MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-11279-PRP/242

USO DE REDES NEURAIS EM VISÃO COMPUTACIONAL E PROCESSAMENTO DE IMAGENS

José Demisio Simões da Silva

Relatório de atividades em estágio SPE/RHAE (CNPq-Processo nº 260.048/95-0), realizado no Nuclear Engineering Department da Universidade of Tennessee, Knovville, USA

INPE
São José dos Campos
2004

Identificação do Bolsista:

Nome: José Demisio Simões da Silva

No. do Processo: 260.048/95-0

Instituição:

Instituto Nacional de Pesquisas Espaciais - INPE

Departamento:

Laboratório de Computação e Matemática Aplicada - LAC

Endereços: Residencial: Rua Helena David Neme, 222 Ap 71

São Dimas - São José dos Campos - SP

CEP 12245-310

Instituição: Av. dos Astronautas, 1758

Jardim da Granja - São dos Campos - SP

CEP 12227-010

Nível de Treinamento:

SPE - Estágio de Especialização no Exterior

Título do Projeto de Tese/Pesquisa:

Uso de Redes Neurais em Visão Computacional e Processamento de Imagens.

Nome do Orientador: Dr. Robert E. Uhrig

Local: Nuclear Engineering Department

The University of Tennessee, Knoxville, USA

São José dos Campos, 07 de Julho de 1997

José Demisio Simões da Silva

Processo : 260.048/95-0

RELATÓRIO

Resumo: Este documento relata as atividades desenvolvidas durante a realização de Estágio de Aperfeiçoamento no Exterior, realizado no "Nuclear Engineering Department" na "The University of Tennessee", nos Estados Unidos. No relatório, são mencionados os estudos bibliográficos realizados e os experimentos efetuados para adquirir maior proficiência no uso de redes neurais. O objetivo principal do trabalho foi adquirir maior familiarização com o paradigma e os algoritmos de redes neurais artificiais existentes, procurando gerar competência para o uso de redes neurais em tarefas de processamento de imagens e visão computacional.

Introdução

O Estágio teve como objetivo adquirir maior proficiência no uso de redes neurais como ferramenta computacional em aplicações de visão computacional, processamento de imagens, e áreas correlatas, buscando capacitação técnica para a realização de pesquisas, que contribuam para o progresso e desenvolvimento das áreas de Visão Computacional e Processamento de Imagens através do uso de ferramentas e métodos computacionais modernos.

As áreas de Visão Computacional e Processamento de Imagens são exploradas por diversos domínios de aplicação, tais como: indústria, ciências espaciais, medicina, etc. As pesquisas realizadas por novos métodos de resolução de problemas nas áreas, tentam desenvolver algoritmos computacionais com as seguintes características: robustez, desempenho e baixo custo computacional.

Na literatura especializada se encontra muitos trabalhos que exaltam procedimentos, métodos e algoritmos para a resolução de problemas, empregando métodos convencionais e modernos. Os objetivos das áreas são tratamento e extração de informação quantitativa e qualitativa de imagens. Os problemas são tratados sob várias abordagens diferentes, cada uma enfatizando técnicas e métodos que tentam superar os problemas de outra.

O surgimento de novas metodologias, teorias e ferramentas faz as pesquisas convergirem naturalmente em busca de soluções para problemas ainda não tratados por falta de recursos computacionais e modelos, e para aqueles que exigem muitos esforços computacionais. Neste sentido, uma linha de pesquisa tenta dotar as máquinas com as mesmas capacidades de processamento dos humanos, tanto para o melhoramento dos dados, como para a extração automática de informações que eles representam.

Vários métodos estão disponíveis para este fim. Há pesquisadores que abordam este problema como um problema de reconhecimento de padrões sob hipótese cognitiva, isto é, os especialistas na aplicação têm conhecimento a priori da informação que pode estar nos dados, como por exemplo em imagens. Em geral estes algoritmos tomam decisões com base em estatísticas obtidas a partir dos dados brutos (níveis de cinza das imagens), que envolve uma fase de treinamento na obtenção do espaço de atributos e formação das classes.

No caso de Processamento de Imagens, a análise de uma imagem não envolve apenas a informação presente, além dela, se usa conhecimento armazenado anteriormente. É o caso, por exemplo, de aplicações em Sensoriamento Remoto. Os especialistas de Sensoriamento Remoto lidam com uma quantidade de informação grande durante a inferência de informações úteis da imagem. Em geral a fase de treinamento para classificação e interpretação das imagens envolve várias horas de análise dos dados das imagens, bem como o levantamento da verdade terrestre. Apenas com a experiência tais especialistas começam a depender menos de informação complementar para certos objetivos. Evidentemente, eles precisam estar atentos para as alterações no uso do solo, que pode mudar com o tempo tornando dinâmica a tarefa de análise de imagens em sensoriamento remoto.

A base para a extração de informações de uma imagem depende exclusivamente do treinamento dado ao especialista. A interpretação de fotos aéreas ou de satélites, envolve o treinamento exaustivo de novos usuários na interpretação dos níveis de cinza da imagem. A interpretação depende da acuidade visual do observador e das condições de iluminação da cena e do processo de aquisição das imagens.

A interpretação é antecedida pela fase de extração de atributos primitivos, como intensidades de níveis de cinza, bordas, contornos e textura, atributos esses que são naturalmente procurados pelo sistema visual humano quando exposto às imagens. Pode-se ver então que a capacidade visual dos humanos pode fazer a diferença entre os observadores. A fase de extração de atributos para interpretação é desenvolvida através de algoritmos de Visão Computacional. Para que as máquinas em Visão Computacional e Processamento de Imagens

tenham desempenho semelhante aos humanos na extração de atributos, faz-se necessário utilizar algoritmos que permitam a integração de dados de natureza diferente, combinando alto desempenho e baixo custo computacional.

Uma forma de alcançar tal objetivo é através do uso de técnicas de Inteligência Artificial que permite o desenvolvimento de algoritmos híbridos adequados para esta integração. Entretanto, a maioria das técnicas utilizadas exige explicitação do conhecimento para construção de inferências pelas máquinas. É necessário então se ter ferramentas que evitem a representação explícita do conhecimento, ou seja, que trabalhem diretamente sobre os dados brutos fornecidos, o tanto quanto possível.

A abordagem por Redes Neurais é adequada para tal tarefa, pois nela o conhecimento é mapeado nos padrões de conexões entre os diversos processadores das redes. O padrão de conexão nas redes representa o conhecimento necessário para a realização da tarefa, diminuindo a necessidade de explicitação dos vários aspectos da informação.

Redes Neurais são utilizadas em diversas áreas de aplicação na resolução de problemas envolvendo estimação de parâmetros e otimização, que são aspectos importantes em tarefas de visão computacional e processamento de imagens, onde as inferências são sistematicamente melhoradas de acordo com o aumento da precisão da informação armazenada. Com o objetivo de tornar as máquinas com a capacidade de inferência próxima à de um observador humano, quando exposto a uma imagem, as tarefas de visão computacional e de processamento de imagens poderiam explorar melhor os modelos conexionistas do que apenas os aspectos de modelagem matemática envolvidos.

Desenvolvimento do Estágio

Etapa 1. Contatos com grupos de pesquisa.

A primeira fase do plano de trabalho apresentado diz respeito a contatos com grupos de trabalhos. Durante o estágio foram mantidos contatos com: o grupo de Visão Computacional de Oakridge National Laboratory, o grupo de Visão Computacional do Departamento de Engenharia Elétrica da University of Tennessee e o grupo de pesquisadores que desenvolvem atividades de pesquisa utilizando redes neurais e lógica nebulosa no Departamento de Engenharia Nuclear da University of Tennessee.

Estes contatos permitiram um conhecimento amplo do nível de pesquisa que os grupos vêm desenvolvendo utilizando redes neurais. Os grupos de visão computacional têm trabalhado prioritariamente com tarefas de reconhecimento de padrões, enquanto o grupo do departamento de engenharia nuclear tem suas atividades voltadas para processamento digital de sinais.

Da observação do estágio dos trabalhos dos grupos conclui-se que, as tarefas desenvolvidas estão prioritariamente voltadas para a resolução de problemas. Apesar disso, devido a existência de tópicos abertos em redes neurais como: a definição da topologia mínima da rede a ser utilizada, o tamanho de um conjunto de aprendizagem a ser utilizado, a generalização da aprendizagem e o tempo para aprendizagem de uma rede, a pesquisa por algoritmos que melhorem o desempenho dos algoritmos em uso, também é objeto de pesquisa durante a realização dos trabalhos.

Durante o estágio foram realizados experimentos, implementando-se algoritmos computacionais para reduzir os conjuntos de treinamento através do uso de técnicas de clusterização. Os experimentos são comentados no decorrer deste relatório. Para se obter redes mais concisas possíveis também foi objeto de estudo no estágio o desenvolvimento e implementação de um algoritmo de crescimento dinâmico de uma rede neural.

Etapa 2. Pesquisa Bibliográfica em redes neurais buscando identificar as mais variadas aplicações nas áreas de visão computacional e processamento de imagens.

Vários trabalhos podem ser encontrados na literatura, que utilizam redes neurais na resolução de problemas em Visão Computacional e Processamento de Imagens. Apenas alguns trabalhos são mencionados ressaltando a variedade de problemas e de modelos que tem sido utilizados.

Uma das aplicações bastante citadas na literatura é o uso de redes neurais em detecção de bordas. As bordas em imagens representam descontinuidades ou mudanças abruptas na distribuição de intensidade, sendo essenciais no reconhecimento dos objetos.

Um primeiro trabalho é Srinivasan et al. (1994), que descrevem uma rede neural sensível a bordas em um campo receptivo largo, assumindo que na retina as bordas são detectadas por camadas neuronais iniciais. Além da detecção de bordas o experimento provê compressão de imagem, que é uma aplicação essencial para Processamento de Imagens.

A figura 2.1 ilustra a arquitetura do detector. A rede opera em um campo receptivo de tamanho *K x K*, com centro em *(x,y)* na imagem de *N x N*. A camada codificadora gera um vetor comprimido $\boldsymbol{q}_{xy}= [q_0 q_1 \ldots q_{L-1}]^T$ a partir da entrada $\boldsymbol{p}_{xy}=[p_0 p_1 \ldots p_k^2{}_{-1}]^T$. As entradas são as intensidades no campo receptivo. A segunda camada mapeia $\boldsymbol{q}_{xy}$ para um vetor de borda em 2D. $\boldsymbol{s}_{xy}= s_0\boldsymbol{i} + s_1\boldsymbol{j}$ dá a intensidade da borda ao longo de duas direções ortogonais em *(x,y)*, o que permite o cálculo da intensidade total e da direção da borda.

A função da camada codificadora é expressa por: $\boldsymbol{q}_{xy}= \boldsymbol{W}^t\boldsymbol{p}_{xy}$, onde $\boldsymbol{W} = [\boldsymbol{w}_0\boldsymbol{w}_1 \ldots \boldsymbol{w}_{L-1}]$ e cada $\boldsymbol{w}_i$ é um vetor coluna com os pesos que ligam cada neurônio ao campo receptivo. O treinamento determina a matriz de pesos que produz um vetor saída $\boldsymbol{q}_{xy}$ com o máximo de informação sobre bordas no campo. Os elementos de $\boldsymbol{q}_{xy}$ têm variância máxima sobre o conjunto de vetores de entrada e correlação mínima com outros elementos, que implica em máximo de informação e mínimo de sobreposição. O treinamento da camada codificadora usa imagens de círculos concêntricos, que variam em intensidade, orientação, curvatura e localização.

Os pesos têm valores iniciais aleatórios. Calcula-se o produto escalar do vetor de intensidade por cada vetor peso. Apenas o neurônio ganhador por inibição lateral tem o peso ajustado.

Para a maximização da variância de saída, modificam-se os pesos pela regra de Hebb $\boldsymbol{w}_l^{\tau+1} = \boldsymbol{w}_l{}^{\tau} + \eta_l{}^{\tau} q_l{}^{\tau} \boldsymbol{p}^{\tau}_{xy}$. η_l é a taxa de aprendizagem, que é gradualmente reduzida forçando o vetor de pesos para um valor final.

A interação entre os neurônios se dá através de conexões inibitórias. Para um neurônio $\boldsymbol{l}$, a entrada inibitória é $\sum_{n\neq l} vq_n{}^{\tau}$.

A aprendizagem tem dois ciclos iterativos ***(k)***. A saída de um neurônio por ciclo é dada por: $q_l{}^{\tau}(k)= \begin{cases} a_l{}^t, & a_l{}^t \geq \varphi \\ 0 & c.c. \end{cases}$. $a_l{}^{\tau}(k)$ é o nível de ativação e φ é um limiar. A ativação inicial é igual à entrada, e posteriormente é calculada por:

$$a_l^t(k) = \frac{\left[a_l^t(k-1)\right]}{\left[1+\sum_{n\neq l} vq_n^t(k-1)\right]} \quad k = 1,2,.....$$

As entradas inibitórias de outros neurônios suprimem continuamente a ativação. As inibições mútuas decrescem as saídas até apenas um neurônio (o ganhador) ter saída não nula.

Neste ponto se tem o equilíbrio e adaptam-se os pesos do ganhador. No próximo ciclo de treinamento, uma nova entrada é escolhida aleatoriamente e o processo se repete.

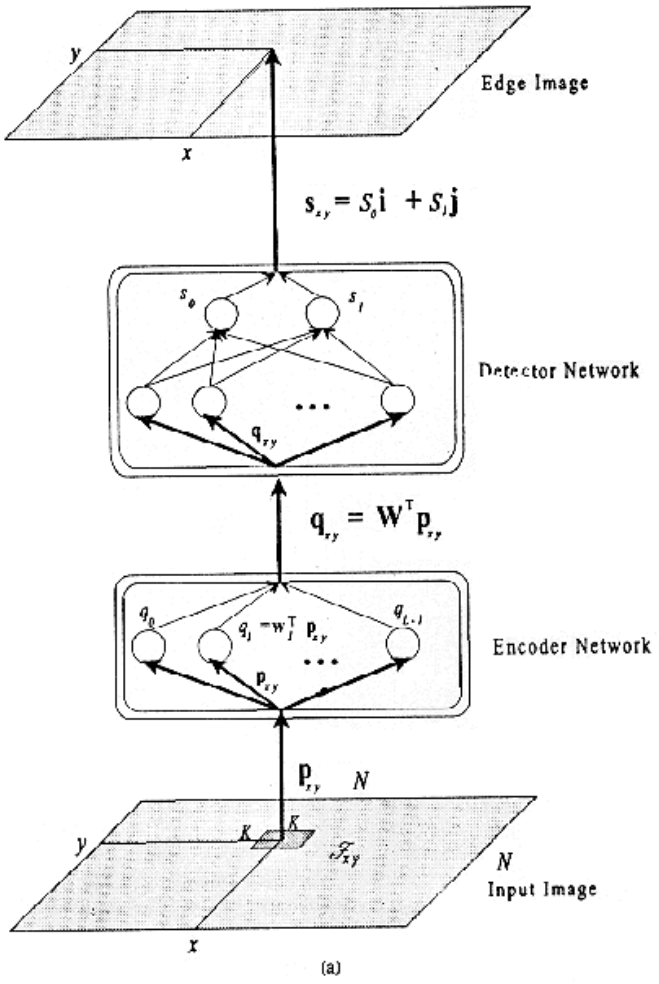


FIGURA 2.1 - Codificador e detector de bordas com rede neural.
FONTE: Sirinivasan et al. (1994, p. 1654).

As taxas de aprendizagem decrescem gradativamente com o treinamento com:

$$\eta_l^{\tau+1} = \begin{cases} \eta_l^{\tau}\alpha(1-\beta q_l^{\tau}), & \text{se o } l\text{-ésimo neurônio é o ganhador} \\ & \text{do } (\tau+1)\text{-ésimo ciclo de treinamento} \\ \eta_l^{\tau}, & \text{caso contrário} \end{cases}$$

α e β controlam o processo de decaimento, o que melhora a adaptabilidade da rede.

O treinamento acaba quando as taxas de aprendizagem têm valores menor que um limiar, por exemplo, *0.01*.

Cada elemento de um vetor de pesos, que liga um pixel *(x',y')* a um neurônio, é multiplicado em seguida por uma função envelope gaussiano com variância σ,

$$e^{\frac{1}{2\sigma^2}\left[(x-x')^2+(y-y')^2\right]}$$

que aumenta a contribuição de pixels na região central de raio σ, em relação aos pixels na periferia do campo receptivo, assim a informação de borda no vetor de saída é localizada.

O detector é uma rede "feed-forward" de duas camadas, como na figura 2.1. A camada escondida tem *M* neurônios e a entrada é o vetor $\boldsymbol{q}_{xy}$, saída do codificador. Cada camada de saída tem dois neurônios, que são os componentes de um vetor $\boldsymbol{s}_{xy}$ de bordas, $\boldsymbol{s}_{xy}=s_0\boldsymbol{i}+s_1\boldsymbol{j}$, na posição *(x,y)* na imagem. s_0 e s_1 são intensidades nas direções $\boldsymbol{i}$ e $\boldsymbol{j}$, sendo mutuamente perpendiculares. A intensidade resultante e a orientação da borda podem ser conseguidas por:

$$s_{xy} = (s_0 + s_1)^{1/2} \text{ e } \theta_{xy} = tan^{-1}(s_1/s_0).$$

Para cada pixel a rede mapeia um vetor $\boldsymbol{q}_{xy}$ num vetor de borda bidimensional $\boldsymbol{s}_{xy}$.

O treinamento da rede detectora é executado por um algoritmo de "back-propagation", com o objetivo de fazer a rede gerar um vetor de borda ideal para cada locação dada, mesmo na presença de ruído. Gera-se $\boldsymbol{q}_{xy}$ para cada ponto *(x,y)* na imagem de treinamento. Calcula-se um vetor $\boldsymbol{s}'_{xy}$ ideal. Estes pares de vetores formam o conjunto de treinamento da rede. No experimento Srinivasan et al. (1994) usaram uma imagem de círculos concêntricos com bordas em todas as orientações.

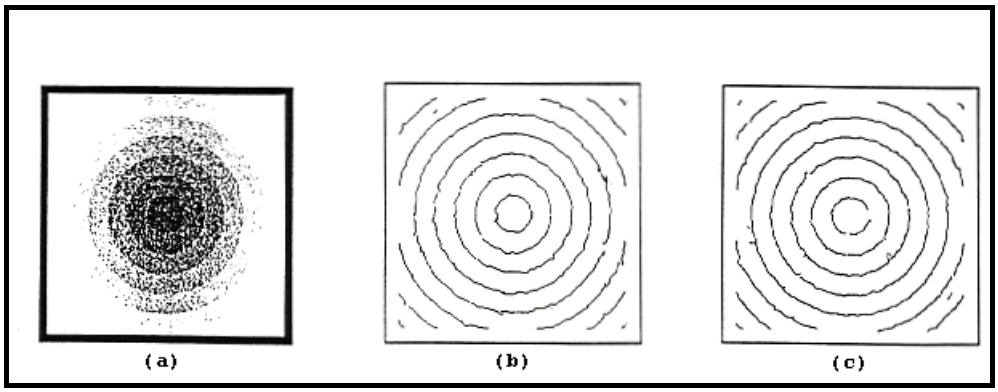


FIGURA 2.2 - a) Imagens de círculos com ruído Gaussiano e *SNR = 6*. b) bordas detectadas pelo detector neural e c) bordas detectadas pelo operador de Canny.
FONTE: Sirinivasan et al. (1994, p. 1660).

Se o deslocamento espacial do pixel *(x,y)* para a borda mais perto é $r_0\boldsymbol{i}+r_1\boldsymbol{j}$ e a orientação da borda mais perto é θ_{xy}, então o vetor de borda é $\boldsymbol{s}'_{xy}=$ $e^{-(r_0^2+r_1^2)/2\sigma^2}(sen\theta_{xy}\,\mathbf{i}+\cos\theta_{xy}\,\mathbf{j})$ que satisfaz as condições gerais de detecção ótima como

proposto por Canny (Canny, 1986). Na ativação, a rede detectora recebe entradas da rede codificadora.

