidos na seção 6.1.1. As Figuras 6.28 e 6.29 representam o resultado da validação cruzada 25-fold para as redes PM e PMD, respectivamente.

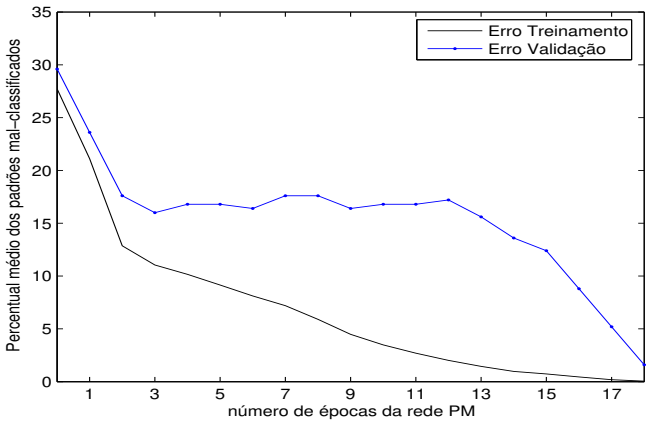

Fig. 6.28: Percentual médio de padrões malclassificados pelo número de épocas da rede PM no problema sintético.

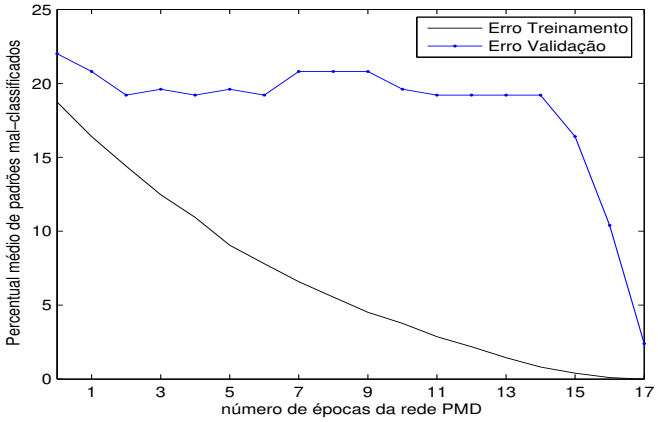

Fig. 6.29: Percentual médio de padrões malclassificados pelo número de épocas da rede PMD no problema sintético.

Como podemos observar, o percentual médio de padrões mal-classificados decresce ao longo das iterações. Este fato sugere que as redes PM e PMD estão fornecendo boa generalização [42]. Isto justifica o fato de estas redes obterem um resultado bem satisfatório com respeito aos erros de classificação no conjunto de teste do problema sintético, como pode ser observado na tabela 6.7. Neste procedimento de validação cruzada 25-fold, dentre as 25 redes obtidas do procedimento, quatro redes PM apresentaram o menor valor de erro de validação. Aplicamos estas quatro redes ao conjunto de teste, implementando parada antecipada, e calculamos a

média dos erros no conjunto de teste definindo esta média como sendo o erro no conjunto de teste. A rede PM apresentou um erro de 11.2% no conjunto de teste. Com respeito à rede PMD, temos que como resultado do procedimento de validação cruzada 25-fold, uma rede PMD apenas apresentou o menor erro de validação em comparação com as 25 redes resultantes do procedimento. Aplicamos esta rede no conjunto de teste e obtivemos um erro de 12.9% no conjunto de teste. A evolução da taxa de erro no treinamento, validação e teste desta rede é ilustrada na Figura 6.30.

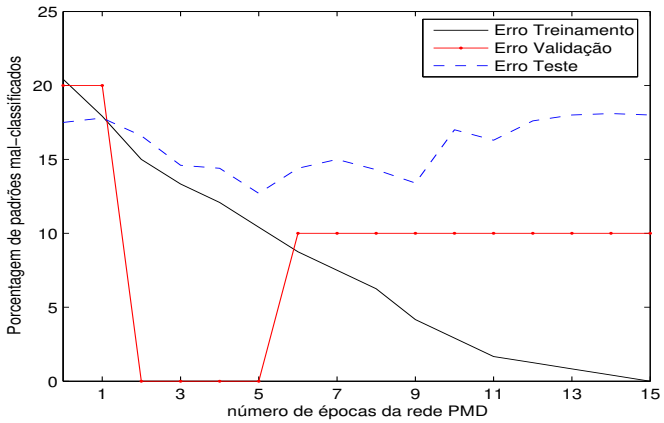

Fig. 6.30: Porcentagem de padrões mal-classificados pelo número de épocas da rede PMD resultante do procedimento de validação 25-fold

A Figura 6.31 ilustra o resultado do procedimento de validação cruzada 25-fold para a rede MRL.

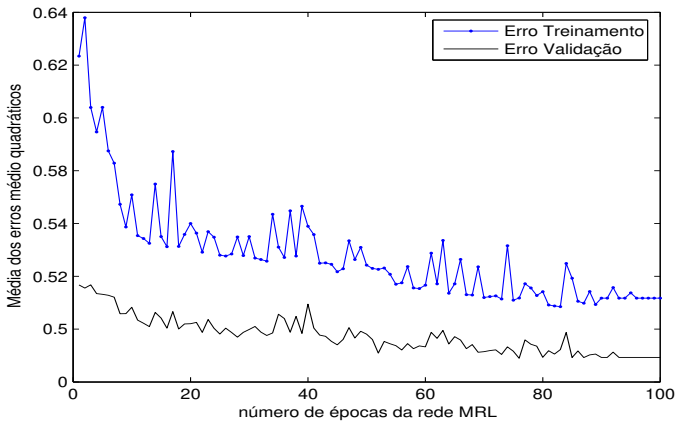

Fig. 6.31: Média dos erros médio quadráticos de padrões malclassificados pelo número de épocas da rede MRL no problema sintético.

Podemos observar que inicialmente nas primeiras iterações a média dos erros médios quadráticos

é alta e a partir da época 85 a média dos erros de validação começa a se estabilizar mas apresenta alguns picos que são uma consequência da descontinuidade da variável posto $r_k^{(l)}$. A rede MRL forneceu um erro de 26.4% no conjunto de teste, como pode ser observado na Tabela 6.7. Aplicando o procedimento de validação cruzada 25-fold para a rede MMNN treinada com algoritmo backpropagation, podemos observar pela Figura 6.32, que apesar da média dos erros de validação apresentar-se relativamente alta, na média, o erro médio quadrado decai ao longo das iterações e se estabiliza por volta da época 200.

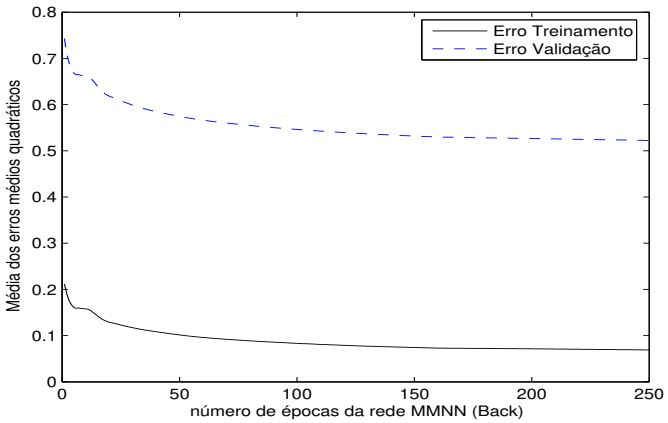

Fig. 6.32: Média dos erros médio quadráticos de padrões malclassificados pelo número de épocas da rede MMNN (Back) no problema sintético.

A rede MMNN (BP) apresentou um erro de 26.8% no conjunto de teste de acordo com a Tabela 6.7. A Figura 6.33 representa o resultado do procedimento de validação cruzada 25-fold para a rede MMNN(AG). Podemos observar que a média dos erros de validação decai à medida que o número de épocas avança no tempo. A rede MMNN (AG) apresentou um erro de 33% no conjunto de teste, como pode ser observado na Tabela 6.7.

Com respeito à rede FLNN, interrompemos o treinamento na época 23 pois podemos observar, a partir da Figura 6.33 que a média dos erros de validação a partir da época 23 começa a subir. Fazemos este procedimento de parada antecipada para garantir boa generalização da rede FLNN. A rede FLNN apresentou um erro de 10% no conjunto de teste, de acordo com a Tabela 6.7.

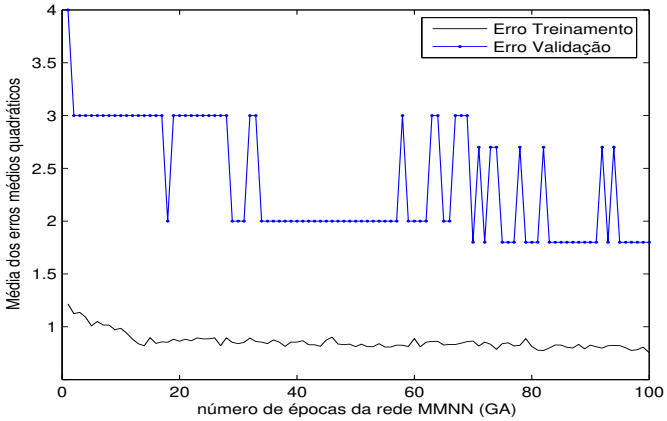

Fig. 6.33: Média dos erros médio quadráticos de padrões malclassificados pelo número de épocas da rede MMNN (AG) no problema sintético.

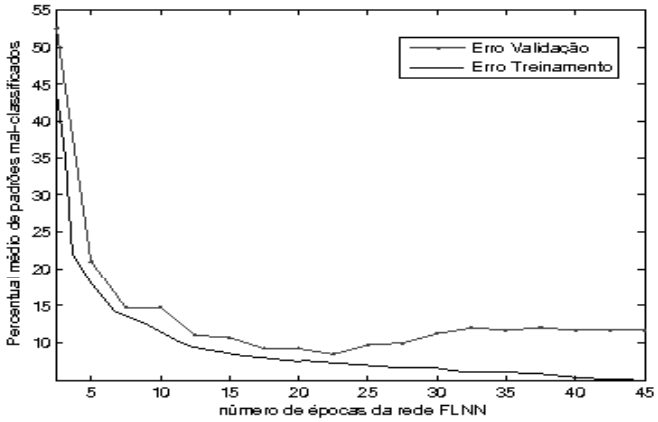

Fig. 6.34: Percentual médio de padrões mal-classificados pelo número de épocas da rede FLNN no problema sintético

Finalmente, aplicando o procedimento de validação cruzada 25-fold à rede MLP, temos que a mesma apresentou um resultado compatível com o resultado obtido na seção 6.1.1. Notamos que a média do erro de validação decai ao longo das épocas de acordo com a Figura 6.35, indicando que a rede generaliza bem. A rede MLP apresentou um erro de $11,6\%$ no conjunto de teste como pode ser observado na Tabela 6.7.