A figura 2.2 ilustra resultados obtidos com a aplicação do detector. As imagens utilizadas são de *256 x 256* e o campo receptivo é de *15 x 15*. A camada codificadora tem *L=16* neurônios e a decodificadora tem *M=32* neurônios. Segundo o trabalho, *L* e *M* foram escolhidos empiricamente em função da carga computacional exigida, $\alpha=0.9999$ e $\beta=0.001$. Na ativação a rede foi testada na imagem com círculos concêntricos, deslocando-se sobre a imagem a um passo de 1 pixel por vez.

Nas considerações dos autores, a rede neural proposta é comparável com a abordagem proposta por Canny para detectar bordas abruptas em imagens ruidosas. A tabela 2.1 mostra resultados da comparação do detector de bordas neural com outros detectores.

TABELA 2.1 - Comparação de figuras de mérito do detector de bordas por rede neural e outros

SNR	Operador Sobel	Operador Prewitt	LoG	Operador de Canny	Rede Neural
Sem ruído	1	1	1	1	1
12	0.60	0.61	0.42	0.89	0.88
6	0.27	0.27	0.32	0.82	0.82
3	0.21	0.21	-	0.67	0.69

FONTE: Srinivasan et al. (1994, p. 1661).

A figura de mérito, parâmetro de comparação, é calculada por $F = \left[\frac{1}{\max(N_I, N_D)}\right]\sum_{i=1}^{N_D}\frac{1}{(1+\delta d_i^2)}$ onde d_i é a distância do pixel declarado como borda e o pixel de borda ideal mais perto, δ é uma constante de calibração, N_I e N_D são números de pixels de bordas ideal (*I*) e detectado (*D*).

Os valores na tabela 2.1 foram obtidos utilizando-se uma imagem sintética com círculos concêntricos, com os operadores de Sobel, Prewitt, Laplaciano da Gaussiana e Casamento de superfície de Haralick e Canny.

Os autores enfatizam algumas vantagens do detector por rede neural em relação ao detector de Canny: 1) pode ser treinado para qualquer tipo de distribuição de ruído e perfil de borda, enquanto que o detector de Canny é apropriado para bordas abruptas na presença de ruído Gaussiano; 2) pode ser implementado em máquinas paralelas, devido a natureza paralela do modelo, com tempo de execução pequeno.

Paik et al. (1992) descrevem um algoritmo usando ADALINES para detectar bordas, que mantém o desempenho computacional e reduz os efeitos de ruído. Os neurônios são lineares adaptativos, com estados +1, 0 e -1. Isso torna o algoritmo um supressor de ruídos, sem aumentar a dimensão do operador.

As entradas da ADALINE são pixels que têm seus valores mapeados para estados discretos, resultantes da classificação de cada ponto da janela em: 1) ponto de borda do lado alto, 2) ponto de borda do lado baixo, ou 3) não é um ponto de borda. Uma entrada *j*-ésima da ADALINE é

$$v_j = \begin{cases} -1, & z_j < m_i - \delta \\ 0, & m_j - \delta \le z_j \le m_i + \delta \\ 1, & z_j > m_i - \delta \end{cases}$$

onde δ controla a resolução das bordas e o nível de ruído. Na equação $m_i = \frac{1}{r} \sum_{j=1}^{r} z_j$ r é o número de pixels da janela e z_j é o valor do *j*-ésimo pixel da subimagem ordenada lexicograficamente, com centro em x_i, ou seja, $z_j = x_k$, $j=1,..,r$, sendo

$$k = i + \left\{ \left[\frac{(j-1)}{l} \right] - \frac{1}{2} \right\} n_2 + \left\{ (j-1) \bmod l - \left\lfloor \frac{1}{2} \right\rfloor \right\}$$

onde l é a dimensão da máscara usada.

O problema foi reformulado como um dispositivo que envolve 3 hipóteses: a) H_1 - *v_j pertence ao lado alto da borda; b)* H_o - *v_j não é ponto de borda; c)* H_{-1} - *v_j pertence ao lado baixo da borda.* Os autores supõem uma borda abrupta de amplitude $2E$, logo as hipóteses são:

a) H_1: $z_j = m_i + E + n_i$; b) H_0: $z_j = m_i + n_i$; c) H_{-1}: $z_j = m_i - E + n_i$

onde n_i é uma componente de um ruído Guassiano com média zero e variância σ^2. Devido à simetria do problema, Paik et al. (1992) consideram o critério de decisão da máxima verossimilhança, entre H_1 e H_0, como:

$$L(z_j) = \frac{\frac{1}{\sqrt{2\sigma^2}} e^{\frac{-1}{2\sigma^2}(z_j - (m_i + E))^2}}{\frac{1}{\sqrt{2\sigma^2}} e^{\frac{-1}{2\sigma^2}(z_j - m_i)^2}} \begin{cases} > & T \quad (H_1) \\ < & T \quad (H_0) \end{cases}$$

T é um limiar para probabilidade de ocorrência de uma borda. Simplificando e tomando o logaritmo da expressão:

$z_j \begin{cases} > (H_1) \\ < (H_0) \end{cases} m_i + \frac{\tau\sigma^2}{E} + \frac{E}{2}$ onde $\tau = ln(T)$. δ pode ser estimado a partir de $\delta = \frac{\tau\sigma^2}{E} + \frac{E}{2}$.

Para ruído com alta intensidade, δ é alto, o que suprime erros de classificação devido ao ruído. Com ruídos pequenos, δ suprime erros devido a erros de textura na imagem, erros de quantização e bordas contínuas (em rampa).

No experimento usou-se 12 padrões de bordas bidirecionais (Tabela 2.2) com *l*=3, somado *9* pixels e assumindo os estados +1, -1 e *X* (que significa qualquer). Aumentando-se *l*, é possível aumentar o número de padrões.

No trabalho Paik et al. (1992) usaram a separabilidade linear para classificar os dados da entrada do detector. A definição de separabilidade linear considera um conjunto P de vetores de dimensão L, cujos componentes assumem um dos *M=j+k+1* valores do conjunto *{-j,…,-1,0,1,…,k }*, ou seja, P=*{x|x*$_i$ $\in$ *{-j,…,-1,0,1,…,k}, i=1,…,L }*. Os valores extremos no intervalo são *-j* e *k*.

TABELA 2.2 - Padrões de bordas em direções diferentes.

1	*X*	*-1*	*-1*	*X*	*1*	*-1*	*-1*	*-1*	*1*	*1*	*1*
1	*X*	*-1*	*-1*	*X*	*1*	*X*	*X*	*X*	*X*	*X*	*X*
1	*X*	*-1*	*-1*	*X*	*1*	*1*	*1*	*1*	*-1*	*-1*	*-1*
90º			270º			0º			180º		
X	*-1*	*-1*	*1*	*X*	*1*	*-1*	*-1*	*X*	*1*	*X*	*1*
1	*X*	*-1*	*-1*	*X*	*1*	*-1*	*-1*	*X*	*-1*	*-1*	*X*
1	*1*	*X*	*-1*	*-1*	*X*						
135º			315º								
-1	*-1*	*X*	*1*	*1*	*X*	*-1*	*-1*	*X*	*1*	*1*	*X*
-1	*X*	*1*	*1*	*X*	*-1*	*-1*	*X*	*1*	*1*	*X*	*-1*
X	*1*	*1*	*X*	*-1*	*-1*	*-1*	*X*	*1*	*1*	*1*	*X*
45º			225º			45º/ 90º			225º/ 270º		
-1	*-1*	*X*	*1*	*1*	*X*	*-1*	*-1*	*X*	*1*	*1*	*X*
-1	*X*	*1*	*1*	*X*	*-1*	*-1*	*X*	*1*	*1*	*X*	*-1*
-1	*-1*	*X*	*X*	*-1*	*-1*	*-1*	*X*	*1*	*1*	*1*	*X*
45º / 315º			225º/ 135º								

FONTE: Paik et al. (1992, p. 1498).

Tomando dois subconjuntos de *P* mutuamente exclusivos, P_0 e P_1, $P_0 \cup P_1 = P$, com um vetor peso w pode-se definir uma função lógica $F(x) = \begin{cases} 1, & w^T x \geq \theta \\ 0, & w^T x \leq \theta \end{cases}$ onde θ é um valor limiar. Se *F* é separável linearmente então $w^T x > w^T y$, $\forall x \in P_1$ e $\forall y \in P_o$.

No caso de multi-estados, Paik et al. (1992) apresentaram um teorema propondo uma função linearmente separável, que separa o vetor de entrada *x,* de dimensão *L*, dos vetores em P_o. Cada componente de *x* assume um valor extremo, formando os conjuntos P_1=*{x}* e o conjunto P_o=*P-*P_1. Para a prova toma-se a definição de separabilidade linear e um vetor *w*,

com componentes $w_i=\begin{cases}1, & x_i=k\\ -1, & x_i=-j\end{cases}, i=1,...,L$ tal que $w_i x_i \geq w_i y_i$, $\forall I$. A desigualdade implica $w^T x > w^T y$. Tomando-se um conjunto $P_{-1}=\{-x\}$, x com valores extremos, vale $w^T(-x) < w^T y < w^T x$, $\forall y \in P_o$. O teorema permite duas formas de determinar os w_i, $i=1,\ldots,L$:

a) os estados X assumem valores extremos, logo na borda os valores X e seus complementares seriam os mesmos com sinais trocados. Com a função discriminante, um mesmo vetor pode detectar bordas em cada par bidirecional de bordas, pois têm valores extremos com sinais trocados. No cálculo dos w_i usa-se a definição anterior. Como exemplo, para detectar bordas direita ou esquerda em uma janela de tamanho $l = 3$:$w_- = [\ 1\ w_2\ -1\ \ 1\ w_5\ -1\ \ 1\ w_8\ -1\ ]^T$ onde w_2, w_5 e w_8 podem ser $+1$ ou -1, dependendo de X no padrão de bordas correspondente. b) os valores X assumem apenas valores extremos sem alteração de sinal. Os componentes correspondentes aos estados X são zero. No exemplo anterior $w_2=w_5=w_8=0$.

O limiar θ na determinação de F, depende das entradas do vetor de peso. Nos casos foi usado θ igual ao número de elementos da janela de análise ($r=lxl$), no primeiro e θ igual ao número de elementos da janela menos os estados X, no segundo. A análise desenvolvida permite que os vetores pesos sejam determinados de forma direta, pois se usa valores extremos, do contrário deve-se usar o algoritmo do *LMS*.O algoritmo para detecção de bordas proposto baseia-se na definição de bordas e na determinação de pesos discutidas. Na figura 2.3 usou-se os 4 primeiros padrões de bordas. A saída $g(.)$ é $g(u)=\begin{cases}1, & u \geq \theta\\ 0, & -\theta < u < \theta\\ -1, & u \leq \theta\end{cases}$

onde θ é um limiar de entrada, definido como δ. A borda resulta da operação *OR* sobre as saídas das quatro ADALINES alteradas por $g(.)$.

O detector foi experimentado e comparado com os operadores LoG e de Canny. O estado X nos padrões foi zero e θ=6, tais parâmetros permitiram localizar bordas em qualquer direção. Usou-se a imagem da figura 2.4 para os testes. A imagem tem níveis de ruído diferentes, ou seja, *SNR=100, 40, 20, 10, 1 dB* (1ª coluna da figura).

Para o operador neural,δ variou na seqüência: *45, 45, 45.3, 48.3* e *71.5*, com E=90 e τ = 0.68. Para o operador LoG σ_L = *0.5, 0.5, 4, 5* e *6*, na seqüência (3ª coluna da figura 5.4).

A supressão de ruído necessita máscaras de tamanhos variados para se ter resultados semelhantes ao operador neural. A comparação em relação a Canny considerou a localização da borda. Tomou-se uma máscara de *5 x 5*. Os resultados estão na quarta coluna da figura 2.4.

Paik et al. (1992) encontraram degradação na qualidade das bordas detectadas pelo operador Canny.

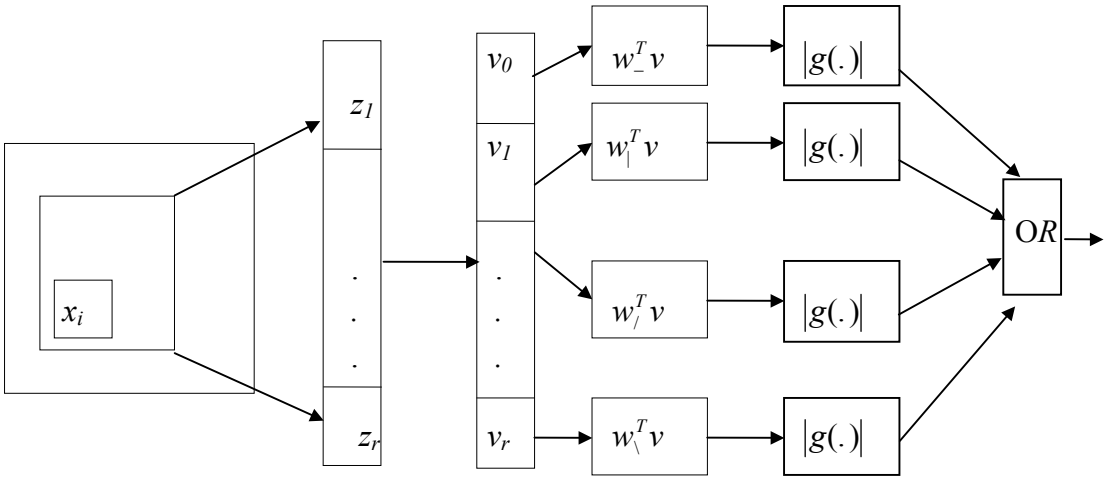


FIGURA 2.3 - Procedimento de detecção de bordas usando ADALINE de três estados.
FONTE: Paik et al. (1992, p. 1500).

O experimento foi repetido para uma imagem real e os resultados foram semelhantes (Paik et al., 1992). Para taxas de ruído altas os três algoritmos têm desempenhos semelhantes, mas os autores observaram que o detector neural era mais sensível às bordas mais finas.

Na conclusão Paik et al. (1992) comentam que detector neural proposto é adequado para detectar bordas e supera os operadores LoG e de Canny, nas mesmas condições de testes. Três vantagens são realçadas do experimento com o operador neural: 1) a possibilidade de redução de ruído, mantendo-se o tamanho da janela de análise e o desempenho na localização da borda; 2) baixa carga computacional; e 3) a largura da borda pode ser controlada modificando-se a definição de borda, isto é, escolhendo valores específicos para os estados *X*.

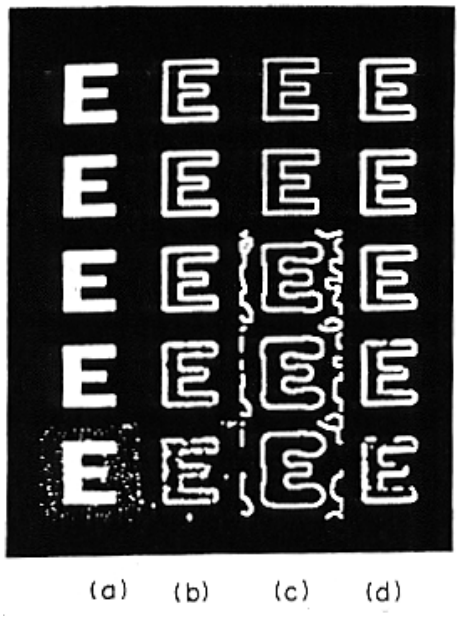


FIGURA 2.4 - Imagens com níveis de ruído diferentes para teste do detector neural e resultados obtidos com os operadores neural, LoG e Canny.
FONTE: Paik et al. (1992, p. 1501).

Bhuiyan et al. (1994) argumentam que a detecção de bordas tem problemas com a não uniformidade de iluminação. Assim, os métodos existentes podem falhar na detecção para objetos similares em regiões com sombreamento diferentes na cena, necessitando de readaptação às novas condições de iluminação. Eles propõem um detector baseado na rede de Hopfield usando resultados de Koch (1986), que mostrou que redes de Hopfield podem ser generalizadas para resolver as funções de energias não quadráticas de visão primária, pelo mapeamento dos processos de linhas binárias em variáveis contínuas limitadas por 0 e 1, vislumbrando também uma função de custo associada. A rede Hopfield trata apenas de problemas quadráticos. O trabalho usa a minimização de energia, e propõe um esquema para mudar os parâmetros da função de energia, em uma imagem com iluminação variável. A implementação proposta foi comparada qualitativamente e quantitativamente com os operadores de Sobel 3 x 3 e LoG, sobre uma imagem sintética. A figura 2.5 mostra a arquitetura proposta.

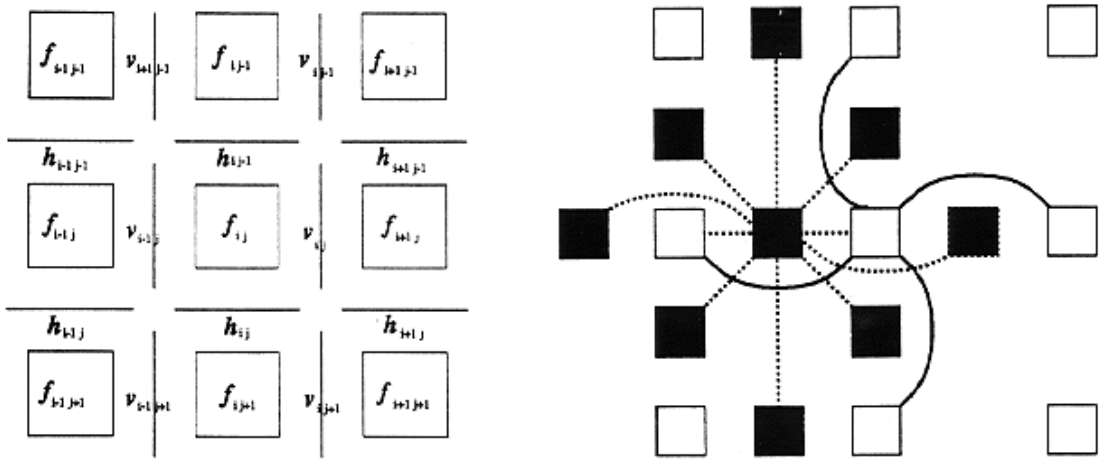


FIGURA 2.5 - Implementação do detector de borda baseado na rede Hopfield.
FONTE: Bhuiyan et al. (1994, p. 622).

No modelo uma imagem $N \times N$ tem $3 \times N \times N$ neurônios, assim um reticulado como na figura 2.5, cada pixel (i,j) tem três neurônios associados, um para o nível de intensidade com variável interna de estado é igual a saída $f_{i,j}$, outro representando o processo de linha horizontal, que decide se há descontinuidade na intensidade entre o pixel $f_{i,j}$ e $f_{i,j+1}$ e que cuja saída é dada pela função sigmóide da variável de estado interna $m_{i,j}$: $h_{i,j} = \frac{1}{1+e^{-2\lambda m_{i,j}}}$, se não há descontinuidade $h_{i,j}=0$, do contrário $h_{i,j}=1$. O último neurônio, responde pelo processo vertical da linha, cuja saída $v_{i,j}$ é semelhante à horizontal, mas depende da variável de estado interna $n_{i,j}$.