De um modo geral, temos que os resultados obtidos, no problema sinético, pelos modelos de redes neurais considerados, são parecidos com os resultados obtidos na seção 6.1.1, porque estes experimentos realizados utilizando validação cruzada visaram simplesmente tratar a generalização das redes de modo a tornar a escolha de conjuntos de treinamento, validação e teste o mais arbitrária possível.

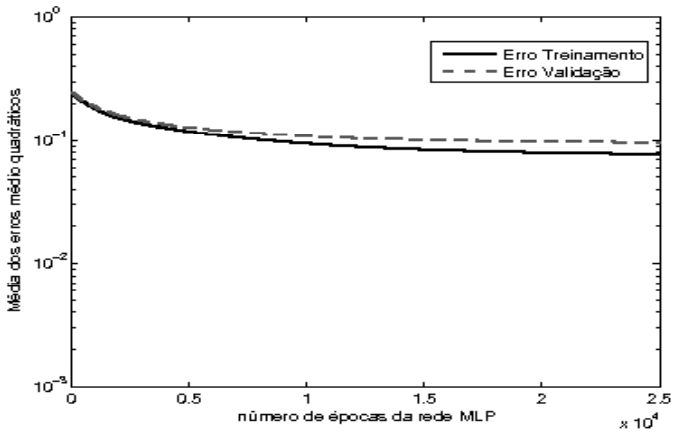

Fig. 6.35: Média dos erros médio quadráticos de padrões malclassificados pelo número de épocas da rede MLP no problema sintético.

Tab. 6.7: Porcentagem de padrões malclassificados no problema sintético utilizando validação cruzada 25-fold para o Treinamento e Teste.

Rede Neural	Treinamento (%)	Teste (%)
PM	0	11.2
PMD	0	12.9
MRL	13.1	26.4
MMNN(BP)	15	26.8
MMNN(AG)	14.6	33
FLNN	0	10
MLP	5.1	11.6

6.4.2 Resultados utilizando validação cruzada no problema de diabetes nos índios Pima

O conjunto de dados do problema diabetes nos índios Pima, disponível em [36], fornece 200 padrões para uma fase de treinamento e 332 padrões para uma fase de teste. Para todos os modelos de redes neurais que consideramos nós aplicamos a validação cruzada 20-fold no problema de diabetes nos índios Pima. Ou seja, o conjunto de treinamento, de tamanho $n_T = 200$, é dividido em 20 partes, cada qual contendo 10 padrões distintos e aplica-se o procedimento de validação cruzada 20-fold. Devemos ressaltar também que utilizamos os mesmos parâmetros definidos na seção 6.21 para a definição das redes MRL, MMNN (BP), MMNN(AG) e MPL. As Figura 6.36 e 6.37 representam o resultado da validação cruzada 20-fold para as redes PM e PMD no problema de diabetes nos índios Pima, respectivamente.

Na Figura 6.36, podemos observar que, para a rede PM, a média dos erros de validação

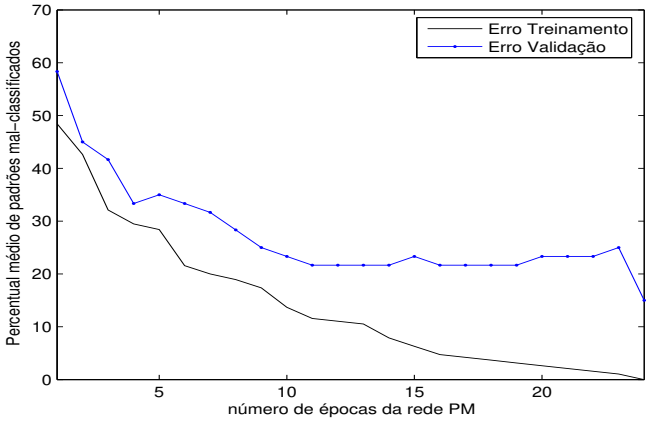

Fig. 6.36: Percentual médio de padrões malclassificados pelo número de épocas da rede PM no problema de diabetes nos índios Pima.