A energia considerada é dada por: $E=E_I+E_D+E_V+E_P+E_C+E_L+E_G$, cujas componentes são dadas por:

$$E_I = C_r \sum_{i,j} [(f_{i,j+1}-f_{i,j})^2(1-h_{i,j})+(f_{i+1,j}-f_{i,j})^2(1-v_{i,j})]$$

$$E_D = C_D \sum_{i,j} (f_{i,j}-d_{i,j})^2$$

$$E_v = C_v \sum_{i,j} [h_{i,j}(1-h_{i,j})+v_{i,j}(1-v_{i,j})]$$

$$E_P = C_P \sum_{i,j} [h_{i,j}h_{i,j+1}+v_{i,j}v_{i+1,j}]$$

$$E_C = C_c \sum_{i,j} [h_{i,j}+v_{i,j}]$$

$$E_L = C_L \sum_{i,j} \{h_{i,j}[1-h_{i+1,j}-v_{i,j}-v_{i,j+1})^2+(1-h_{i-1,j}-v_{i-1,j}-v_{i-1,j+1})^2]+$$

$$v_{i,j}[(1-v_{i,j+1}-h_{i,j}-h_{i+1,j})^2+(1-v_{i,j-1}-h_{i,j-1}-h_{i+1,j-1})^2]\}$$

$$E_G = C_G \sum_{i,j} [\int_0^{h_{i,j}} g_{i,j}^{-1}(h_{i,j})dh_{i,j}+\int_0^{v_{i,j}} g_{i,j}^{-1}(v_{i,j})dv_{i,j}]$$

Bhuiyan et al, (1994) observaram que as componentes introduzem restrições. A componente E_I depende das intensidades dos pixels vizinhos. A componente E_D representa a credibilidade do dado, nela $d_{i,j}$ é a intensidade do pixel, portanto, $(f_{i,j}-d_{i,j})^2$ é a diferença entre o valor original e o valor aproximado $f_{i,j}$. O peso C_D depende da relação sinal/ruído. E_V introduz uma restrição de fronteira e força os processos de linha para 0 ou 1. E_P introduz a restrição de linha única, penalizando a formação de linhas paralelas adjacentes. O custo pago pela introdução de cada linha é E_C. Processos de linha ativos ao longo de uma linha ou curva são representados por E_L, que é um termo que favorece linhas contínuas e penaliza interações de múltiplas linhas e segmentos descontínuos. A componente E_G força os processos de linha para dentro do hipercubo $[0,1]^N$.

As variações em $f_{i,j}$, $m_{i,j}$ e $n_{i,j}$ no tempo são dadas por: $\frac{df_{i,j}}{dt} = -\frac{\partial E}{\partial f_{i,j}}; \frac{dm_{i,j}}{dt} = -\frac{\partial E}{\partial h_{i,j}}; \frac{dn_{i,j}}{dt} = -\frac{\partial E}{\partial v_{i,j}}$. Portanto, a taxa de variação no tempo da energia total $E(f_{i,j},h_{i,j},n_{i,j})$ é:

$$\frac{dE}{dt} = \sum_{i,j} \frac{\partial E}{\partial f_{i,j}} \frac{df_{i,j}}{dt} + \sum_{i,j} \frac{\partial E}{\partial m_{i,j}} \frac{dm_{i,j}}{dt} + \sum_{i,j} \frac{\partial E}{\partial n_{i,j}} \frac{dn_{i,j}}{dt}$$

$$= \sum_{i,j} \frac{\partial E}{\partial f_{i,j}}\left(-\frac{\partial E}{\partial f_{i,j}}\right) + \sum_{i,j} \frac{\partial E}{\partial m_{i,j}}\left(-\frac{\partial E}{\partial h_{i,j}}\right) + \sum_{i,j} \frac{\partial E}{\partial n_{i,j}}\left(-\frac{\partial E}{\partial v_{i,j}}\right)$$

$$= -\sum_{i,j} \left\{\left(\frac{\partial E}{\partial f_{i,j}}\right)^2 + \left(\frac{\partial E}{\partial h_{i,j}}\right)^2 \frac{\partial h_{i,j}}{\partial m_{i,j}} + \left(\frac{\partial E}{\partial v_{i,j}}\right)^2 \frac{\partial v_{i,j}}{\partial n_{i,j}}\right\}$$

Considerando a definição de $h_{i,j}$, $\frac{\partial h_{i,j}}{\partial m_{i,j}} = 2\lambda h_{i,j}(1-h_{i,j}) \geq 0$. De forma semelhante $\frac{\partial v_{i,j}}{\partial n_{i,j}} \geq 0$. Como nenhum termo é negativo, $\frac{dE}{dt} \leq 0$, ou seja, a energia será sempre decrescente e o sistema tenta encontrar um mínimo para E.

A figura 2.6 ilustra a variação de E e componentes em função da diferença de intensidades $\left| f_{i,j+1} - f_{i,j} \right|$.

Na figura 2.6 a) a energia total não é proporcional a soma das componentes, com o aumento da diferença das intensidades. Assim, a relação entre cada componente e a energia total aumenta com a diferença. Os processos de linhas são sensíveis a contraste. Para mostrar as evidências foi usada uma imagem com duas regiões de borda, uma com intensidade 10 vezes maior que a outra. No experimento encontrava-se falsas bordas com os parâmetros ajustados para detectar na região de baixo contraste e mascarava-se bordas quando se ajustava os parâmetros para detecção na região de alto contraste. A figura 2.6 a) tem-se uma região de ótimo, onde não há problemas na detecção de bordas.

No trabalho, Bhuiyan et al. (1994) propõem um diagrama de energia (figura 2.6b) que mantém relativamente constante a relação entre cada componente e a energia total E, acomodando toda a região para o propósito de detecção de bordas. Para isso optou-se por variações de segunda ordem, $\left(f_{i,j+1} - f_{i,j}\right)^2$, para as componentes em função da diferença de intensidades. As componentes são variadas pelos parâmetros C_* (C_V, C_P, C_C, C_L e C_G) em segunda ordem, como função das diferenças de intensidades. O parâmetro C_D é constante para um determinada relação sinal/ruído. A figura 2.7 exibe a proposta de variação de segunda ordem dos parâmetros C_*.

Bhuiyan et al. (1994) usaram um limite inferior para os valores C_*, para evitar a detecção de bordas ruidosas se $\left| f_{i,j+1} - f_{i,j} \right| < f_{th}$, valor limiar ótimo. Os valores iniciais dos parâmetros foram determinados heuristicamente como $C_I=1$, $C_D=2$, $C_V=0.2$, $C_P=2$, $C_C=0.4$, $C_L=1.6$, $C_G=0.2$, $\lambda = 25$ e $f_{th} = 0$.

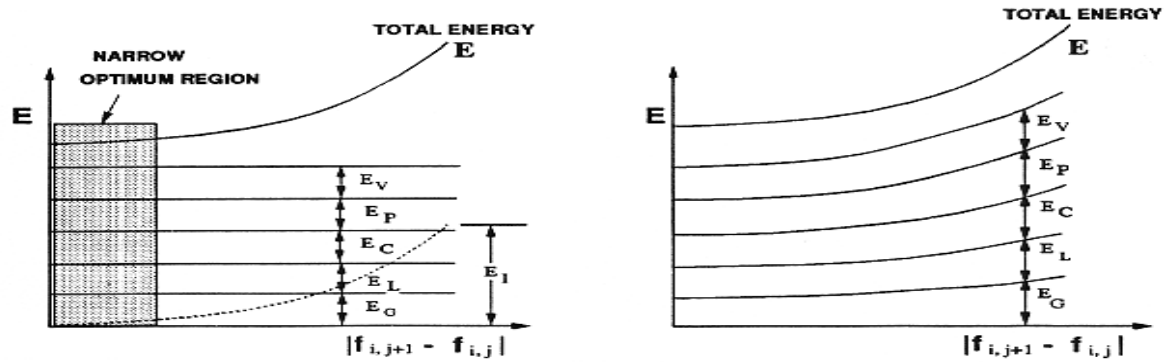
O controle sobre a atualização é mantido, expressando-se as varáveis em função de C, ($C_V=C$, $C_P=10C$, $C_C=2C$, $C_L=8C$, $C_G=C$). C_I e C_D mantêm-se nos valores iniciais. Na atualização o algoritmo considera 7 valores de intensidades com centro no pixel (i,j) e toma a média das intensidades na horizontal e na vertical:

$$C(h_{i,j}) = [C(h_{i,j}) + C(h_{i-1,j}) + C(h_{i+1,j}) + C(v_{i,j}) + C(v_{i,j+1}) + C(v_{i-1,j}) + C(v_{i-1,j+1})] / 7$$

$$C(v_{i,j}) = [C(v_{i,j}) + C(v_{i-1,j}) + C(v_{i+1,j}) + C(h_{i,j}) + C(h_{i,j+1}) + C(h_{i-1,j}) + C(h_{i-1,j+1})] / 7$$

Os valores são atualizados a cada 5 iterações. Abaixo de C_{min} a formação de linhas é penalizada. A tabela 2.3 mostra a atualização para C_{min} com o número de iterações. Os valores de C_{min} foram escolhidos observando-se imagens de duas barras idênticas, dois círculos

idênticos e dois tabuleiros de xadrez idênticos, sob faixas diferentes de contraste (de médio para alto contraste, ou seja, nas razões de 6:1, 10:1, 15:1 e 20:1, respectivamente).



a) b)

FIGURA 2.6 - a) Relação da energia *E* e suas componentes. b) Diagrama de energia proposto por Bhuiyan et al. (1994).
FONTE: Bhuiyan et al. (1994, p. 623).

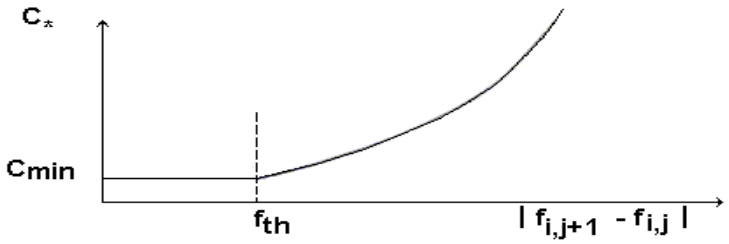


FIGURA 2.7 - Relação de segunda ordem dos parâmetros $C*$ com a diferença de intensidades.
FONTE: Bhuiyan et al. (1994, p. 623).

Inicialmente no processo de detecção, $h_{i,j}=v_{i,j}=0.5$ e $f_{i,j}\approx d_{i,j}$. A rede computa a superfície mais suave tomando os processos de linha como zero. O processo de intensidade é atualizado *100* vezes para cada atualização da rede de processo de linha, ou seja, a rede do processo de linha é estacionária. Para isso Bhuiyan et al. (1994) usaram unidades de tempos distintas: $\Delta t1 = 0.0001$ para o processo de intensidade e $\Delta t2 = 0.01$ para o processo de linha. Calcula-se $C(h_{i,j})$ e $C(v_{i,j})$. As bordas existem se $h_{i,j}$, $v_{i,j} \rightarrow 1$.

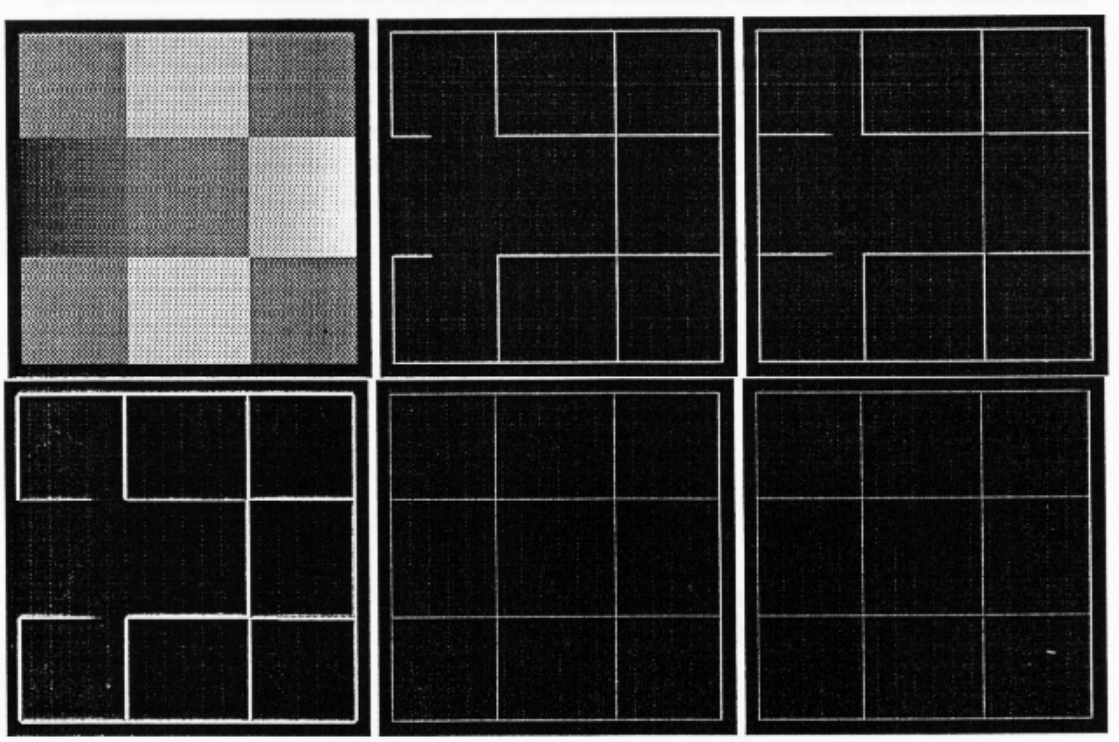


FIGURA 2.8 - Imagem usada para detecção de bordas em Bhuiyan et al. (1994).
FONTE: Bhuiyan et al. (1994, p. 625).

TABELA 2.3 - Variação de C_{min} em função do número de iterações.

Número de iterações	C_{min}
1 - 4	0.20
5 - 19	0.10
20 - 49	0.05
50 - 99	0.02
100 - 199	0.01

FONTE: Bhuiyan et al. (1994, p. 624).

Na tabela 2.3 C_{min} cai com o aumento do número de iterações para a minimização da energia. O valor de *0.2* corresponde a detecção de bordas com grandes diferenças entre pixels. O decréscimo gradativo significa que bordas cada vez menos abruptas estão sendo detectadas. O processo continua até se atingir um estágio de energia mínima. No experimento observou-se um limite máximo de *200* interações.

O processo detecta muitas bordas pequenas, mas a restrição E_L permite a conexão das várias bordas pequenas, formando bordas alongadas e contínuas.

O algoritmo foi testado, juntamente com os operadores LoG, Sobel 3 x 3, Sobel 3 x 3 baseado em contraste, em uma imagem com tamanho de *470 x 470*, cujas intensidades alternavam-se em áreas claras com nível de cinza *200* e áreas ligeiramente mais escuras com nível de cinza *100* e uma área variando em rampa de *10* a *244* conforme figura 2.8.

Ao longo dos limites do quadrado $h_{i,j}=v_{i,j}=1$ como condições de contornos. Como condições iniciais $m_{i,j}=n_{i,j}=0$, para $h_{i,j}=v_{i,j}=0.5$. A figura 2.8 exibe os resultados da detecção pelos operadores e o padrão de borda ideal. Inspeção visual à figura, mostra que os operadores LoG e de Sobel não detectaram bordas na variação em rampa. Por sua vez o detector neural deu um resultado muito próximo do padrão esperado. Outro aspecto é a largura da borda que variou com o detector. O detector neural resultou na borda mais fina.

Uma análise quantitativa está resumida na tabela 2.4. Nela *P(AE/IE)* é a probabilidade condicional de um pixel de borda atribuído, dado um pixel de bordas ideal e *P(IE/AE)* é a probabilidade condicional de um pixel de borda ideal, dado um pixel atribuído. O limiar de binarização foi escolhido de forma que $P(AE/IE) \geq P(IE/AE)$ em cada caso. *P(AE/IE)* e *P(IE/AE)* retratam medidas de rejeição e detecção falsa. O erro é uma medida de desvio das bordas verdadeiras. Bhuiyan et al. (1994) concluem então que os detectores devem ter capacidade de maximizar as probabilidades condicionais e minimizar o erro.

TABELA 2.4 - Avaliação quantitativa dos algoritmos.

Algoritmo	*P(AE/E)*	*P(IE/AE)*	*Erro de distância*
Neural	1.000	1.000	0.000
Sobel	0.753	0.501	0.750
Sobel contraste	0.510	0.500	0.508
LoG	0.481	0.438	0.706

FONTE: Bhuiyan et al. (1994, p. 624).

De acordo com os resultados das probabilidades e erro, na tabela 2.4, o algoritmo neural apresentou o melhor desempenho, comparado a outros detectores nas mesma situação de teste.

Outros trabalhos utilizando redes neurais em problemas de visão computacional e de processamento de imagens, comentados com menos detalhes de implementação, são relacionados a seguir.

Cruz et al. 1995, realizaram um experimento para resolver o problema de casamento de características em visão estéreo. O casamento consiste em encontrar características comuns em pares de imagens que possibilitem o estabelecimento de referências entre as imagens. Neste trabalho os autores empregam redes neurais auto-organizáveis. O objetivo é a

classificação dos pares de características, no caso segmentos de bordas, como casamentos verdadeiros ou falsos, o que implica na existência de duas classes. O vetor parâmetro de correspondência de duas funções densidade componentes, que representam as classes e parecem com densidades normais, são estimadas usando um método de aprendizagem não supervisionado. A rede utilizada tem uma topologia de três camadas e implementa a função de densidade de mistura e a regra de Bayes. O método de aprendizagem não supervisionado é uma regra de aprendizagem e as restrições de visão estéreo levam a uma regra de ativação.

Haring et al. 1994 utilizaram a rede de Kohonen para segmentação de imagens em multiescala. Nesta abordagem a imagem é descrita atribuindo-se um padrão de característica para cada elemento da imagem. Este padrão consiste de uma família de características geométricas diferenciais invariantes. Cada padrão da imagem de treinamento é uma entrada para uma rede de Kohonen, para se obter uma descrição do espaço de características em termos de padrões de características protótipos, representados pelos vetores pesos da rede neural. Após esta fase é possível atribuir-se rótulos, através de um algoritmo supervisionado, para cada um dos protótipos, utilizando-se classes obtidas de uma segmentação a priori da imagem de treinamento. A segmentação de qualquer imagem semelhante a de treinamento, é feita comparando-se a representação do padrão de características de cada elemento de imagem, com todos os vetores pesos e atribuindo-se cada elemento a classe que apresente o melhor vetor de casamento.

Accoto et al. 1993, utilizaram redes neurais na análise de textura em imagens. A medida de textura é considerada baseia-se no conceito de dimensão fractal e é avaliada por uma rede neural. O treinamento da rede neural utilizou um conjunto de padrões sintéticos com dimensões fractais conhecidas. A metodologia desenvolvida foi utilizada em um problema real de análise de imagens do mar, para identificação de vazamentos de óleo.

Wenzel e Gimblett 1993, experimentaram redes neurais em dois problemas clássicos de processamento de imagens: compressão e reconstrução de imagens. No trabalho utilizou-se um perceptron em multicamadas treinado pelo algoritmo "backpropagation". Observou-se que o mesmo algoritmo realizava compressão dos dados e conservava o significado associado aos "bytes" correspondentes aos padrões, o que pode ser utilizado para reconstrução.

Etapa 3. Estudo teórico de alguns algoritmos de redes neurais.

A seguir é feita uma introdução às redes neurais ressaltando alguns algoritmos mais utilizados em aplicações práticas. Existem várias publicações com material didático criterioso, que introduzem as Redes Neurais como ferramentas computacionais. Nas referências bibliográficas são citados alguns livros e artigos que trazem textos básicos sobre o assunto.

Redes Neurais Artificiais

As Redes Neurais Artificiais (RNA) são modelos matemáticos que representam os princípios de atividades do cérebro com base na neurobiologia e na teoria do comportamento. Estes modelos são adequados para resolver problemas que envolvem classificação ou predição, usando informações conflitantes ou incompletas, tais como em aplicações de reconhecimento de padrões, otimização, controle, compressão de dados, diagnóstico e aproximação de funções, entre outras.

As redes neurais artificiais diferem de outros modelos computacionais, por possuir uma arquitetura composta de várias unidades de processamento simples interconectadas (neurônios), formando uma estrutura semelhante a uma rede de tecido nervoso cerebral. Apesar da semelhança na descrição não são modelos completos do cérebro humano. Para aproximações mais realistas é necessário pesquisas em ciências cognitivas e neurobiológicas e na tecnologia de processamento analógico-digital em larga escala.

O princípio de funcionamento das redes neurais está diretamente ligado ao funcionamento de cada unidade de processamento individual. Cada unidade quando excitada processa as informações apresentadas nas sua entradas (dendritos) até sua saída (axônio), segundo uma função de transferência, inspirando-se no comportamento de neurônios biológicos. Como no cérebro, as entradas da rede neural correspondem aos sensores e, a saída aos neurônios motores (atuadores) que respondem aos estímulos.

O funcionamento básico de cada nó (elemento de processamento ou neurônio), consiste no somatório da multiplicação de cada elemento de entrada pelo peso (reforçador ou inibidor) associado a ele. O resultado passa por uma função de transferência produzindo a saída do nó, que é submetida aos dendritos dos nós da camada seguinte.

A figura 3.1 ilustra esquematicamente um modelo de neurônio utilizado em redes neurais artificiais.

O interesse em redes neurais está na possibilidade de construção de redes de computação artificiais (Hertz et al., 1991), que têm a capacidade de imitar o cérebro nas tomadas de decisão e no raciocínio envolvendo informações que apresentam características complexas, ruidosas, irrelevantes e/ou parciais (Eberhart e Dobbins, 1990), e cujo padrão de conexão codifica as estruturas complexas envolvidas na solução de um problema, o que implica no termo "conexionismo" (Khanna, 1990). Estas características das redes neurais podem ser usadas na solução de problemas ditos difíceis de resolver por outras abordagens conhecidas (Eberhart e Dobbins, 1990).

As primeiras implementações de redes neurais foram feitas por pesquisadores da biologia, fisiologia e psicologia, que tentavam explicar resultados experimentais e observações do comportamento e da construção do cérebro (Eberhart e Dobbins, 1990).

A forma das redes neurais armazenar conhecimento é através de aprendizagem que é a medida da variação da memória no tempo, $dW/dt \neq 0$. A memória resulta do processo de atualização dos pesos das conexões, provocado pela aquisição de novos conhecimentos.

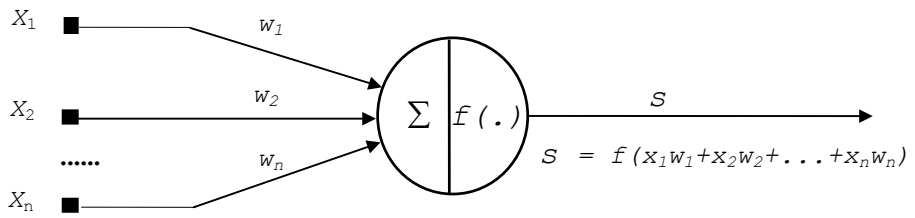


FIGURA 3.1 - Modelo de neurônio usado em redes neurais artificiais.

A aprendizagem pode ser não-supervisionada ou supervisionada. Na primeira o processo iterativo continua até se atingir uma resposta estável para um determinado conjunto de padrões de entrada. Na aprendizagem supervisionada é feito o monitoramento da amostragem de pares do conjunto de padrões de entrada e saída definidos, observando-se o princípio da aprendizagem por condicionamento.

A figura 3.1 ilustra um elemento de processamento básico utilizado em redes neurais. O efeito da soma na unidade resulta na determinação de um hiperplano discriminante, que separa os pontos nos dois lados do plano de forma diferente. Os sinais dos pesos (conexões) tornam a entrada excitatória ou inibitória. Portanto, as conexões entre as unidades de processamento caracterizam-se: a) pela natureza (excitatória ou inibitória); b) pelo grau de influência (peso), e mudam em resposta a novas entradas.

Uma regra utilizada para a aprendizagem é a regra de Hebb (Khanna, 1990) definida como: $w_i(t+1)=w_i(t)+Cx_i(t)y(t)$, onde $x_i(t)$ é a entrada i do nó, com saída $y(t)$; C - taxa de aprendizagem (ou constante de aprendizagem - autores diferentes usam nomenclaturas diferentes para representar esta constante); e $w_i(t)$ - matriz de pesos, que corresponde à memória. Com esta regra uma rede funciona como um associador de padrões. Na ativação, apresentando-se parcialmente um padrão, espera-se que a rede gere a versão completa do padrão associado. Para generalizar esta regra faz-se os pesos mudarem em função de um sinal de reforço R, $w_i(t+1)=w_i(t)+CR_i(t)$, onde $R_i(t)=x_i(t)y(t)$ é o reforço para a sinapse i no instante t, e C é a taxa de aprendizagem.

Em geral aplicações que envolvem correlação simultânea ou espacial, podem ser apresentadas em uma forma que a regra de Hebb pode implementar. A regra de Hebb não produz os aspectos temporais do condicionamento estímulo-resposta clássico, mesmo com atrasos e modificações. Portanto, não há memória de longo termo.

Outra regra de aprendizagem é a de Widrow-Hoff onde o sinal de reforço é definido por: $R_i(t) = [z(t) - y(t)]x_i(t)$, e portanto, $w_i(t+1) = w_i(t)+C[z(t) - y(t)]x_i(t)$, onde $y(t) = \sum_{j=1}^{n} w_i(t)x_i(t)$ e $z(t)$ é o sinal de saída desejado. Aqui os pesos convergem para que a resposta seja um número real desejado para cada estímulo. A regra implementa um algoritmo iterativo que computa uma solução para um conjunto de equações lineares, cuja solução existe se os padrões de estímulo $x_1,...,x_k$ são linearmente independentes.

Na regra de Hebb, uma resposta perfeita para uma entrada é possível apenas para padrões de estímulos ortogonais. Pela regra de Widrow-Hoff, a recuperação ocorre para conjuntos de padrões de estímulo linearmente independentes.

A aprendizagem competitiva é um esquema não associativo que ocorre quando as unidades de processamento competem e apenas a unidade de resposta mais forte a um padrão é modificada. A ativação de uma unidade em particular induz ou inibe a ativação de outras unidades ligadas a ela, de acordo com o tipo de conexão.

A atividade das unidades da rede é dada por uma função de ativação monotônica não decrescente, como por exemplo funções linear, rampa, degrau e sigmóide como na figura 3.2.

O arranjo das unidades de processamento forma a rede neural, dando origem aos modelos de redes neurais artificiais. Os modelos disponíveis na literatura fazem uso de tipos de conexões diferentes. As conexões podem ser inibitórias ou excitatórias em “feed forward”,

com a informação fluindo em um sentido único, ou em "feedback" com a informação fluindo em ambos os sentidos e/ou recursivamente.

Duas fases caracterizam bem o funcionamento de redes neurais: a aprendizagem, desenvolvida através de uma das técnicas discutidas antes, e a ativação, quando a rede, uma vez treinada, é colocada em funcionamento para recuperação de informações, dado uma certa entrada.

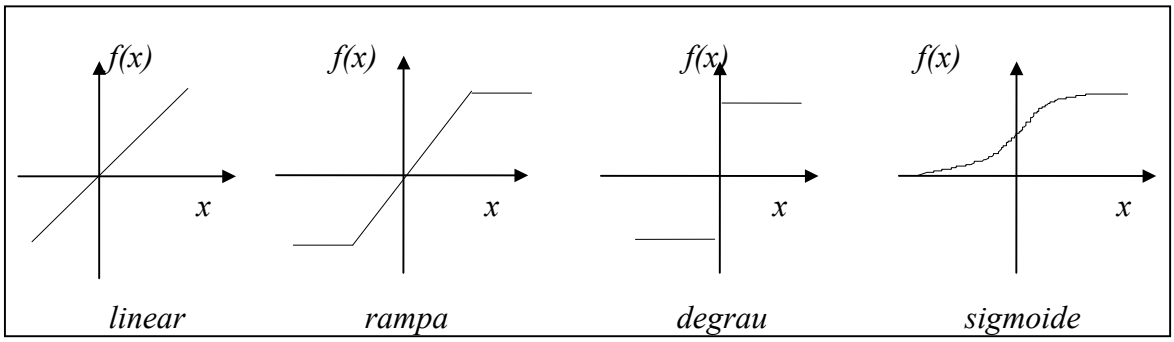


FIGURA 3.2 - Tipos de funções limiar utilizados na ativação de células (neurônios).

Modelos de Redes

Perceptron

O perceptron foi desenvolvido por Rosenblatt em 1957 que provou a convergência para padrões linearmente separáveis. Ele possui duas camadas, é heteroassociativo (associa padrões decorrelacionados) e faz o reconhecimento de padrões pelo método do vizinho mais próximo. O perceptron permite o armazenamento de pares de padrões (A_k,B_k), onde $k=1,...,m$.

A aprendizagem do é "offline" e ele opera em tempo discreto. A figura 3.3 ilustra um perceptron de duas camadas. Os n componentes de F_A são os elementos de A_k, e os m componentes de F_B são os elementos de B_k.

Na fase de aprendizagem do perceptron encontra-se a melhor configuração de pesos que separa linearmente os padrões. Os pesos das conexões entre os elementos de processamento (Simpson, 1990), juntamente com os valores de limiares (ou polarizações), são inicializados aleatoriamente para valores dentro do intervalo [+1,-1].

A cada padrão A_k a saída de cada elemento de processamento é calculada por: $b_j = f\left(\sum_{i=1}^{n} w_{ij} a_i - \theta_j\right)$, $j=1,...,m$, onde $f(x) = \begin{cases} 1, & x > 0 \\ -1, & c.c. \end{cases}$.

Tendo-se os b_j, calcula-se o erro em relação ao valor desejado, ou seja, $e_j=d_j^k - b_j$, para $j=1,...,m$. Os erros são usados para atualizar os pesos, para cada elemento, por: $\Delta w_{ij}=\alpha\, a_i\, e_j$, para $i=1,...,n$ e $j=1,...,m$. O processo continua até se atingir um erro suficientemente baixo ou zero.

Na ativação apresenta-se um padrão e observa-se a resposta. Para uma boa aprendizagem a saída será o padrão que foi associado à entrada durante o treinamento, ou uma aproximação de algum padrão aprendido. A ativação é dada por $b_j = f\left(\sum_{i=1}^{n} w_{ij} a_i - \theta_j\right)$.

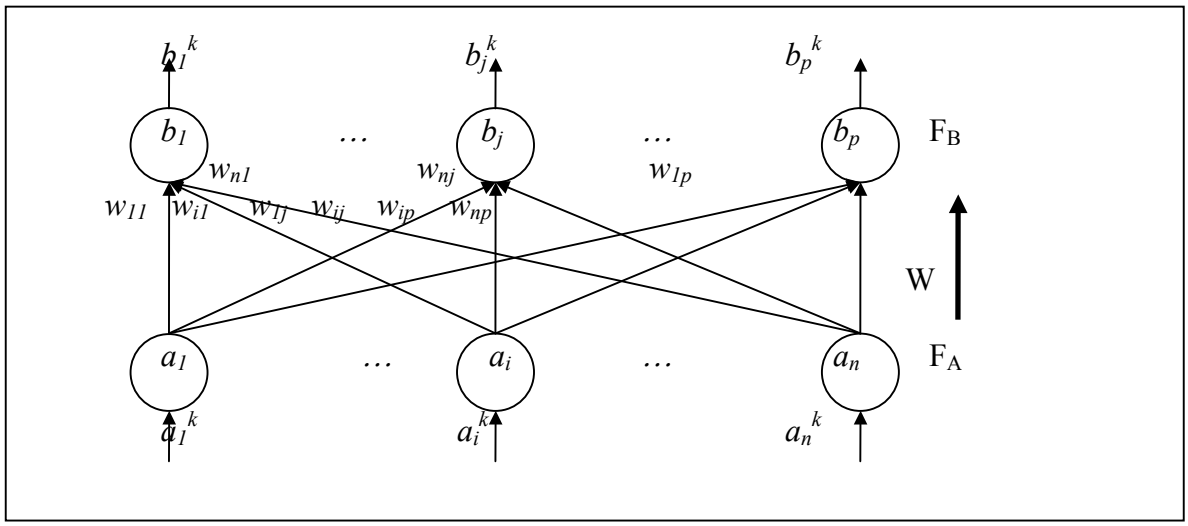


FIGURA 3.3 - Perceptron Elementar.
FONTE: Simpson (1990, p. 101).

É provado que o algoritmo de aprendizagem discutido converge, ou seja, sempre encontra um solução para configuração linearmente separável em tempo finito. Simpson (1990) referencia os trabalhos que provam o teorema da convergência do perceptron.

O perceptron tem capacidade de armazenamento adequada e ativação imediata, mas oferece a desvantagem de ser limitado pela condição de separabilidade linear. Portanto, em situações as categorias não são linearmente separáveis, não podemos usar o perceptron.

Adaline

A rede Adaline (Adaptive Linear), proposta por Widrow e Hoff, é um elemento lógico de limiar adaptativo (Paik et al., 1992) com um número arbitrário de entradas, que podem assumir -1 ou +1, e um elemento de polarização igual +1. As entradas e a polarização ("bias") são modificados pelos pesos correspondentes antes da soma. Na saída um quantificador atribui +1 se a soma é maior que zero ou -1 se a soma é menor que zero.

Considerando um vetor de entrada com L elementos, $\boldsymbol{x}(t)=[x_0(t), x_1(t), \ldots x_L(t)]^T$ onde cada elemento x_i, pode assumir qualquer valor em $\{-j,...,0,...,k\}$, sendo k e j inteiros positivos. Se o conjunto de coeficientes é denotado pelo vetor peso $w(t)= [w_0(t),w_1(t),...,w_L(t)]^T$, a fase de ativação segue o modelo da figura 3.4.

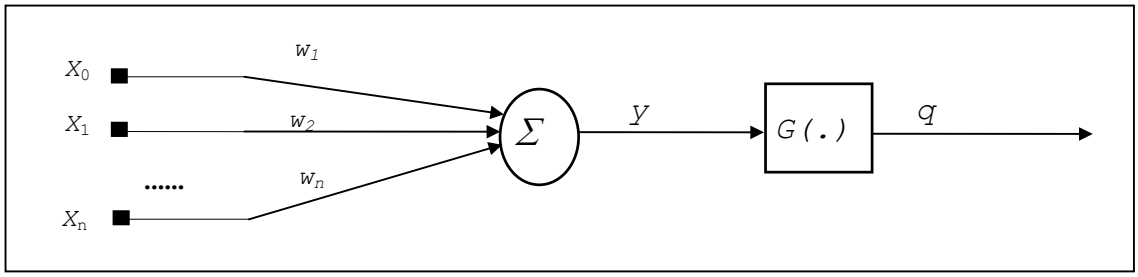


FIGURA 3.4 - Diagrama em bloco da fase de ativação da ADALINE.
FONTE: Paik et al. (1992, p. 1496).

$y(t)$ é o produto interno de $\boldsymbol{x}(t)$ e $\boldsymbol{w}(t)$, $y(t)=\boldsymbol{x}(t)^T\boldsymbol{w}(t)$. Uma entrada $x_0(t)$ constante é associada ao peso $w_0(t)$ para controle do nível de limiar (polarização). $q(t)$ assume valores binários produzidos por $G(.)$ segundo a equação: $q(t)= G(y(t))$, onde $G(.)$ pode ser a função sinal, degrau ou sigmóide. Para a função sinal ou o degrau, tem-se o perceptron.

Widrow e Hoff propuseram um algoritmo que tornam a aprendizagem mais rápida e precisa. Os pesos são ajustados em função do erro na saída, minimizando o erro quadrático médio sobre todos os padrões do conjunto de treinamento.

O método dos mínimos quadráticos médios é calculado pela diferença entre o padrão esperado e a saída da rede. A minimização da soma do erro quadrático envolve uma minimização "gradiente descendente". O cálculo é dado por: $E\{\varepsilon(t)^2\}=E\{[d(t)-y(t)]^2=E\{[d(t)-\boldsymbol{w}^T\boldsymbol{x}(t)]^2$, onde $E\{.\}$ é o valor esperado. Este método permite a aprendizagem mesmo para saídas corretas.

Perceptron de Multicamadas

Esta é um classe importante de redes do tipo "feedforward" (ativação para frente). Basicamente consiste de uma camada de entrada, uma ou mais camadas escondidas e uma camada de saída que aprende “offline” e opera em tempo discreto. Também é possível se ter conexões que ignoram camadas, conexões recorrentes e conexões laterais.

A figura 3.5 ilustra um perceptron de três camadas que armazena pares de padrões espaciais arbitrários $(\boldsymbol{A}_k,\boldsymbol{C}_k)$, k=1,...,m usando um algoritmo (de aprendizagem) de correção de erro baseado no gradiente descendente para multicamadas (minimização do erro quadrático), através da retroprogação ("backpropagation") do erro. Outras funções de minimização podem ser usadas como entropia, erro linear, potências do erro (Simpson, 1990).

O interesse em perceptron multicamadas em ciências cognitiva e da computação, aumentou com o trabalho de Rumelhart, Hinton e Williams em 1986, que exploraram a potencialidade do algoritmo "backpropagation" (Simpson, 1990). Na aprendizagem os valores iniciais dos pesos entre camadas são aleatórios, juntamente com os limiares para cada elemento da rede. Na ativação, um padrão de entrada na camada F_A, gera a ativação de cada elemento nas camadas F_B e F_C dadas por $b_i = f\left(\sum_{h=1}^{n} a_h v_{hi} + \theta_i\right)$, para $i=1,..,p$ e

$c_j = f\left(\sum_{i=1}^{n} b_i v_{ij} + \Gamma_j\right)$, para $j=1,..,q$.

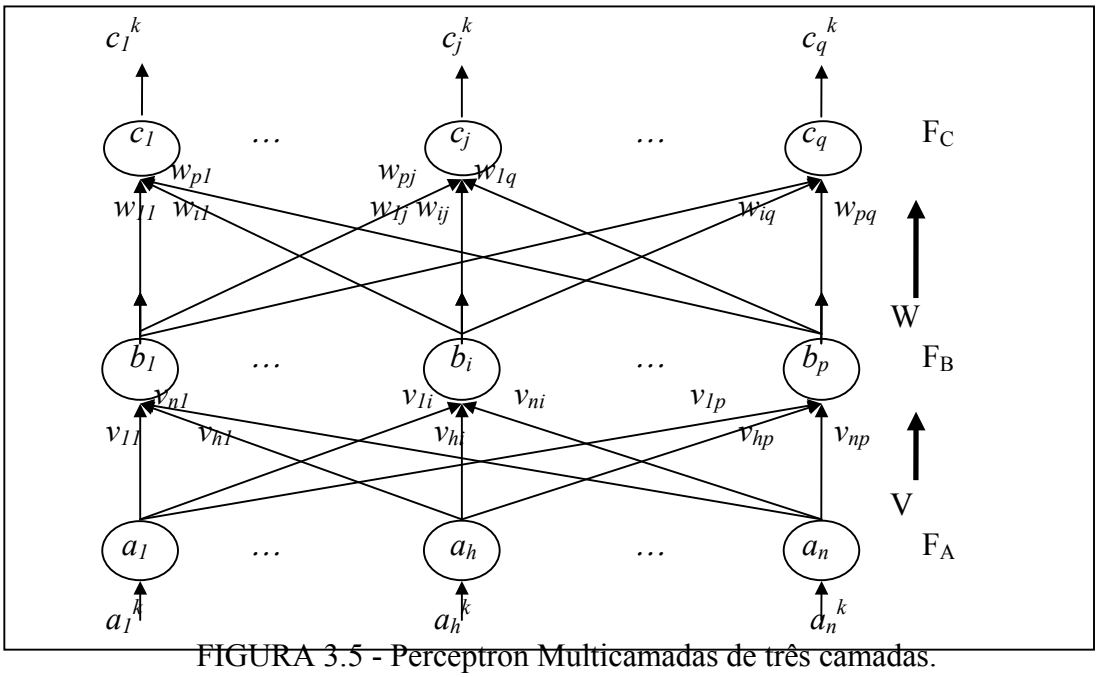


FIGURA 3.5 - Perceptron Multicamadas de três camadas.
FONTE: Simpson (1990, p. 114).

As redes perceptron multicamadas podem ser treinadas através de vários algoritmos de aprendizagem. O algoritmo mais referenciado na literatura é o "backpropagation" (retropropagação do erro) que não tem convergência garantida, i.é, não é garantido que um

erro mínimo global é encontrado. Em geral mínimos locais são encontrados, provocando oscilações nas mudanças dos pesos.