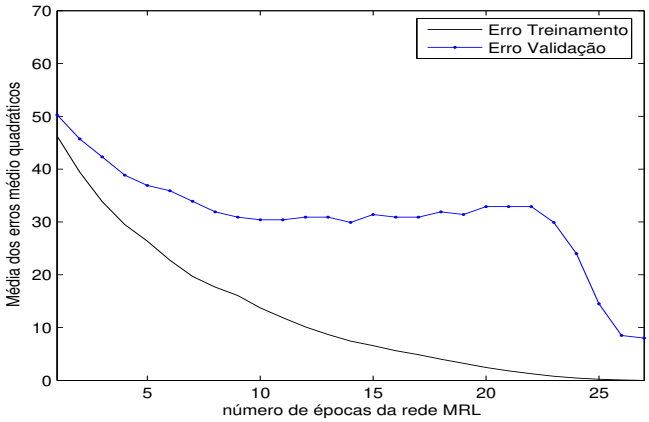

Fig. 6.37: Percentual médio de padrões malclassificados pelo número de épocas da rede PMD no problema de diabetes nos índios Pima.

apresenta um decrecimento ao longo do tempo. Observamos na Figura 6.37 que a rede PMD apresenta um decrescimento da média dos erros de validação até a décima época, depois a média destes erros de validação se estabiliza até a vigésima época, aproximadamente, e em seguida notamos um descrescimento contínuo da média dos erros de validação, até a época 27. Devemos ressaltar que também aplicamos o procedimento de parada antecipada quando o erro de validação da rede selecionada, do procedimento de validação cruzada, começava a subir. Este comportamento das redes PM e PMD, considerando o procedimento de validação cruzada 20-fold, revela que estas redes têm boa capacidade de generalização como pode ser observado na Tabela 6.8 para o problema de diabetes nos índios Pima. Neste procedimento de validação cruzada 20-fold, temos que a rede PM e a rede PMD apresentaram um erro de 20% e 28% no

conjunto de teste, respectivamente.

A Figura 6.38 ilustra o resultado do procedimento de validação cruzada 20-fold para a rede MRL no problema de diabetes nos índios Pima.

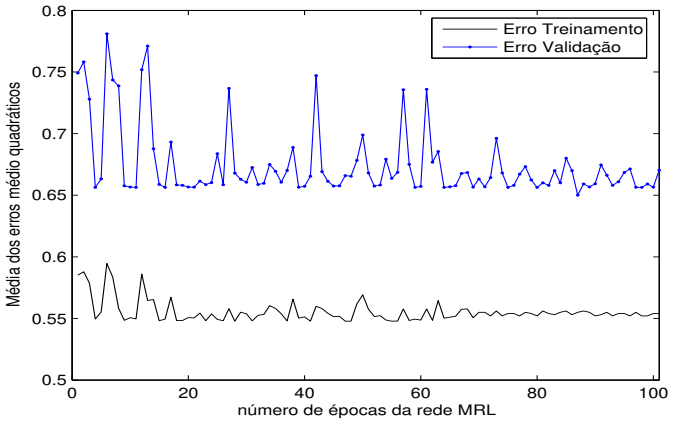

Fig. 6.38: Média dos erros médio quadráticos de padrões malclassificados pelo número de épocas da rede MRL no problema de diabetes nos índios Pima.

Podemos observar que, inicialmente, nas primeiras iterações, a média dos erros médios quadráticos apresenta alguns picos altos que são provocados pela descontinuidade da variável posto $r_k^{(l)}$ e a partir da época 75, aproximadamente, a média dos erros de validação começa a se estabilizar mas ainda apresenta alguns picos que são uma consequência da descontinuidade da variável posto $r_k^{(l)}$. A rede MRL forneceu um erro de 36.4% no conjunto de teste, como pode ser observado na Tabela 6.8. Aplicando o procedimento de validação cruzada 20-fold para a rede MMNN treinada com algoritmo backpropagation, podemos observar pela Figura 6.39, que a média dos erros de validação decrescem até a época 150 e a partir daí se estabiliza até a época 200.

A rede MMNN (BP) apresentou um erro de 39.2% no conjunto de teste de acordo com a Tabela 6.8. A Figura 6.40 representa o resultado do procedimento de validação cruzada 20-fold para a rede MMNN(AG). Podemos observar que a média dos erros de validação decai até a época 40 e se estabiliza até a época 100. A rede MMNN (AG) apresentou um erro de 33.3% no conjunto de teste, como pode ser observado na Tabela 6.8.

A Figura 6.41 representa a evolução da taxa percentual média de erros de classificação para a rede FLNN, no problema de diabetes nos índios Pima. Observamos que esta taxa sempre decresce ao longo das épocas da rede FLNN, indicando que a rede generaliza bem. A rede FLNN apresentou um erro de 15.1% no conjunto de teste, de acordo com a Tabela 6.8.

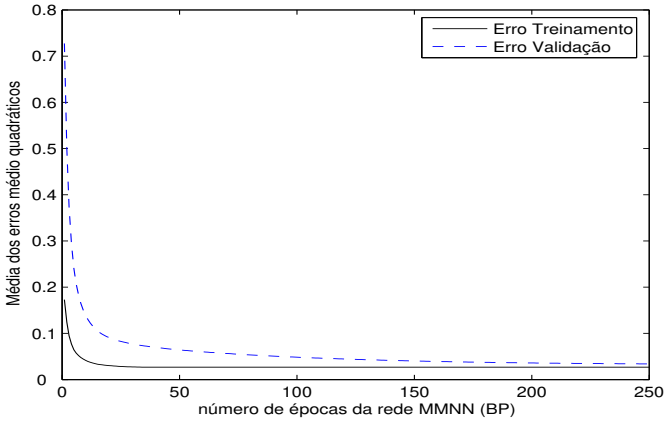

Fig. 6.39: Média dos erros médio quadráticos de padrões malclassificados pelo número de épocas da rede MMNN (BP) no problema de diabetes nos índios Pima.