O algoritmo "backpropagation" tem a capacidade de generalização (responder aos padrões de entrada que não estavam no conjunto de treinamento), dependente do número de nós nas camadas internas e do próprio número de camadas internas (escondidas). Uma desvantagem do algoritmo é o longo tempo para treinamento “offline”, o que não impede de ser um dos mais referenciados na literatura em várias áreas de aplicação. Em visão computacional há trabalhos em detecção de bordas e visão estéreo (Srinivasan et al, 1994; Mousavi e Schalkoff, 1994). Em processamento de imagens o algoritmo já foi utilizado reconhecimento de padrões em imagens de sensoriamento remoto. Para melhorar o desempenho do algoritmo "backpropagation", em termos do tempo de aprendizagem, existem várias estratégias possíveis, referenciadas na literatura especializada, que mostram alternativas e melhoramentos aos algoritmos existentes que tentam resolver este problema. Os trabalhos variam desde o monitoramento da taxa de aprendizagem empregada até a escolha adequada de conjuntos de treinamento reduzidos, que mantenham a generalização da rede, em um certa aplicação. O capítulo 6 de Haykin (1994), que trata de perceptrons de multicamadas, mostra algumas alternativas para acelerar a convergência do algoritmo "backpropagation" através da adaptação da taxa de aprendizagem.

A rede "Cascade-Correlation"

Um dos algoritmos alternativos ao "backpropagation" para treinamento de redes multicamadas é o algoritmo "cascade-correlation" (algoritmo da correlação em cascata) concebido por Fahlman (1990) e implementado durante a realização deste estágio utilizando o ambiente o Matlab (Silva 1997). Uma das principais vantagens de uso desse algoritmo é poder iniciar o processo de treinamento da rede sem especificação de camadas escondidas e, gradativamente, acrescentar neurônios escondidos formando assim uma estrutura em cascata à medida que são instalados na rede.

Este algoritmo pode otimizar o tempo de treinamento de uma rede multicamada. Como não podemos adivinhar o tamanho apropriado da rede neural para resolver o problema (uma que tenha tamanho mínimo e um bom desempenho (Haykin, 1994)), pode-se levar bastante tempo na tentativa de se chegar à rede apropriada, variando-se o número de parâmetros livres e/ou o número de camadas escondidas na rede. Entretanto, há técnicas que

permitem que se comece: a) com uma rede grande que é podada durante o treinamento (ou seja, técnica de “prunning” - poda); b) uma rede pequena que é monitorada para a solução do problema e gradualmente é incrementada para o tamanho final apropriado (técnica de “growing” - crescimento)(Haykin, 1994). Também é possível combinar as duas técnicas, "prunning" (poda) e "growing" (crescimento) (Lin e Lee, 1996 página 427). Para o crescimento de redes, pode-se utilizar duas abordagens diferentes: 1) particionamento do espaço de entradas; ou 2) seleção de protótipos (Lin e Lee, 1996 página 427).

A abordagem 1) está relacionada com o particionamento do espaço usando-se o menor número de hiperplanos possível, o que é equivalente a projetar uma rede com o menor número de neurônios possível. Lin e Lee (1996) páginas 427-433, descrevem alguns algoritmos que constroem redes neurais gradualmente.

A abordagem por seleção de protótipos, 2), envolve o uso de uma rede classificadora de padrões que consiste de uma cascata de duas subredes: a rede de casamento de pontos (matching score - MS) e a rede de máximos (MAXNET) ou uma rede do tipo "winner-take-all" (o vencedor leva tudo), que é uma rede recorrente (Lin e Lee, 1996, página 309). No treinamento de uma rede classificadora de padrões, é dado um conjunto grande de entrada com diferentes categorias e a rede cria um conjunto menor de protótipos que estatisticamente representa padrões de categorias diferentes. A subrede MS armazena estes protótipos para casar com os padrões no conjunto de teste, quando atribui um "score" para o grau de casamento. A seleção de protótipos reduz a complexidade e aumenta a habilidade de generalização do classificador (Lin e Lee, 1996). Dado que o número de categorias em um problema de classificação não é conhecido e/ou as entradas para treinamento são produzidas em execução ou em tempo real, a rede de classificação deveria selecionar ou criar um novo protótipo incrementalmente durante o processo de treinamento, o que é equivalente ao processo de criação de novos neurônios, uma vez que cada protótipo é representado por um neurônio na rede MS (Lee e Lin, 1996).

O algoritmo de correlação em cascata foi proposto por Fahlman (1990) como uma arquitetura e um algoritmo de aprendizagem supervisionada para redes neurais. Esta arquitetura treina e automaticamente adiciona novos neurônios à rede neural, um a um, enquanto a aprendizagem não for satisfatória. O resultado final é uma rede com múltiplas camadas. Duas idéias principais estão por trás do algoritmo: 1) a arquitetura em cascata; e 2) o algoritmo de aprendizagem. A arquitetura em cascata significa que os neurônios são

adicionados um a cada vez. O algoritmo de aprendizagem é responsável pela criação e instalação das novas unidades escondidas na rede existente.

Na aprendizagem, para cada novo neurônio escondido, a amplitude da correlação entre a saída do novo neurônio e o sinal de erro residual é maximizada. O neurônio novo recebe uma conexão de cada uma das entradas originais, assim como de cada neurônio escondido já existente. Os pesos da entrada no neurônio escondido são mantidos constantes no momento que o neurônio é adicionado à rede e apenas as conexões na saída são treinadas de forma repetitiva. Assim, cada neurônio adicionado representa uma nova camada (com um neurônio) na rede. O algoritmo de aprendizagem inicia sem neurônios escondidos, e a rede é treinada sobre todo o conjunto de treinamento. Como não há camadas escondidas, usa-se uma regra de aprendizagem sem necessidade de retropropagação do erro, como por exemplo a regra de Widrow-Hoff. Senão houver redução significativa no erro, depois de um certo número de ciclos de treinamento, denominado paciência, e o erro final não for satisfatório, tenta-se reduzir ainda mais os erros residuais, adicionando-se um novo neurônio. Para tanto, toma-se um neurônio candidato que recebe conexões de entrada treináveis a partir de todas as entradas externas da rede e de todos os neurônios escondidos já existentes. A saída do neurônio candidato ainda não é conectada a rede ativa. Em seguida roda-se alguns ciclos sobre o conjunto de treinamento, ajustando-se os pesos de entrada do neurônio candidato após cada passagem, de forma a maximizar S definido por: $S = \sum_o \left| \sum_p (v_p - \bar{v})(E_{p,o} - \overline{E}_o) \right|$ onde: o - é a saída da rede onde o erro é medido; p - é o padrão em treinamento; v - é a saída do neurônio candidato; $E_{p,o}$ - é o erro residual da saída observado no neurônio o (saída); $\bar{v}$ e $\overline{E}_o$ - são os valores médio de v e E_o sobre todos os padrões.

S mede a correlação (ou covariância) entre o valor da saída do neurônio candidato e o erro residual na saída. Para maximizar S, o gradiente $\frac{\partial S}{\partial w_i}$ é obtido como: $\frac{\partial S}{\partial w_i} = \sum_{p,o} \sigma_o (E_{p,o} - \overline{E}_o) a'_p I_{i,p}$ onde σ_o é o sinal da correlação entre o valor do neurônio candidato e a saída o; a'_p é a derivada para o padrão p da função de ativação do neurônio candidato, em relação à soma de suas entradas; $I_{i,p}$ é a entrada que o neurônio candidato recebe a partir do neurônio I para o padrão p.

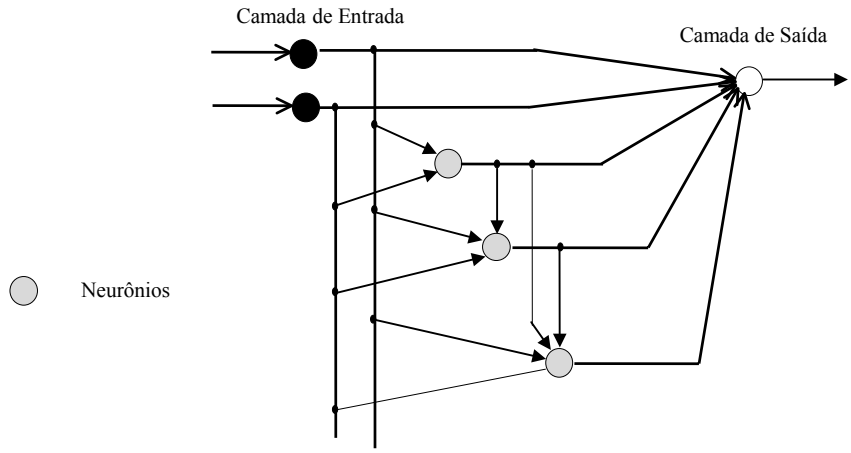


FIGURE 3.6 - Arquitetura em cascata com 3 neurônios escondidos acrescentados. Rede inicial com 2 entradas e 1 saída.

Uma vez que o gradiente $\frac{\partial S}{\partial w_i}$ está disponível, é possível fazer um gradiente ascendente para maximizar *S*. Novamente se está treinando uma camada única de pesos, podendo-se usar a regra delta. Quando *S* pára de crescer, o novo neurônio candidato é instalado como um nó ativo na rede e seus pesos de entrada são mantidos constantes. Os neurônios da saída são treinados pela regra delta novamente e o ciclo se repete até o erro convergir para um valor aceitável.

Em Silva (1997) três experimentos são descritos utilizando a rede "cascade-correlation": o problema do OU-exclusivo, uma aplicação de aproximação de função e uma aplicação de modelagem.

Kohonen

A rede de Kohonen é auto-organizável e consiste de uma camada bidimensional de neurônios arranjados espacialmente. Todas as entradas são conectadas a cada nó na rede. A realimentação está restrita a interconexões laterais entre nós vizinhos. Não há camada de

saída separada, cada neurônio na rede é também um neurônio de saída. A figura 3.7 ilustra um mapa de características, de Kohonen.

O algoritmo de aprendizagem organiza os nós na grade em vizinhanças locais, que funcionam como classificadores de características sobre os dados de entrada. O mapa topográfico é organizado autonomamente por um processo cíclico de comparação dos padrões de entrada com os vetores armazenados em cada nó. A aprendizagem é não supervisionada.

Nos locais onde as entradas casam com os vetores nós, a área do mapa é otimizada seletivamente para representar uma média dos dados de treinamento para aquela classe. A partir de um conjunto de nós aleatoriamente organizado, a grade ajusta-se em um mapa que tem representação local e é auto-organizado.

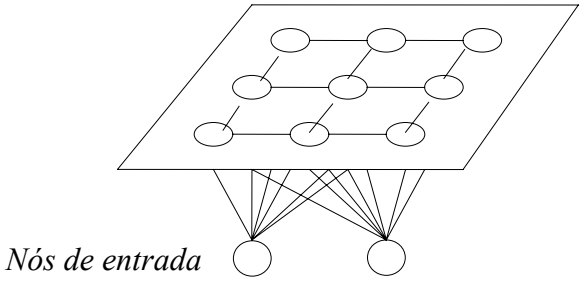


FIGURA 3.7 - Mapa de Kohonen.
FONTE: Beale e Jackson (1992, p. 110).

O algoritmo da rede de Kohonen funciona da seguinte forma:

a) Inicialização.

Define-se $w_{ij}(t)$ $(0 \leq i \leq n\text{-}1)$ como o peso entre a entrada i e o nó j no tempo t. Os pesos são inicializados para valores aleatórios pequenos. Toma-se o raio da vizinhança em torno do nó j, $N_j(0)$, muito grande.

b) Apresenta-se a entrada $x_0(t),x_1(t),..,x_{n-1}(t)$; $x_i(t)$ é a entrada para o nó i no tempo t.

c) Calcula-se as distâncias d_j entre a entrada e cada nó de saída j utilizando o método da distância Euclideana:

$$d_j = \sum_{i=0}^{n-1} \left(x_i(t) - w_{ij}(t) \right)^2$$

d) Seleciona-se a menor distância, atribuindo índice j^*.

e) Atualiza-se os pesos para o nó j^* e seus vizinhos, definidos pelo tamanho da vizinhança $N_{j^*}(t)$. Os novos pesos serão:

$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(x_i(t) - w_{ij}(t))$. Para j em $N_{j^*}(t)$, $0 \leq i \leq n\text{-}1$

$\eta(t)$ é um ganho, $(0<\eta(t)<1)$ que decresce no tempo, diminuindo a adaptação dos pesos. A vizinhança $N_{j^*}(t)$ decresce com o tempo, convergindo para área de atividade mais alta.
f) Repete-se todos os passos a partir de b).

O esquema de aprendizagem empregado é o "winner-take- all" (vencedor leva tudo), apropriado para agrupamentos e classificações. Entretanto, as redes com este esquema só podem ser treinadas se as classes ou grupos são linearmente separáveis por hiperplanos que passam pela origem.

A rede de Kohonen foi usada em reconhecimento de objetos (Bebis e Papadourakis, 1992) e em segmentação de imagens em multiescala (Haring et al, 1994).

Neocognitron

O neocognitron foi proposto por Fukushima (1982) como modelo do mecanismo de reconhecimento de padrões visuais, com plausibilidade biológica. “Modelar redes neurais ajuda a descobrir o mecanismo do cérebro” (Fukushima, 1988). A rede original é auto-organizável com aprendizagem não-supervisionada, e pode reconhecer padrões mesmo com variações em posição e forma. Outras versões do modelo utilizam aprendizado supervisionado.

Para emular o sistema visual, começando pela retina, Fukushima (1988) tomou como base considerações fisiológicas de que as áreas visuais do cérebro apresentam neurônios que respondem de forma seletiva a características locais em um padrão, tais como bordas e linhas em certas orientações. Células em áreas superiores do córtex visual respondem a certas figuras como círculos, triângulos, quadrados ou mesmo faces humanas. Assim, o cérebro possui uma estrutura hierárquica para extração das primeiras características de um padrão visual. As características em seguida são integradas em outras mais complexas. Dentro desta hierarquia, uma célula em estágios superiores geralmente recebe sinais de uma grande área da retina, sendo mais sensível à posição do estímulo, e os sinais fluem em sentido para frente e para trás.

A rede neocognitron tenta modelar tal mecanismo. A rede é analógica, ou seja, as entradas e saídas têm valores analógicos não negativos. Uma camada de entrada é seguida por um número de módulos conectados em série. Cada módulo consiste de duas camadas, a primeira com células "S" (células simples do córtex visual) e a segunda com células "C” (células mais complexas). Cada célula “S” tem várias entradas, excitatórias ou inibitórias.

Para sinais de entradas excitatórios a saída da célula cresce e para sinais inibitórios ela decresce. Cada componente do vetor que forma a entrada tem seu próprio peso com valor positivo, que pondera as entradas.

A saída da célula pode ser conectada a várias entradas de outras células. Para uma célula S, a saída é dada por $w = \varphi\left[\left(\left(1 + \sum_{i=1}^{N} a(i)u(i)\right) / \left(1 + bv\right) \right) - 1 \right]$, onde *a(i)* e *b* são coeficientes de interconexões excitatórias e inibitórias, respectivamente. *N* é o número de unidades de entrada excitatórias (*u(1),...,u(N)),* e *v* é a entrada inibitória. $\varphi(.)$ é a função $\varphi(x) = \begin{cases} x, & (x \geq 0) \\ 0, & (x < 0) \end{cases}$. Fazendo $e = \sum_{i=1}^{N} a(i)u(i)$ e $h=b.v$ reescreve-se $w = \varphi\left[\frac{1+e}{1+h} - 1\right] = \varphi\left[\frac{e-h}{1+h}\right]$. A figura 3.9 ilustra uma célula usada no neocognitron.

As células “C” também têm características similares às células “S”, mudando-se $\varphi(.)$ para $\psi(x) = \begin{cases} x / \left(\alpha + x\right), & (x \geq 0) \\ 0, & (x < 0) \end{cases}$ onde α é uma constante positiva que determina o grau de saturação da saída.

Os pesos das ligações entre as células "C" em uma camada e células "S" na próxima camada são modificáveis, assim como os pesos entre a entrada e a outra camada na célula "C". Os pesos intracamadas são fixos. Pode existir vários "planos" (níveis) dentro de cada camada. Cada célula recebe uma entrada de uma camada relativamente pequena e fixa que a antecede. Quando a camada de saída é alcançada, uma célula de saída vê a entrada inteira como um resultado deste efeito telescópico de decrescer o número de células em cada plano com a profundidade na rede, como está ilustrado na figura 3.8.

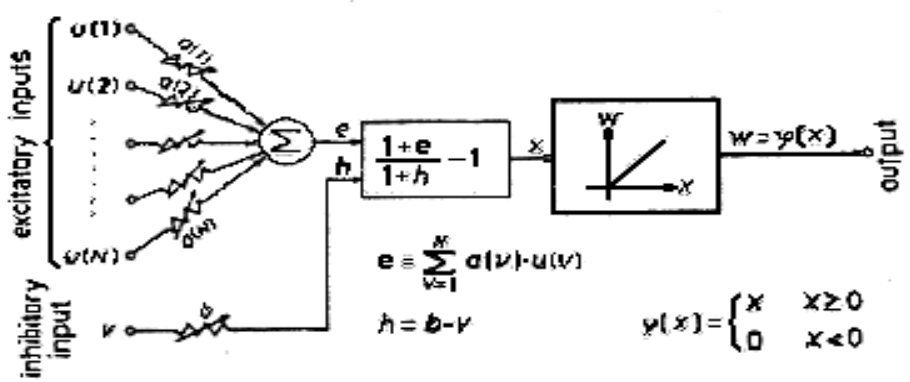


FIGURA 3.8 - Célula S.
FONTE: Fukushima (1982, p. 456).

A auto-organização se dá por aprendizagem não-supervisionada. Um conjunto de padrões é apresentado várias vezes à entrada. Um característica do neocognitron é a invariância a posição, forma e tamanho do padrão de entrada, mas a implementação é problemática devido ao número de nós e caminhos que eleva a complexidade.

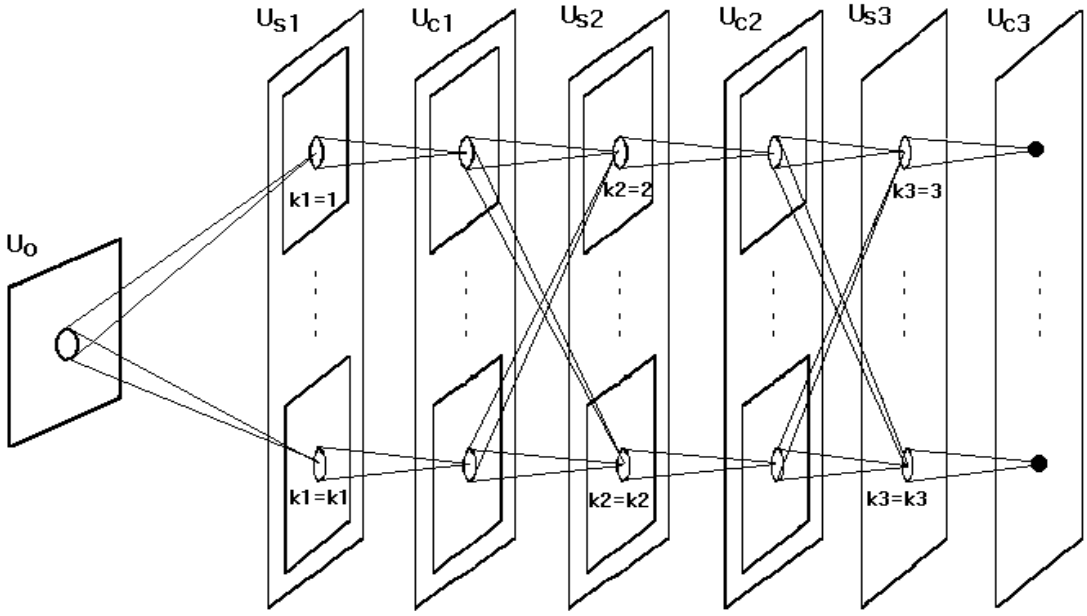


FIGURA 3.9 - Conexões no neocognitron.
FONTE: Fukushima (1982,p. 459).