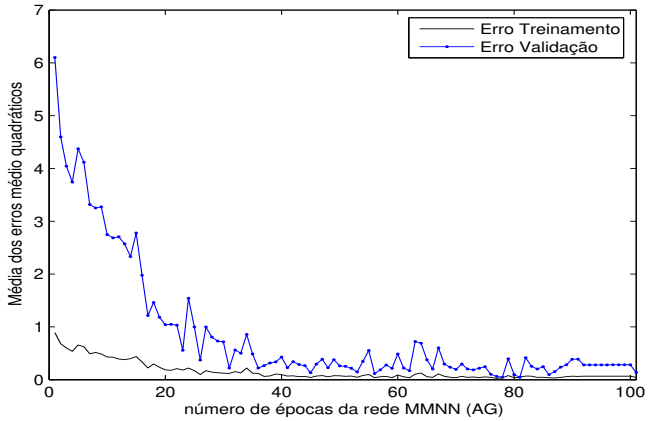

Fig. 6.40: Média dos erros médio quadráticos de padrões malclassificados pelo número de épocas da rede MMNN (AG) no problema de diabetes nos índios Pima.

Aplicamos o procedimento de validação cruzada 20-fold para a rede MLP também. Notamos que a média do erro de validação decai inicialmente e se estabiliza a partir da época 1500, de acordo com a Figura 6.42, indicando que a rede generaliza bem. A rede MLP apresentou um erro de 21% no conjunto de teste como pode ser observado na Tabela 6.8.

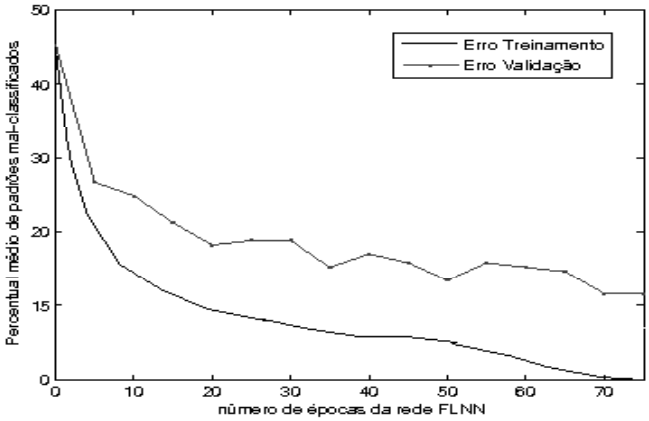

Fig. 6.41: Percentual médio de padrões mal-classificados pelo número de épocas da rede FLNN no problema de diabetes nos índios Pima

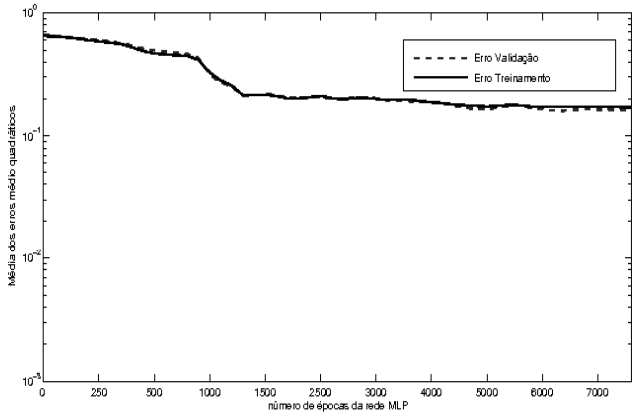

Fig. 6.42: Percentual médio de padrões mal-classificados pelo número de épocas da rede MLP no problema de diabetes nos índios Pima

Tab. 6.8: Porcentagem de padrões malclassificados no problema de diabetes nos índios Pima para o Treinamento e Teste.

Rede Neural	Treinamento (%)	Teste (%)
PM	0	20
PMD	0	28
MRL	33.4	36.4
MMNN(BP)	32	39.2
MMNN(AG)	24	33.3
FLNN	0	15.1
MLP	23.6	21

Capítulo 7

Conclusão

Em termos gerais, podemos dizer que esta dissertação teve por motivação fornecer um resumo a respeito da caracterização de redes neurais morfológicas no contexto da teoria de reticulados completos [3, 40, 46, 47, 60] e apresentar uma comparação destes modelos de redes neurais morfológicas com a rede neural artificial MLP em relação aos problemas de classificação considerados. Definimos as unidades de processamento dos modelos morfológicos estudados como sendo um neurônio artificial generalizado e devemos ressaltar que podemos considerar a rede FLNN pertencente à classe de redes neurais morfológicas devido ao fato que a função de ordem parcial nebulosa $p(., \mathbf{w}) : \mathbb{L} \longrightarrow [0, 1]$(função de agregação dos neurônios da rede FLNN), para um vetor de pesos sinápticos fixo $\mathbf{w} \in \mathbb{L}$ representa uma anti-dilatação e uma anti-erosão [65]. Em relação aos algoritmos de aprendizagem, propusemos modificações interessantes no algoritmo construtivo de Sussner [60] que foram a construção dos conjuntos C_i, definidos na equação (5.10) e ilustrados na Figura 5.2, e a determinação de uma caixa $Q' \subseteq Q$ que satisfaz a hipótese de ter o maior volume e conter o maior número de padrões da classe C_1 que estão mal-classificados. Estas modificações resultaram em bons resultados nos problemas de classificação em comparação com os modelos implementados, de um modo geral.