A rede neocognitron foi usada para reconhecer os dígitos (0-9) na forma escrita (Fukushima, 1982), e também para reconhecimento visual, cite-se por exemplo, o trabalho de Santos e Oliveira (1995) para identificação de veículos.

Hopfield

Hopfield estudou uma rede auto-associativa similar ao perceptron, propondo também uma função de energia. A rede consiste de vários neurônios conectados entre si conforme figura 3.10.

A rede é simétrica pois os pesos nas conexões são os mesmos em ambas as direções. Cada nó tem um limiar (polarização) e uma função limiar do tipo degrau. As entradas podem ser (0,1) ou (-1,+1). A rede se diferencia de outras pela forma de produzir uma solução.

A aprendizagem é feita alterando-se os pesos das conexões em função da entrada. Inicializa-se a rede com um padrão desconhecido $\mu_i(0) = x_i$, $0 \le i \le N\text{-}1$, onde $\mu_i(t)$ é a saída do nó i no tempo t. As iterações continuam até atingir a convergência $\mu_i(t+1) = f\left[\sum_{i=0}^{N-1} w_{ij}\,\mu_j(t)\right]$, $0 \le j \le N\text{-}1$, onde $f(.)$ é a função degrau, não linear. Isto acontece para todos os padrões de todas as classes.

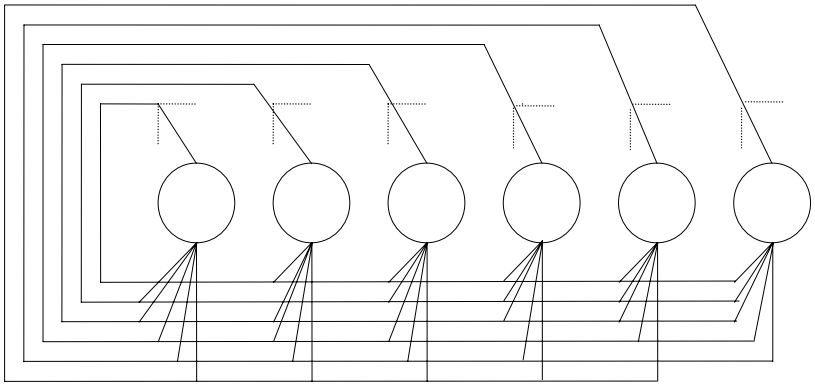


FIGURA 3.10 - A rede Hopfield.
FONTE: Beale e Jackson (1990, p. 134).

Na ativação a saída da rede é forçada a casar com um padrão desconhecido no tempo zero. Ela então começa a interagir livremente até alcançar uma situação estável quando a saída permanece inalterada. Neste ponto diz-se que a rede convergiu para a solução. Esta auto-associação de padrões significa que a apresentação de um padrão corrompido, resultará na ativação do padrão perfeito na saída.

A função de energia na rede de Hopfield é dada por $E = -\frac{1}{2}\sum_i \sum_{j \ne i} w_{ij} x_i x_j + \sum_i x_i T_i$

onde w_{ij} é o peso entre um nó i e o nó j e x_i é a saída no nó i. T_i é o limiar do nó i. Os pesos retratam a informação do padrão, portanto, todos os padrões estão incluídos na função de energia. Como os nós não são conectados com eles mesmo, $w_{ii} = 0$.

A rede Hopfield tem uso em várias aplicações. Em aplicações de visão computacional (Bhuiyan et al., 1994) usaram uma rede Hopfield para detecção de bordas e (Lee et al., 1994) utilizaram uma rede para visão estéreo.

Etapa 4. Levantamento e caracterização do problema de análise de imagem a ser abordado.

Muitos problemas podem ser abordados na área de Processamento de Imagens e de Visão Computacional, com o objetivo de adquirir maior familiaridade com a utilização do paradigma de redes neurais nas aplicações destas áreas. Entretanto, em cada uma delas existem inúmeros problemas relacionados com o projeto e adequação de uma rede neural para resolver o problema pretendido, bem como, como é comentado na seção relativa a etapa 5 a seguir, os aspectos relativos à implementação, que podem ser amenizados pela utilização de um ambiente de programação versátil.

Um mesmo problema pode ser resolvido por redes distintas, utilizando aprendizagens diferentes. Isso significa um universo de possibilidades de uso das redes neurais nas tarefas de qualquer área de aplicação, em particular nas áreas de Processamento de Imagens e Visão Computacional.

Em Uhrig (1995) são citados várias possibilidades de uso de redes neurais em aplicações a engenharia nuclear. Isso reflete que os grupos envolvidos devem considerar muitos aspectos de engenharia no projeto de uma rede a ser utilizada.

Neste trabalho, a aplicação de redes neurais esteve restrita a um problema de classificação de imagens de sensoriamento remoto, utilizando dados do satélite Landsat 5 obtidos no Instituto Nacional de Pesquisas Espaciais - INPE. A restrição a uma aplicação deve-se ao fato de haver necessidade de familiarização com: o ambiente de programação, os modelos de redes possíveis e os aspectos de engenharia envolvidos no projeto de uma rede. Entretanto, neste relatório são relatados alguns trabalhos que exibem resultados de solução de problemas utilizando redes neurais em processamento de imagens e visão computacional (ver etapa 2).

O trabalho abordado tem como objetivo identificar alvos em imagens de satélites. O processo consiste na obtenção de amostras para classificação e no desenvolvimento do algoritmo com redes neurais a ser utilizado.

A classificação de imagens de satélite em Sensoriamento Remoto é importante, dado que um especialista da área pode extrapolar resultados de classificação de uma área reduzida na imagem, para uma área mais abrangente. A verdade terrestre pode ser levantada a partir de pequenas áreas de estudo e os resultados de classificação são então estendidos para as várias áreas dentro da imagem, que apresentam características radiométricas similares.

Portanto, também há redução de gastos com trabalhos de campo para identificação de verdades terrestres.

As imagens utilizadas formam um conjunto das bandas 2, 3 e 4 do Landsat 5, da área de Brasília, Distrito Federal. Como o enfoque do treinamento era proficiência no uso de redes neurais aplicadas no Processamento de Imagens e Visão Computacional, os alvos escolhidos para classificação foram estabelecidos através da observação das imagens, sendo eles: água, vegetação, área residencial e solo exposto.

Na seção referente a etapa 8, "Identificação de atributos e dados para uso na fase de análise, em função do problema específico", são feitas as considerações de como as diversas amostras para classificação foram selecionadas.

Etapa 5. Análise da adaptabilidade de modelos de redes neurais na modelagem dos problemas.

Possivelmente a forma mais eficiente para determinar se uma aplicação de redes neurais é satisfatória, é comparar os resultados obtidos com aqueles conseguidos através de técnicas já existentes, que em certas aplicações podem apresentar disparidades por falta de suficiência de modelo.

Tsoukalas e Uhrig (1997) e Haykin (1996) enumeram alguns itens que devem ser observados na decisão de se utilizar redes neurais para resolver um problema. Entre eles estão: aplicações onde a tecnologia de computação existente não seja adequada; problemas que requerem raciocínio qualitativo ou quantitativo complexo; problemas cujas soluções provêm de parâmetros extremamente interdependentes e que não têm quantificação precisa; aplicações que envolvem parâmetros múltiplos interagindo; aplicações envolvendo dados ruidosos; aplicações com dados incompletos; etc. Todos esses tópicos refletem aspectos de engenharia, que são importantes quando se considera aplicações de situações não simuladas. Haykin (1996) discute critérios práticos para aceitação de uma rede neural, para a resolver um problema em processamento de sinais, dado que os seguintes atributos são observados: preservação ótima de informação disponível e performance ótima, dentro de uma visão estatística; robustez na performance com respeito a pequenas variações nas condições ambientais. Se estes atributos são observados, as redes neurais podem ganhar aceitação como ferramentas para resolver problemas práticos.

Um dos problemas de uso das redes neurais está relacionado com o tempo usado no treinamento, que é um problema mal-condicionado ("ill-posed") (Haykin, 1996), e que pode ser influenciado por fatores como: 1) escolha do tipo de rede; 2) quantidade de dados no conjunto de treinamento disponível; 3) arquitetura da rede utilizada; 4) natureza dos dados nos conjuntos de treinamento; e 5) limitações de máquina, por exemplo. Como forma de superar estes problemas de início, Tsoukalas e Uhrig (1997) comentam vários aspectos práticos do uso de redes neurais, que englobam: a) seleção da rede neural adequada para a solução de um problema; b) projeto da rede neural; c) fontes e Processamento de dados; d) Representação dos dados; e) escalonamento e normalização; e f) seleção de dados para treinamento e testes.

A seleção da rede neural, a), depende dos dados disponíveis para a aplicação. Se os dados consistirem de pares de entrada e saída, ou seja, se além dos dados de entrada, existir os dados que se deseja, então é possível se escolher uma rede neural com treinamento supervisionado. Em geral estas situações são difíceis do ponto de vista prático, pois nem sempre é fácil se obter tais pares de dados de entrada e saída, ou seja, nem sempre se tem conhecimento a priori do que se espera que a rede neural classifique, aproxime ou modele. Para aplicações que não favorecem a supervisão, o treinamento não supervisionado classifica os dados de entrada segundo propriedades dos padrões dos agrupamentos possíveis, traduzidas por medidas de similaridades associadas aos padrões.

Como mencionado anteriormente, um aspecto importante relativo ao modelo de rede a ser utilizado é o tempo necessário para treinamento e para ativação. Em geral as redes neurais demandam bastante tempo para treinamento, mas na ativação uma única passagem é suficiente. O tempo de ativação pode ser ainda menor, quase instantâneo, se a rede for implementada em "hardware" (circuitos digitais).

No projeto de uma rede neural artificial, b), os pontos seguintes devem ser observados: o tamanho da rede neural; a escolha da saída; o tipo de função de ativação para as unidades; o número de camadas; e o número de neurônios em cada camada. O tamanho da rede é dependente da natureza do problema e da experiência do usuário. Os novatos no uso de redes neurais preferem redes pequenas, enquanto tentam reduzir o problema. Para os usuários experientes, é preferível que o problema decida o tamanho da rede a ser usada. A escolha da saída está relacionada com a natureza da aplicação, que pode requerer ativação binária ou multivalorada. Números reais na saída podem ser traduzidos em unidades (monetária, tempo, distância) podendo ser codificadas em forma binária ou em multivalores. As saídas nas redes

neurais são geralmente interpretadas como classificações, padrões, números reais e escolha ótima, todas apresentando requisitos específicos. Para o caso de classificações que mapeam estatisticamente os padrões de entrada em categorias discretas, faz-se necessário se ter 2 ou mais neurônios de saída, sendo que apenas um será ativado para uma dada entrada. Por outro lado, as redes que identificam padrões freqüentemente têm múltiplos neurônios na saída, sendo que todos são ativados ao mesmo tempo, o que forma um padrão em resposta a entrada.

A função de transferência no neurônio tem como objetivo principal manter as saídas dos neurônios dentro de limites razoáveis para evitar saturação. As funções degrau ou sinal (limiar) são usadas quando as entradas e saídas são binárias (0 e 1) ou bipolares (-1 e +1). Outras funções utilizadas são a linear, a sigmoide e a tangente hiperbólica. A escolha entre elas baseia-se nos tipos de entrada e saída e também no algoritmo de aprendizagem utilizado. Alguns algoritmos requerem que a derivada da função de transferência seja contínua, restringindo a escolha às funções do tipo sigmoide e tangente hiperbólica. As funções tipo degrau e sinal são preferidas quando se tem em mãos pares de entradas e saídas binárias.

Alguns modelos de redes neurais por definição já têm um número fixo de camadas, por exemplo, Adaline, Madaline, Hopfield, Art-1, Mapas de Kohonen e Memórias Associativas, que requerem 2 ou 3 camadas. No caso de redes do tipo perceptron de multicamadas, há a necessidade de ser ter as camadas escondidas, que podem variar em número dependendo da complexidade do problema a ser tratado. Estas camadas escondidas funcionam como extratores de características das entradas. Quando usadas em grande número podem melhorar significativamente a força de processamento da rede, mas demandar mais tempo de treinamento e conjunto de treinamento maiores. Tsoukalas e Uhrig (1997) comentam que uma rede de 3 camadas (entrada, escondida e saída), com um número suficiente de neurônios, é adequada para representar qualquer mapeamento, e que novas camadas só devem ser acrescentadas se a arquitetura em mãos não for adequada.

O número de neurônios em cada camada também são determinados pela natureza do problema em mãos. Em particular, o número de neurônios nas camadas de entrada e de saída dependem das dimensões dos vetores utilizados para representá-las. Já em se tratando das camadas internas, o número de neurônios é determinado através de experimentação, por ainda não existir métodos capazes de calcular precisamente o número de neurônios requerido para um problema. Haykin (1994) no capítulo 6, cita uma abordagem numérica para calcular o número de neurônios provavelmente necessários para resolver um problema. Os resultados encontrados foram conseguidos através de experimentos. Duas situações podem ocorrer, que

exigem a interação na determinação do número de neurônios na camada escondida: um número de neurônios insuficiente ou um número maior do que realmente é necessário. Poucos neurônios na camada escondida fará com que ela não faça o mapeamento correto das entradas para a saída, enquanto neurônios em excesso degrada a generalização e aumenta o tempo de treinamento. Tsoukalas e Uhrig (1997) enfatizam que muitos neurônios, na realidade, fazem a rede memorizar os padrões sem extrair as características pertinentes para generalização.

As fontes e o processamento de dados para redes neurais, c), deseja-se que o conjunto de treinamento de seja representativo dos padrões a ser reconhecidos e que englobe toda a faixa de padrões de entrada, para que a rede consiga ser eficiente na generalização, com capacidade de extrapolação e interpolação. Todos os dados relacionados com a aplicação devem ser analisados para remover dados tecnicamente considerados espúrios, o que pode ser feito com pré-processamento dos dados. No caso de conjuntos de dados incompletos, o que pode causar problemas, é possível criá-los a partir de estimativas com base nos dados presentes utilizando técnicas como por exemplo, tomar o valor esperado da variável em questão, ou a mediana, etc.

Um base de dados pode apresentar variações temporais, importantes. Uma rede neural pode detectar as tendências nessas bases dados temporais. Por exemplo, uma aplicação que envolve a predição de vendas em período futuro necessita que a rede seja treinada com informações referentes até o período anterior à previsão desejada, caso deseje-se maior precisão. Mas em aplicações onde a disponibilidade de dados não é dinâmica, pode-se usar dados existentes para o treinamento, que cobrem vários períodos de tempo, dependendo da unidade utilizada para análise, e usar o os dados do período a seguir como saída desejada.

Quando a informação é representada em forma de imagem, os dados para a rede neural são mais adequados se forem representados de forma não distribuída. Para o caso de imagem em nível de cinza, cada neurônio recebe um número que representa a intensidade de um pixel, o elemento de imagem, do campo visual para cada ponto na imagem, assim como, os parâmetros que indicam a locação do pixel. O maior problema com as imagens é que elas possuem muitos pixels, o que implica bastante tempo para o treinamento. Por exemplo, se uma imagem tem dimensões 1024 x 1024, mais de 1 milhão de pixels existirão, o que tornar o treinamento proibitivo. Nas aplicações práticas pode-se aplicar técnicas de extração de características para formar descritores da imagem que podem ser usados no treinamento da rede, reduzindo consideravelmente o tempo de treinamento.

As aplicações que necessitam de entrada de dados sensoriados são prejudicadas quando há a interrupção da transferência de dados do equipamento. Portanto, os meios de aquisição de dados devem ser testados exaustivamente, para evitar problemas no treinamento da rede, pois se em um conjunto de sensores faltar um, o sistema poderá dar resultados não desejados.

A representação dos dados, d), trata da conversão dos mesmos de uma forma, para outra mais significativa, ou seja, que tenha mais representatividade dos padrões existentes no conjunto de treinamento. Os dados podem ser fornecidos para a rede em forma de valores contínuos ou binários. Valores binários são idéias para situações onde ocorrem grupos naturais, pois constituem um método melhor para fazer correlações. Valores binários devem ser evitados quando se tem valores muito contínuos, pois do contrário, é difícil para a rede aprender exemplos com valores perto ou na fronteira entre dois grupos (Tsoukalas e Uhrig, 1997). Outro aspecto considerado na representação de dados é a codificação dos dados, ou seja, conversão dos dados para forma adequada para apresentar à rede neural. Junto com isso um algoritmo de decodificação deve ser idealizado de imediato. Apesar da codificação e decodificação ser dependentes do modelo de rede, deve-se observar que os neurônios trabalham com entradas e saídas que correspondem ao valores de ativação dos neurônios (Tsoukalas e Uhrig, 1997), assim, a codificação precisa gerar valores dentro das faixas "entendidas" pela função de ativação. A entrada deve interpretar os dados brutos e a decodificação da saída deve tomar a seqüência de números obtidos e transformar na forma requerida pelo usuário.

Outros aspectos importantes relativo ao projeto de redes neurais são o escalonamento e a normalização, e). A necessidade destes processamentos decorre da sensibilidade das redes neurais às magnitudes absolutas. Se uma entrada está na faixa entre 1.000 e 1.000.000 e uma segunda entrada está entre 0 e 1, as flutuações na primeira entrada fará com que a segunda tenha sua importância diminuída, o que pode tornar-se um problema se a segunda entrada, por exemplo, for muito mais importante para a predição da saída desejada. A minimização dessa influência todas as entradas pode ser conseguidas se as entradas são escalonadas e normalizadas, para corresponder à mesma faixa de valores, em geral 0 a 1 ou -1 a 1.

Uma das características das redes neurais é o poder de processamento em situações que envolvem não-linearidades. Por outro lado, o uso de relações lineares torna mais fácil a aprendizagem e a emulação em redes neurais. Por tanto, a tentativa de minimização de não-linearidades nos problemas, significa a implementação de redes neurais mais simples, com

treinamento mais rápido e melhor desempenho no geral. A minimização da não-linearidade pode ser conseguida através da preparação dos dados.

A normalização dos dados consiste em dividir os valores de um conjunto de entrada por uma referência arbitrária, geralmente escolhida como o valor máximo, o que faz o novo máximo ser 1. Este processo pode significar perda de informação quando se tem no conjunto, valores muito maiores ("spikes" - anomalias) do que a grande maioria, ou mesmo quando todos os dados estão dentro de uma faixa estreita. Tomando-se o escalonamento, estabelece-se uma relação linear entre duas variáveis dentro de uma faixa escolhida para cada uma delas. A normalização pode ser encarada como um caso especial do escalonamento, quando o valor mínimo de ambas as variáveis é zero.

Se escolhermos a faixa de atuação com valor mínimo 0.1 e valor máximo 0.9, então para um conjunto de entrada *x*, seus valores são convertidos para o conjunto *y* escalonado, através da relação: $y = mx + b$, assim $y = 0.1$ para $x = x_{min}$ e $y = 0.9$ para $x = x_{max}$, onde $m = 0.8/(x_{max}-x_{min})$ e $b = 0.9-0.8x_{max}/(x_{max}-x_{min})$

Portanto, $$y = \frac{0.8}{x_{\max} - x_{\min}} x + 0.9 - \frac{0.8 x_{\max}}{x_{\max} - x_{\min}}$$

O escalonamento linear com valores entre 0.1 e 0.9 é preferível quando se usa a função de transferência sigmóide, o que evita a paralisia na aprendizagem, que significa a repetição de um mesmo valor do erro a partir de um ciclo de aprendizagem, causada pela saturação da função de transferência. No caso da função de ativação tangente hiperbólica, é preferível usar a técnica de "Z scores" (o número de desvios padrão acima e abaixo da média) (Tsoukalas e Uhrig, 1997). Para esta transformação primeiro se calcula a média e o desvio padrão para a variável em todo o conjunto de treinamento. Depois converte-se cada valor para um "Z score", subtraindo-se a média e dividindo a diferença pelo desvio padrão, ou seja, $y = (x - \mu) / \sigma$, onde μ é a média e σ é o desvio padrão para a variável *x* dentro do conjunto de treinamento.

A seleção de dados para treinamento e testes, f), também se constitui em um passo importante, pois para se treinar uma rede neural tudo que se precisa é uma quantidade adequada do tipo de informação importante para a solução do problema. Em situações em que há dúvidas sobre a importância dos dados é preferível incluir os dados, pois a rede neural pode aprender a ignorar as entradas que têm pouca importância para o problema se houver exemplos suficientes no conjunto de treinamento. O uso de dados inadequados torna as

correlações mais difíceis. O tempo de treinamento pode crescer muito se não houver tipos de dados suficientes para as associações, o que ocorre freqüentemente com redes do tipo "backpropagation" com um número excessivo de neurônios escondidos, pois a rede memoriza valores individuais, sendo bem treinada mas com baixo desempenho em testes com dados novos.

Etapa 6. Familiarização com o ambiente computacional e as ferramentas disponíveis para a simulação dos modelos.

O uso de redes neurais em diversas áreas de aplicação, tornou possível o surgimento de vários tipos de sistemas para simulação dos modelos de redes existentes. A maioria destes sistemas são pacotes fechados, ideais para usuários que procuram utilizar o paradigma de redes neurais para a solução de problemas na sua área de aplicação. Entretanto, para fins acadêmicos e de treinamento no paradigma, é preferível se ter sistemas ou ambientes computacionais abertos que favoreçam o desenvolvimento dos algoritmos existentes. Assim pode-se proceder com modificações nas implementações, procurando tornar os algoritmos mais elaborados, para aumentar as possibilidades dos modelos de redes neurais artificiais existentes. Para isso, tais sistemas ou ambientes computacionais precisam dispor de linguagens de programação de fácil compreensão, pois como se trata de implementações de modelos computacionais para solução de problemas, faz-se necessário que ênfase seja dada aos aspectos teóricos do paradigma e não aos aspectos de implementação.

No ambiente de desenvolvimento deste trabalho, utiliza-se a caixa de ferramentas ("toolbox") de redes neurais do ambiente MATLAB, licenciado pela Mathworks Inc.(1997).

Todos os experimentos realizados foram desenvolvidos no MATLAB. Este software provê os meios necessários para a implementação dos algoritmos de redes neurais, tornando o trabalho de implementação voltado para os aspectos mais teóricos dos modelos, o que permite a exploração de várias possibilidades de combinações distintas, sem empreendimento demasiado de tempo na solução de problemas de implementação.

O **MATLAB** é um sistema interativo baseado em matrizes, para cálculos científicos e de engenharia. O nome MATLAB em si é a abreviação de **MAT**rix **LAB**oratory (Laboratório de Matrizes).

Como sistema, o MATLAB é composto de um núcleo que interliga os "toolboxes" e os interpreta em tempo de execução. O núcleo (kernel) recebe o comando e vê se ele existe nos vários "toolboxes" ou se faz parte das funções básicas do MATLAB. Se a função existir é executada, do contrário o ambiente sinaliza para o usuário. A idéia é ter um "toolbox" para cada área de aplicação, com possibilidade de interação entre eles, possibilitando a combinação de abordagens distintas para a solução dos problemas.

Cada "toolbox" possui um manual que rever tópicos teóricos da área de aplicação e introduz experimentos demonstrativos da utilização das várias funções disponíveis.

Por se tratar de um sistema também voltado para cálculos científicos, o MATLAB é um sistema aberto, oferecendo recursos para modificar e/ou acrescentar aplicativos nos "toolboxes" existentes, bem como para criar novos "toolboxes".

Os "toolboxes" são conjuntos de funções codificadas na linguagem de programação do MATLAB e interpretadas em tempo de execução pelo núcleo (kernel). Tais funções podem ser traduzidas para outras linguagens de programação compiladas, permitindo a utilização de MATLAB em aplicações que demandam pouco tempo de processamento. A tradução das funções é possível, devido a existência de "toolboxes", desenvolvidos para permitir a portabilidade das funções para outros sistemas computacionais.

O "toolbox" de redes neurais constitui-se em um conjunto de funções que implementam alguns modelos de redes neurais mais utilizados e referenciados na literatura. Entretanto, a linguagem de programação do ambiente permite que novos modelos sejam desenvolvidos. Na versão atual do "toolbox" é possível implementar modelos tais como: perceptron, redes lineares, backpropagation, redes de bases radiais (radial basis networks), redes auto-associativas e redes recorrentes.

A familiarização com o ambiente foi feito por etapas. Primeiro procurou-se utilizar as implementações dos modelos de redes neurais existentes, utilizando os vários experimentos disponíveis no "toolbox". Como os experimentos são abertos, foi possível proceder com modificações no projeto dos mesmos procurando extrair todas as possibilidades de uso. Esta primeira etapa permitiu a familiarização com o funcionamento do ambiente na interpretação das funções desenvolvidas, bem como, com a combinação de diferente "toolboxes".

Em uma segunda etapa alguns algoritmos de redes neurais foram desenvolvidos utilizando-se a linguagem de programação do MATLAB, o que permitiu a familiarização com um ambiente de programação matricial.

Etapa 7. Pré-processamento das imagens para a normalização dos dados para os algoritmos de redes neurais.

No estágio foram realizados experimentos, implementando-se algoritmos computacionais para reduzir os conjuntos de treinamento através do uso de técnicas de clusterização. Os experimentos são comentados no decorrer deste relatório. Para se obter redes o mais concisas possíveis também foi objeto de estudo no estágio, a implementação de um algoritmo de crescimento dinâmico de uma rede neural, o algoritmo Cascade-correlation.

A normalização dos dados de entrada foi feita usando-se escalonamento linear como mencionado na seção referente à etapa 5 da análise da adaptabilidade de modelos de redes neurais na modelagem dos problemas.

O modelo de rede utilizado foi um perceptron de multicamadas treinado pelo algoritmo "backpropagation" citado na seção referente à etapa 3 do estudo teórico de alguns algoritmos de redes neurais.

A rede poderia ter duas arquiteturas diferentes, uma de três camadas (camada de entrada, camada escondida e camada de saída) e quatro camadas (camada de entrada, 2 camadas escondidas e camada de saída). Uma rede de três camadas foi escolhida para resolver o problema contendo neurônios com função de ativação do tipo sigmoide (figura 3.2). O fato de se escolher escalonamento linear para os dados de entradas deveu-se exatamente ao fato de que a camada escondida tinha neurônios do tipo sigmoide, cujo intervalo de definição está entre 0 e 1. Entretanto, observando a curva da função sigmoide na figura 3.2, verifica-se que abaixo de 0.1 e acima de 0.9 a função satura facilmente. Portanto, se os dados de entrada forem normalizados neste intervalo, podemos diminuir eventuais saturações da função de ativação.

Etapa 8. Identificação de atributos e dados para uso na fase de análise, em função do problema específico.

Esta fase esta relacionada com a aquisição das amostras para formação do conjunto de treinamento para a rede neural a ser utilizada.

O problema a ser resolvido é classificar dados de satélite multiespectrais. No trabalho três bandas (faixas espectrais) são dadas correspondendo cada uma a um sensor sensível a um

comprimento de onda específico. Os dados foram adquiridos nos comprimentos de onda correspondentes a luz visível e infravermelho.

No caso do satélite Landsat 5 do programa LANDSAT, 7 sensores provêm informações em diferentes comprimentos de onda, que permitem chamá-lo mapeador temático, sendo possível utilizá-lo para inferir informações de uso do solo ou para identificação de alvos em imagens terrestres. Devido as suas propriedades físico-química, o material ou alvo sobre a superfície da terra responde diferentemente nos diferentes comprimentos de onda. A combinação das respostas então pode ser usada para se inferir informações sobre o alvo, por exemplo, inferir se uma plantação está ou não com algum tipo de doença.

A aquisição das amostras é feita iterativamente, como nos vários sistemas de processamento de imagens ou de geoprocessamento existentes. O processo de aquisição das amostras é supervisionado e depende do conhecimento do usuário de Sensoriamento Remoto. Ele consiste em posicionar um cursor sobre a área de interesse, da qual se tem certeza da verdade terrestre, e atribuí-lo a uma das possíveis classes. Este procedimento é seguido normalmente em se tratando de aplicações reais. Neste trabalho, tomou-se apenas 4 alvos distintos para serem identificados dentro das imagens, ou seja, água, vegetação, área residencial e solo exposto. O processo de aquisição consiste em colocar o cursor dentro da região de interesse e assim grupos de elementos das imagens (pixels) são extraídos das imagens correspondentes às bandas (faixas espectrais em uso, no caso 3 imagens correspondentes às bandas 2, 3 e 4).

Cada amostra deve apresentar a propriedade de homogeneidade, podendo ser medida pelo desvio padrão da amostra em estudo. Quanto menor o desvio padrão, melhor é a amostra na representação da classe.

Podemos ver que este ainda é um problema subjetivo, já que a qualidade da amostra depende do que se estipulou como correspondente à verdade terrestre adquirida.

A figura 8.1 ilustra o processo de aquisição para uma única banda. Um aspecto importante no processo de aquisição de amostras está relacionado com a escolha do tamanho da janela que será utilizada. O tamanho da janela depende do quanto esta é representativa da classe em questão.

Nos classificadores estatísticos em geral as características utilizadas como parâmetros de classificação são medidas estatísticas das amostras. Assim, a dimensão da janela pode ser qualquer na extração de uma amostra. No caso de redes neurais, podemos utilizar medidas

estatísticas das amostras ou os próprios elementos da imagem (pixels). Se as medidas estatísticas forem preferidas, as janelas podem ter dimensões quaisquer, mas se os próprios pixels forem utilizados, então se faz necessário utilizar um tamanho máximo de janela que pode ser utilizada.

No caso de utilizar os próprios pixels da janela como dados de entrada para a rede neural, estes podem ser apresentados à rede em forma bidimensional ou unidimensional. A forma mais comum é unidimensional, quando então os dados bidimensionais devem ser convertidos para um formato unidimensional. A figura 8.1 ilustra o processo de conversão. Observa-se que isso traz uma complicação para a arquitetura da rede a ser utilizada, pois a camada de entrada se torna muito grande, dependendo da dimensão de janela utilizada. No trabalho, as janelas podem ter dimensão quadrática de 3, 5, 7 ou 9. Isso pode resultar em camadas de entrada da rede com 9, 25, 47 ou 81 neurônios por banda. Como três bandas são utilizadas, pode-se chegar a uma camada de entrada com dimensão de até 3 x 81 = 243 neurônios.

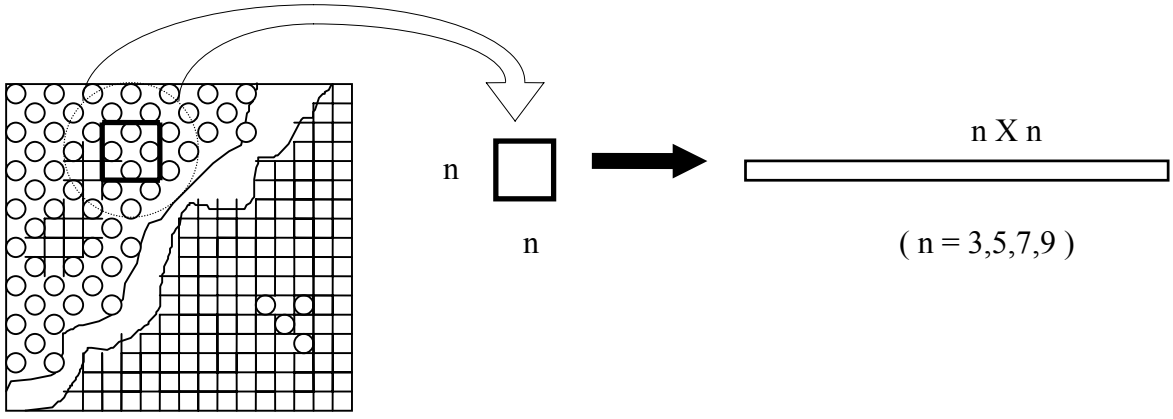


FIGURA 8.1 - Processo de aquisição de amostras para classificação em uma imagem.

A figura 8.2 ilustra o processo de aquisição para o caso de três bandas.

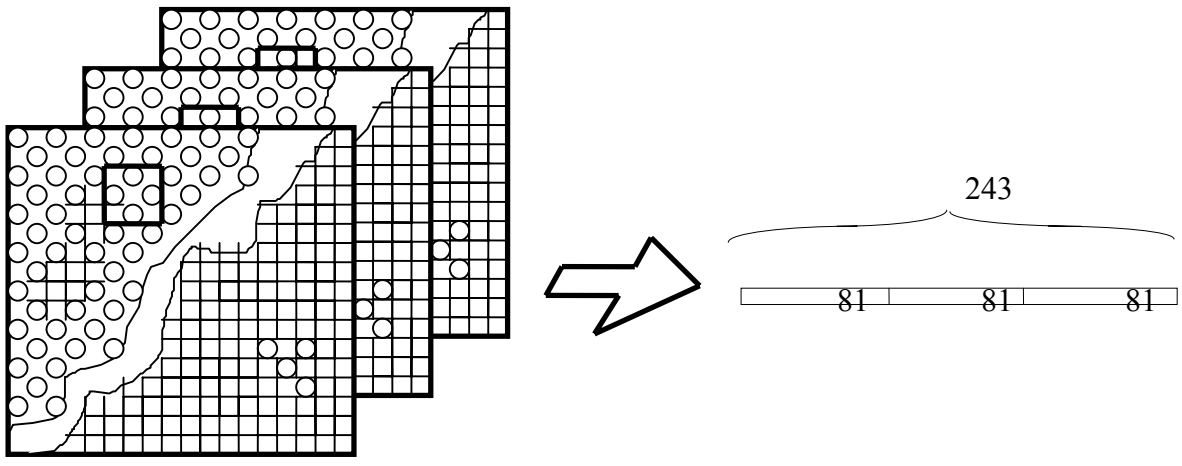
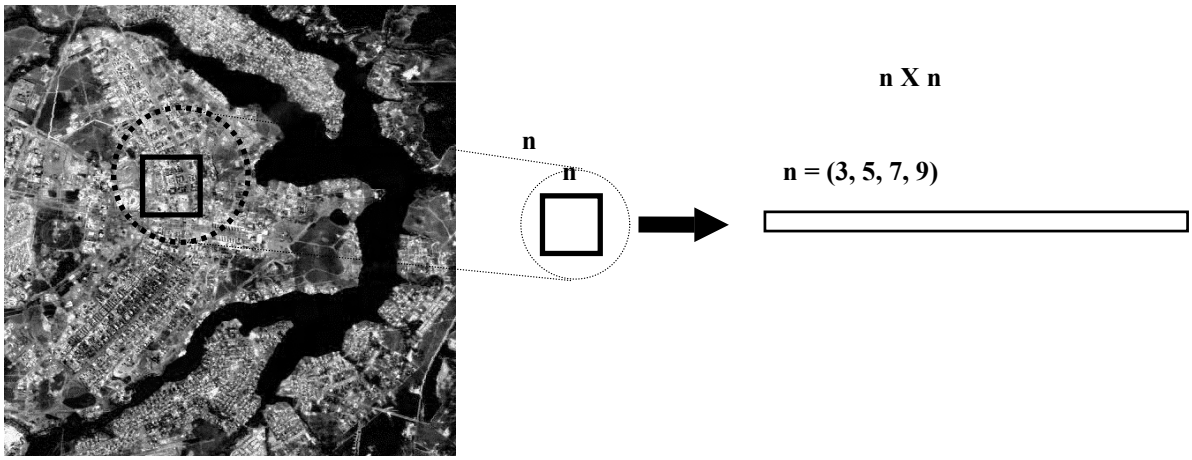
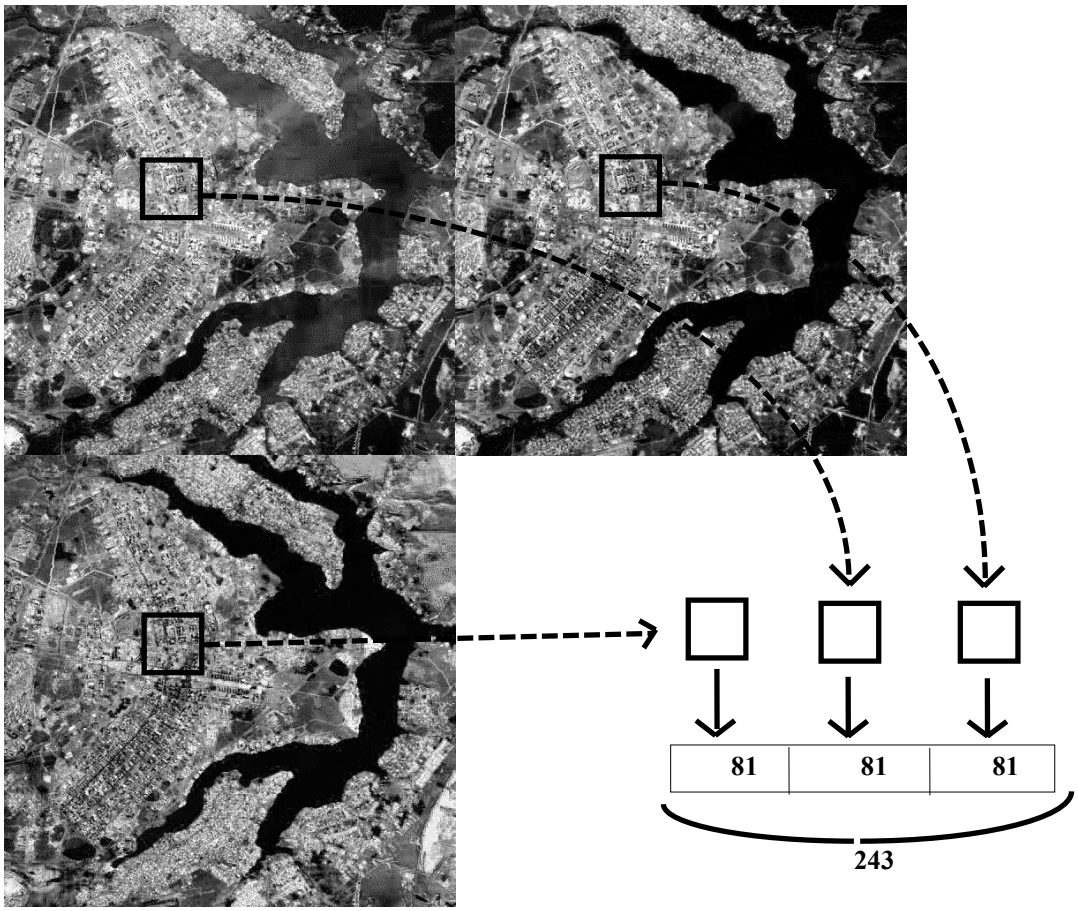


FIGURA 8.2 - Aquisição de amostras em três bandas diferentes, utilizando janelas 9 x 9, o que resulta em vetores de entrada de 243 neurônios.

A figura 8.3 ilustra os processos das figuras 8.1 e 8.2 utilizando um conjunto de 3 bandas, correspondentes a uma pequena área cidade de Brasília, DF, nas bandas 2,3 e 4.



a)



b)

FIGURA 8.3 - Exemplo de aquisição de amostras e imagens.

A aquisição das amostras foi efetuada por uma rotina (adrdsamp.m), que permite a escolha entre tamanhos de janela fixos, variáveis com iteração ou variáveis em função de uma medida de homogeneidade. O algoritmo também permite a escolha sobre os tipos de dados que podem ser fornecidos durante o treinamento: os próprios pixels ou medidas estatísticas.

No trabalho utilizou-se apenas a abordagem por tamanho fixo, com tamanho de janela 3 x 3, resultando então em uma camada de entrada para rede de 9 pixels. Outro teste também foi realizado utilizando medidas estatísticas das amostras para representar as propriedades da classe. Foram utilizadas então a variância e a média. A variância usada como medida de homogeneidade e a média da amostra garantindo a identidade da classe.