De fato o esforço computacional está associado às unidades de processamento e número de épocas na fase de treinamento. Portanto definimos unidades de processamento e a noção de época para alguns modelos morfológicos que em princípio não tinham uma definição precisa na literatura. O cálculo realizado por um neurônio morfológico não envolve multiplicações, apenas as operações de máximo ou mínimo e adição. Isto fornece cálculos mais rápidos [50] e fácil implementação de hardware [41]. Outra vantagem que as operações baseadas em reticulados possuem, é que existe uma conexão natural entre cálculos baseados em reticulados e teoria de conjuntos fuzzy [26], fazendo com que redes neurais baseadas em reticulados sejam mais flexíveis para lidar com tipos de dados bem gerais. Com respeito às redes MP, MPD e FLNN, temos que os algoritmos de aprendizagem operam em um número finito de passos e não têm

problemas de convergência.

Com base nos resultados experimentais obtidos, observamos que a rede FLNN apresentou os melhores resultados com respeito ao erro de classificação no teste nos três problemas. A rede MLP e a rede MP apresentaram porcentagem de padrões incorretamente classificados similares, onde a rede MRL e a rede MMNN exibiram o pior desempenho dentre os modelos que nós testamos. Verificamos que o aprendizado do perceptron morfológico requer um número maior de unidades de processamento em comparação com as redes MRL e MMNN. Entretanto esta rede, treinada com o algoritmo construtivo de Sussner [60], produziu resultados comparáveis com a FLNN no problema sintético e no problema de segmentação de imagens. A razão pela qual a rede MRL não demonstrou um desempenho satisfatório nos problemas considerados pode estar associada a descontinuidade da variável $r_k^{(l)}$. Com relação à rede MMNN, temos que a sua performance em problemas de classificação depende fortemente da seleção das características iniciais do conjunto de dados e das correspondentes formas das regiões de decisão. Neste sentido, acreditamos que seria necessário investigar o pré-processamento dos dados e o comportamento da rede MMNN em um número maior de problemas. Com respeito aos experimentos utilizando validaçã cruzada para o problemas sintético e para o problema de diabetes nos índios Pima, temos que os resultados foram razoavelmente compatíveis com os resultados obtidos nas seções 6.1.1 e 6.2.1, respectivamente. O procedimento de validação cruzada fornece uma medida do erro de generalização dos modelos e ameniza o problema da não arbitrariedade da escolha de conjuntos de treinamento, validação e teste, desta maneira podemos concluir que os modelos morfológicos PM, PMD e FLNN além de apresentarem propriedades de arquiteturas, regras de treinamento e convergência interessantes, quando comparados com a rede MLP, também fornecem uma capacidade de generalização comparável com a MLP. Os modelos MRL, MMNN (BP) e MMNN (AG) apresentaram os piores desempenhos com respeito a generalização, e este fato pode ser mais conclusivo com os experimentos de validação cruzada.

Esta dissertação representa um primeiro trabalho que compara estes diferentes modelos de redes neurais morfológicas em problemas de classificação. Acreditamos que este texto deve desempenhar o papel de uma literatura introdutória didática às redes neurais morfológicas, de um modo geral, para os futuros estudantes neste ramo de pesquisa e também apresentar a importância das teorias matemáticas discutidas para a fundamentação destes modelos de redes neurais morfológicas. De fato, falta conduzir mais pesquisa com respeito, por exemplo, à aplicabilidade destes modelos de redes neurais morfológicas à problemas de múltiplas classes e considerar conjuntos de treinamento com atributos não contínuos. Podemos citar também a possibilidade de unir os algoritmos de treinamento da rede FLNN e do perceptron morfológico, bem como a aproximação de funções usando a decomposição de Banon e Barrera [3] além de outras questões teóricas. Esperamos que os resultados teóricos e experimentais apresentados

nesta dissertação possam contribuir em trabalhos científicos em geral.

Bibliografia

[1] Araújo, R., Madeiro, R., Sousa, R., and Pessoa, L. Modular morphological neural network training via adaptative genetic algorithm for design translation invariant operators. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (Toulouse, France, 2006), vol. 2.

[2] Araújo, R., Madeiro, R., Sousa, R., Pessoa, L., and Ferreira, T. An evolutionary morphological approach for financial time series forecasting. In *Proceedings of the IEEE Congress on Evolutionary Computation* (Vancouver,Canada, 2006), pp. 2467–2474.

[3] Banon, G., and Barrera, J. Decomposition of mappings between complete lattices by mathematical morphology, part 1. general lattices. *Signal Processing 30*, 3 (Feb. 1993), 299–327.

[4] Baum, E. What size of neural net gives valid generalization? *Neural Computation 1*, 4 (1989), 151–160.

[5] Beale, R., and Fiesler, E., Eds. *Handbook of Neural Computation*. Institue of Physics Publishing and Oxford University Press, 1997.

[6] Birkhoff, G. *Lattice Theory*, 3 ed. American Mathematical Society, Providence, 1993.

[7] Birkhoff, G., and Lipson, J. Heterogeneous algebras. *Journal of Combinatorial Theory 8*, 115-133 (1970).

[8] Bishop, C. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

[9] Bryson, A., and Ho, Y. *Applied Optimal Control*, rev ed. John Wiley and Sons, October 1979.

[10] Carpenter, G., and Grossberg, S. A massively parallel architecture for self-organizing neural pattern recognition machine. *Comput. Vision, Graphics, and Image Understanding 37* (1987), 54–115.

[11] Cuninghame-Green, R. *Minimax Algebra: Lecture Notes in Economics and Mathematical Systems 166*. Springer-Verlag, New York, 1979.

[12] Davidson, J. Foundation and applications of lattice transforms in image processing. In *Advances in Eletronic and Electron Physics*, P. Hawkes, Ed., vol. 84. Academic Press, New York, NY, 1992, pp. 61–130.

[13] Davidson, J., and Ritter, G. A theory of morphological neural networks. In *Digital Optical Computing II* (July 1990), vol. 1215 of *Proceedings of SPIE*, pp. 378–388.

[14] de Sousa, R. P., de Carvalho, J. M., de Assis, F. M., and Pessoa, L. F. C. Designing translation invariant operations via neural network training. *Proceedings of the IEE International Conference on Image Processing 1* (2000), 908–911.

[15] Deng, T., and Heijmans, H. Grey-scale morphology based on fuzzy logic. *Journal of Mathematical Imaging and Vision 16*, 2 (Mar. 2002), 155–171.

[16] Gader, P. D., Khabou, M., and Koldobsky, A. Morphological regularization neural networks. *Pattern Recognition, Special Issue on Mathematical Morphology and Its Applications 33*, 6 (June 2000), 935–945.

[17] Gen, M., and Cheng, R. *Genetic Algorithms and Engineering Optimization*. John Wiley, New York, NY, 2000.

[18] Godoi, A. Aprendizado supervisionado em fuzzy lattice neural networks com aplicações em reconhecimento de padrões. Relatório de iniciação científica, Universidade Estadual de Campinas, IMECC, 2005.

[19] Graña, M., Gallego, J., Torrealdea, F. J., and D'Anjou, A. On the application of associative morphological memories to hyperspectral image analysis. *Lecture Notes in Computer Science 2687* (2003), 567–574.

[20] Hadwiger, H. *Vorlesungen Über Inhalt, Oberfläche und Isoperimetrie*. Springer-Verlag, Berlin, 1957.

[21] Hassoun, M. H. *Fundamentals of Artificial Neural Networks*. MIT Press, Cambridge, MA, 1995.

[22] Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NJ, 1999.

[23] Hebb, D. *The Organization of Behavior*. John Wiley & Sons, New York, 1949.

[24] Heijmans, H. *Morphological Image Operators*. Academic Press, New York, NY, 1994.

[25] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences 79* (Apr. 1982), 2554–2558.

[26] Kaburlasos, V., and Petridis, V. Fuzzy lattice neurocomputing (FLN) models. *Neural Networks 13* (2000), 1145–1170.

[27] Khabou, M., and Gader, P. Automatic target detection using entropy optimized shared-weight neural networks. *IEEE Transactions on Neural Networks 11*, 1 (Jan. 2000), 186–193.

[28] Khabou, M., Gader, P., and Keller, J. Ladar target detection using morphological shared-weight neural networks. *Machine Vision and Applications 11*, 6 (Apr. 2000), 300–305.

[29] Klir, G. J., and Yuan, B. *Fuzzy Sets and Fuzzy Logic; Theory and Applications*. Prentice Hall, Upper Saddle River, N. Y., 1995.

[30] Lippmann, R. P. An introduction to computing with neural nets. 4–20.

[31] Luenberger, D. *Linear and nonlinear programming*. Adison-Wesley Publishing Company, 1986.

[32] Matheron, G. *Random Sets and Integral Geometry*. Wiley, New York, 1975.

[33] McCulloch, W., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics 5* (1943), 115–133.

[34] Minkowski, H. *Gesammelte Abhandlungen*. Teubner Verlag, Leipzig-Berlin, 1911.

[35] Minsky, M., and Papert, S. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

[36] of machine learning databases, U. R. http://www.ics.uci.edu/∼mlearn/ MLRepository.html.

[37] Page, B. R. H. http://www.stats.ox.ac.uk/pub/PRNN/.

[38] Pessoa, L., and Maragos, P. Neural networks with hybrid morphological/rank/linear nodes: a unifying framework with applications to handwritten character recognition. *Pattern Recognition 33* (2000), 945–960.

[39] Pessoa, L. F. C., and Maragos, P. Mrl-filters: A general class of nonlinear systems and their optimal desingn for image processing. *IEEE Transactions on Image Processing 7*, 7 (July 1998).

[40] Petridis, V., and Kaburlasos, V. Fuzzy lattice neural network (FLNN): a hybrid model for learning. *IEEE Transactions on Neural Networks 9*, 5 (Sept. 1998), 877–890.

[41] Porter, R., Harvey, N., Perkins, S., Theiler, J., Brumby, S., Bloch, J., Gokhale, M., and Szymanski, J. Optimizing digital hardware perceptrons for multi-spectral image classification. *Journal of Mathematical Imaging and Vision 19*, 2 (2003), 133–150.

[42] Prechelt, L. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks 11*, 4 (1998), 761–767.

[43] Raducanu, B., Graña, M., and Albizuri, X. F. Morphological scale spaces and associative morphological memories: Results on robustness and practical applications. *Journal of Mathematical Imaging and Vision 19*, 2 (2003), 113–131.

[44] Reed, R., and Marks, R. *Neural Smithing: Supervised Learning in Feedforward Neural Networks*. The MIT Press, Cambridge, MA, 1999.

[45] Ripley, B. *Pattern Recognition and Neural Networks*. Press Sindicate of the university of Cambridge, New York, 1996.

[46] Ritter, G. X., and Sussner, P. An introduction to morphological neural networks. In *Proceedings of the 13th International Conference on Pattern Recognition* (Vienna, Austria, 1996), pp. 709–717.

[47] Ritter, G. X., and Sussner, P. Morphological neural networks. In *Intelligent Systems: A Semiotic Perspective; Proceedings of the 1996 International Multidisciplinary Conference* (Gaithersburg, Maryland, 1996), pp. 221–226.

[48] Ritter, G. X., and Sussner, P. Morphological perceptrons. In *ISAS'97, Intelligent Systems and Semiotics* (Gaithersburg, Maryland, 1997).

[49] Ritter, G. X., Sussner, P., and de Leon, J. L. D. Morphological associative memories. *IEEE Transactions on Neural Networks 9*, 2 (1998), 281–293.

[50] Ritter, G. X., and Urcid, G. Lattice algebra approach to single-neuron computation. *IEEE Transactions on Neural Networks 14*, 2 (March 2003), 282–295.

[51] Ritter, G. X., Wilson, J. N., and Davidson, J. L. Image algebra: An overview. *Computer Vision, Graphics, and Image Processing 49*, 3 (Mar. 1990), 297–331.

[52] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review 65* (1958), 386–408.

[53] Rubesam, A. Perceptrons com aplicações em reconhecimento de padrões. Relatório de iniciação científica, Universidade Estadual de Campinas, IMECC, 2001.

[54] Schaffer, C. Selecting a Classification Method by Cross-Validation. In *Fourth Intl. Workshop on Artificial Intelligence & Statistics* (January 1993), pp. 15–25.

[55] Serra, J. Mathematical morphology and cmm : a historical overview. Avaliable at: http://cmm.ensmp.fr/Recherche/pages/nav0b.htm.

[56] Serra, J. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.

[57] Sternberg, S. Grayscale morphology. *Computer Vision Graphics and Image Processing 35* (1986), 333–355.

[58] Suarez-Araujo, C., and Ritter, G. Morphological neural networks and image algebra in artificial perception systems. In *Image Algebra and Morphological Image Processing III* (San Diego, CA, July 1992), vol. 1769 of *Proceedings of SPIE*, pp. 128–142.

[59] Sussner, P. Results on morphological neural networks. Unpublished manuscript, Universidade Estadual de Campinas, IMECC.

[60] Sussner, P. Morphological perceptron learning. In *Proceedings of IEE ISIC/CIRA/ISAS Joint Conference* (Gaithesburg, MD, 1998), pp. 447–482.

[61] Sussner, P. Perceptrons. In *Wiley Encyclopedia of Electrical and Electronics Engineering*, J. G. Webster, Ed. John Wiley and Sons, Inc., New York, 1999, pp. 44–59.

[62] Sussner, P. Observations on morphological associative memories and the kernel method. *Neurocomputing 31* (Mar. 2000), 167–183.

[63] Sussner, P. Associative morphological memories based on variations of the kernel and dual kernel methods. *Neural Networks 16*, 5 (July 2003), 625–632.

[64] Sussner, P. Generalizing operations of binary morphological autoassociative memories using fuzzy set theory. *Journal of Mathematical Imaging and Vision 9*, 2 (Sept. 2003), 81–93. Special Issue on Morphological Neural Networks.

[65] Sussner, P. Algoritmos de aprendizado para redes neurais morfológicas. Technical report, Universidade Estadual de Campinas, IMECC, 2006.

[66] Sussner, P., and Valle, M. A brief account of the relations between gray-scale mathematical morphologies. In *Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)* (Natal, Brazil, October 2005), pp. 79 – 86.

[67] Sussner, P., and Valle, M. Gray scale morphological associative memories. *IEEE Transactions on Neural Networks 17*, 3 (May 2006), 559–570.

[68] Sussner, P., and Valle, M. Implicative fuzzy associative memories. *IEEE Transactions on Fuzzy Systems 14*, 6 (Dec. 2006), 793–807.

[69] Valle, M., Sussner, P., and Gomide, F. Introduction to implicative fuzzy associative memories. In *Proceedings of the IEEE International Joint Conference on Neural Networks* (Hungary, July 2004), pp. 925 – 931.

[70] Valle, M. E. MATLAB Source Code for MRL and FLNN, Available at `http://www.ime.unicamp.br/~mevalle/`.

[71] Werbos, P. J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Committee on Applied Mathematics, Havard University, Cambridge, Mass, 1974.

[72] Widrow, W., and Hoff, M. Adaptive switching circuits. *WESCON Convention Record* (1960), 96–104.

[73] YU, A., G. M., and POGGIO, T. Biophysiologically plausible implementations of the maximum operation. *Neural Computation. 14*, 12 (2002), 2857–2881.