Para o exemplo relatado foram extraídas 50 amostras por classe, totalizando 200 amostras ao todo. Estas amostras compunham os dados de entrada. Como o algoritmo de rede neural utiliza aprendizado supervisionado, foi necessário também criar o conjunto de padrões desejados que seria apresentado na saída com a apresentação do conjunto de entrada. Tal conjunto foi formado por vetores binários representando as classes desejadas. Como apenas 4 classes foram escolhidas, os vetores de saída desejadas tinham dimensão 4. Portanto, a camada de saída da rede tem 4 neurônios.

As classes escolhidas para classificação são: água, vegetação, área residencial e solo exposto.

Etapa 9. Implementação dos algoritmos.

O trabalho necessita do uso de um algoritmo de aprendizagem supervisionada. Neste trabalho optou-se pelo uso de perceptron de multicamadas utilizando-se o algoritmo "backpropagation" para o treinamento.

A rede neural utilizada consistiu de 4 camadas: camada de entrada, 2 camadas escondidas e camada de saída. A camada de entrada tem entre 6 a 243 neurônios, dependendo do tamanho das amostras utilizadas e se os dados usados para treinamento são os próprios pixels ou medidas estatísticas extraídas das amostras. Especificamente no experimento conduzido, utilizou-se janelas de tamanho 3 x 3, o que equivale a 9 pixels por amostra em cada imagem, totalizando 27 pixels ordenados em um vetor unidimensional. A camada de saída tem o número de neurônios igual ao número de classes desejadas. O experimento foi realizado para a classificação de 4 classes distintas, como relatado na seção relativa a etapa 8.

O número de neurônios nas camadas internas foram conseguidos através de tentativa erro. Os neurônios nas camadas internas têm como função de ativação a função sigmóide (figura 3.2). Na camada de saída usou-se a função linear. Assim, os neurônios na saída, durante a ativação da rede, apresentam valores não binários. Para recuperar os vetores binários utiliza-se uma rede do tipo MAXNET, que iterativamente busca o neurônio que apresenta maior ativação. Este processo é feito se a soma das ativações superar 0.4, ou seja, se os neurônios apresentarem pelo menos um décimo da ativação correspondente ao nível 1, escolhido para representar o neurônio correspondente a cada classe.

O algoritmo de rede neural utilizando o treinamento por "backpropagation" foi implementado em ambiente MATLAB como a seguir. Entretanto, durante o experimento de classificação de imagens de satélite, utilizou-se as implementações existentes no próprio ambiente MATLAB, como está no apêndice A, onde se encontra todos os arquivos em formato "scripts" utilizados no experimento.

- Algoritmo de Perceptron em Multicamadas Utilizando o Treinamento por Backpropagation:

```
% Algoritmo da Rede Perceptron em Multicamadas utilizando Treinamento por
Backpropagation
% Recebe Parametros de Entrada e Devolve Parametros Treinados.
%       04.22.96
%       Jose Demisio Simoes da Silva
%
% Descrição dos parâmetros:
%     Parâmetros de Saídas:
%       wih_initial             - Pesos iniciais entre a camada de entrada e a camada escondida
1
%       biasih_initial          - Limiares iniciais dos neurônios da camada escondida 1
%       whh_initial             - Pesos iniciais entre camadas escondidas 1 e 2.
%       biashh                  - Limiares iniciais dos neurônios da camada escondida 2
%       who_initial             - Pesos iniciais entre camada escondida 2 e camada de saída
%       biasho_initial          - Limiares iniciais dos neurônios da camada de saída
%       wih                     - Pesos finais entre a camada de entrada e a camada escondida 1
%       whh                     - Pesos finais entre as camadas escondidas 1 e 2
%       who                     - Pesos finais entre a camada escondida 2 e a camada de saída
%       thresholdih             - Limiares finais dos neurônios da camada escondida 1
%       thresholdhh             - Limiares finais dos neurônios da camada escondida 2
%       thresholdho             - Limiares finais dos neurônios da camada de saída
%       epoch                   - Número final de iterações
%       erroacumulado                   - Evolução do erro de treinamento
%
%     Parâmetros de Entrada:
%       n_de_no         s_input         - Número de entradas
```

```
%       functypeih          - Função de ativação da camada escondida 1
%       n_de_nos_hidden1    - Número de neurônios da camada escondida 1
%       unctypehh           - Função de ativação da camada escondida 2
%       n_de_nos_hidden2    - Número de neurônios da camada escondida 2
%       functypeho          - Função de ativação da camada de saída
%       n_de_nos_output             - Número de neurônios da camada de saída
%       i                   - Vetor de entradas
%       t                   - Vetor de saídas desejadas
%       alfa                - Constante de momentum
%       eta                 - Taxa de aprendizagem
%       n_max_de_epochs     - Número máximo de iterações
%       err_desejado        - Erro mínimo desejado
%

function
[wih_initial,biasih_initial,whh_initial,biashh,who_initial,biasho_initial,wih,whh,who,threshol
dih,
thresholdhh, thresholdho,epoch,erroacumulado] =
bp_all(n_de_nos_input,functypeih,n_de_nos_hidden1,
functypehh,n_de_nos_hidden2,functypeho,
n_de_nos_output,i,t,alfa,eta,n_max_de_epochs,err_desejado)

% Computa Tamanho dos Vetores de Entrada:
[n_de_padroes_entrada n_input]=size(i);
[n_de_padroes_saida n_output]=size(t);

% Inicializa Valores de Biases (polarizações):
thresholdih=ones(1,n_de_nos_hidden1)*rand(1,1);

% Normaliza Biases:
thresholdih=thresholdih-max(max(thresholdih))/2;
biasih_initial=thresholdih;

thresholdhh=ones(1,n_de_nos_hidden2)*rand(1,1);
thresholdhh=thresholdhh-max(max(thresholdhh))/2;
biashh_initial=thresholdhh;

thresholdho=ones(1,n_de_nos_output)*rand(1,1);
thresholdho=thresholdho-max(max(thresholdho))/2;
biasho_initial=thresholdho;

% Inicializa Pesos da Camada de Entrada e das Camadas Escondidas:

        % Camada de Entrada para a Camada Escondida 1
wih = rand(n_de_nos_input,n_de_nos_hidden1);
wih=wih-max(max(wih))/2;
wih_initial=wih;
wihant=zeros(n_de_nos_input,n_de_nos_hidden1);
wihant=wih;
        % Camada Escondida 1 para Camada Escondida 2
```

```
whh = rand(n_de_nos_hidden1,n_de_nos_hidden2);
whh=whh-max(max(wih))/2;
whh_initial=whh;
whhant=zeros(n_de_nos_hidden1,n_de_nos_hidden2);

% Inicializa Pesos da Camada Escondida para a Camada de Saida:
who = rand(n_de_nos_hidden2, n_de_nos_output);
who=who-max(max(who))/2;
who_initial=who;
whoant = zeros(n_de_nos_hidden2, n_de_nos_output);
whoant=who;

% Seta Variaveis de Controle de Treinamento:
% Para o aprendizado:
para_learning=0;

% Conta o Numero de Tentativas (Iteracoes):
epoch=0;

% Conta Periodo para Apresentacao de Resultados Parciais:
period_epoch=1;

% Inicia Treinamento:
while( ~para_learning )

        % Conta as Iteracoes:
        epoch=epoch+1;

        % Apresenta Padrao por Padrao:
        for padrao=1:n_de_padroes_entrada

                in = [-1 i(padrao,:)];
                toih= in *[thresholdih; wih];

                % Chama Funcao de Ativacao :
                oih = actvfunc(toih,functypeih);

                tohh = [-1 oih]*[thresholdhh; whh];
                ohh = actvfunc(tohh,functypehh);

                toho= [-1 ohh]*[thresholdho; who];
                oho = actvfunc(toho,functypeho);

                % Calcula o Erro entre o Padrao Desejado e o Obtido pela Rede:
                e(padrao,:)=t(padrao,:)-oho;

                % Faz a Soma dos Erros Quadrativos Instantanea:
                errodisc(padrao)=sum(sum(e(padrao,:).*e(padrao,:)))/2;

                % Calcula Gradiente:
```

```
                derivsigho=deriv(oho,toho,functypeho);
                gradho = e(padrao,:).*derivsigho;
                derivsighh=deriv(ohh,tohh,functypehh);
                gradhh = derivsighh*sum(gradho*who');
                derivsigih=deriv(oih,toih,functypeih);
                gradih = derivsigih*sum(gradho*who');

                % Acha o Acrescimo para os Pesos:
                deltaih = eta*i(padrao,:)'*gradih;
                deltahh = eta*oih'*gradhh;
                deltaho = eta*ohh'*gradho;

                % Atualiza Pesos:
                wih=wih+alfa*(wih-wihant)+ deltaih;
                whh=whh+alfa*(whh-whhant)+ deltahh;
                who=who+alfa*(who-whoant)+ deltaho;
                if epoch > 1
                        wihant=wih;
                        whhant=whh;
                        whoant=who;
                end

                % Atualiza Biases:
                thresholdih=thresholdih-eta*gradih;
                thresholdhh=thresholdhh-eta*gradhh;
                thresholdho=thresholdho-eta*gradho;
        end

        % Acumula o Erro Medio por Iteracao (epoch)
        erroepoch=sum(errodisc)/n_de_padroes_entrada;
        errodisc=0;

        % Apresenta Resultados Parciais
        if period_epoch == 5
                fprintf('epochs %d , erro %f\n',epoch,erroepoch)
                period_epoch=0;
        end

        % Verifica a Condicao de Parada do Treinamento
        if erroepoch < err_desejado | epoch > n_max_de_epochs
                para_learning = 1;
        end

        % Acumula Erro Final
        erroacumulado(epoch)=erroepoch;
        period_epoch=period_epoch+1;
end

% Imprime a Evolucao do Erro.
plot (erroacumulado)
```

O algoritmo acima foi implementado com base em leituras do capítulo 6 de Haykin 1994. O objetivo da implementação foi desenvolver o algoritmo de perceptron multicamada em ambiente MATLAB, para adquirir proficiência com a implementação de redes neurais artificiais e para familiarização com a linguagem de programação do ambiente MATLAB.

Assim sendo, a implementação anterior não apresenta adendos ao algoritmo para torná-lo mais rápido, como acontece com várias implementações existentes do mesmo algoritmo, inclusive aquelas utilizadas nos experimentos de classificação de imagens abordados neste trabalho. Dentre as abordagens que tentam acelerar o desempenho de redes perceptron em multicamadas está o algoritmo "Cascade-Correlation" comentado na seção referente à etapa 3 do estágio. Outra abordagem para aceleração envolve o uso de lógica nebulosa ("fuzzy logic") para controle da taxa de aprendizagem, como em Choi et al. 1992.

Etapa 10. Testes dos algoritmos de redes neurais em tarefas bem conhecidas de análise de imagens.

Como mencionado na seção anterior os experimentos com classificação de imagens foram desenvolvidos utilizando as implementações do perceptron em multicamadas disponíveis no "toolbox" do ambiente MATLAB. No apêndice A tem-se a descrição das várias rotinas implementadas como arquivos "scripts", que durante a execução invocam as funções que implementam a rede neural utilizada.

Dois arquivos "scripts" básicos são utilizados: um para treinamento da rede (treiclas.m) e outro para ativação da rede (classify.m). O arquivo treiclas.m executa a fase de treinamento da rede neural. O "script" foi projetado para permitir que o usuário construa o conjunto de treinamento ou utilize conjuntos definidos anteriormente.

Para teste de aprendizagem foi feito um outro conjunto teste contendo o mesmo número de amostras por classe. Este conjunto foi utilizado para medir a generalização da rede após treinamento, o que será comentado na seção referente à medidas de desempenho.

Quatro conjuntos (samples2, samples3, samples4 e samples5) de dados foram extraídos das imagens. Sendo dois (samples2 e samples3) com janelas de tamanho fixo e dois (samples4 e samples5) com janelas de tamanhos variáveis em função da variância dentro da

janela. Neste caso a janela escolhida é aquela que apresenta a menor variância, interpretada no contexto como sendo a mais suave, do ponto de vista de distribuição de radiometrias. As amostras do arquivo samples2 foram utilizadas para treinar a rede com abordagens estatísticas e não estatísticas. A abordagem não estatística utiliza os próprios níveis de cinza das amostras como dados de entrada para rede, enquanto que a abordagem estatística utiliza as medidas de média e variância das amostras.

Os dados obtidos com janelas de tamanho variável foram treinados apenas na abordagem não estatística.

A seguir são exibidos os resultados do treinamento obtido através das curvas de evolução do erro de treinamento.

A tabela 10.1 exibe um resumo da arquitetura da rede e do tipo de abordagem utilizada no treinamento.

TABELA 10.1 - Resumo da arquitetura da rede neural utilizada com a abordagem utilizada no treinamento.

Experimento	Abordagem	Número de nós na camada interna
exp1	Não estatística	20
exp2	Estatística	20
exp3	Não estatística	20

A tabela 10.2 mostra um quadro das configurações e resultados de treinamento obtidos, salientando as bases de dados utilizadas.

TABELA 10.2 - Resultado do treinamento para os três experimentos conduzidos.

Arquivo com resultados de treinamento	Conjunto de treinamento utilizado	SSE estabelecido como alvo	SSE alcançado em treinamento	Número de iterações no treinamento
exp1	samples2	40	39.9840	475
exp2	samples2	40	39.9792	435
exp3	samples4	40	39.82	135

As figuras 10.1, 10.2 e 10.3 mostram a curvas de erro obtidas durante o treinamento.

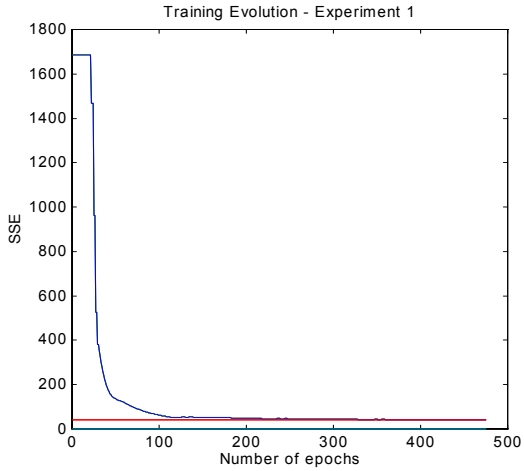


FIGURA 10.1 - Curva de treinamento do experimento 1. A linha azul mostra como o erro de treinamento. A linha vermelha é o erro alvo

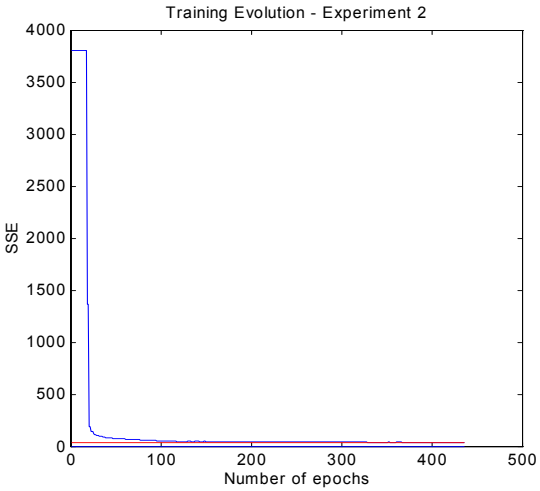


FIGURA 10.2 - Curva de treinamento do experimento 2. A linha azul mostra como o erro de treinamento. A linha vermelha é o erro alvo

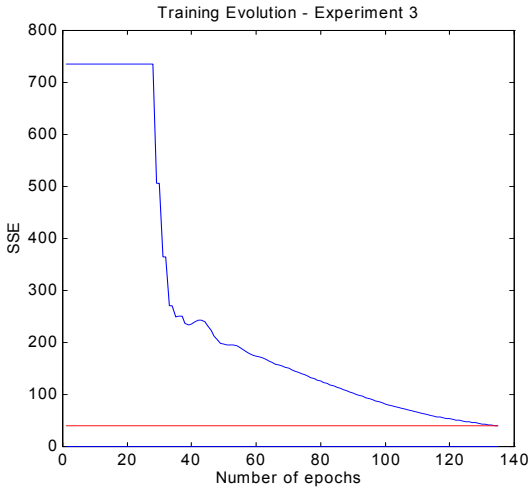


FIGURA 10.3 - Curva de treinamento do experimento 3. A linha azul mostra como o erro de treinamento. A linha vermelha é o erro alvo (SSE=40).

Após o treinamento foi realizado o teste de generalização da rede treinada. Este teste consiste em apresentar dados não utilizados no treinamento e analisar o comportamento da rede. A tabela 10.3 mostra um resumo dos testes de generalização realizados.

TABELA 10.3 - Resultados da generalização das redes treinadas.

Experimento	SSE generalizacao (samples3)	SSE generalização (samples5)
exp1	104	
exp2	110	
exp3		66

Os dados na tabela 10.3 estão organizados segundo a utilização dos arquivos de dados usados no treinamento. A coluna referente ao arquivo samples3, significa que o treinamento foi realizado com as amostras do arquivo samples2 e a coluna referente ao arquivo samples5, significa que o treinamento foi realizado com as amostras do arquivo samples4.

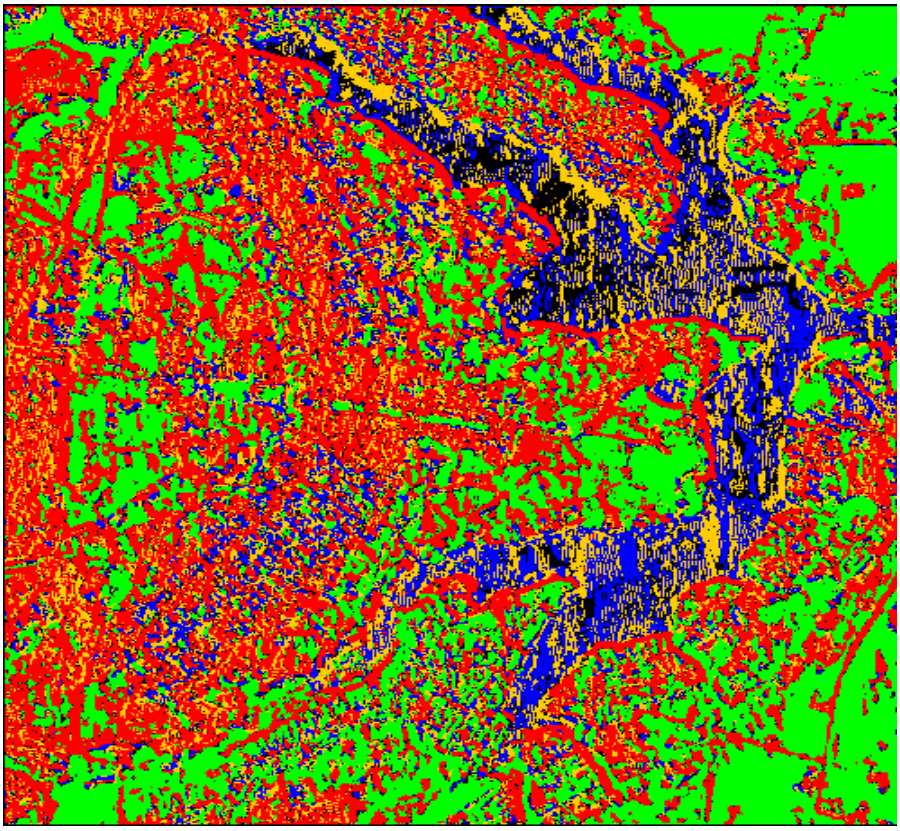


FIGURA 10.4 - Resultado da classificação obtida com a rede neural treinada no experimento 1 (abordagem não estatística, com janela de 3x3).

As figuras 10.4, 10.5 e 10.6 exibem o resultado da classificação realizada pelas redes neurais, como resultado da apresentação das imagens originais filtradas por filtro passa baixas para a eliminação de ruídos. A figura 10.4 corresponde a classificação com os dados do experimento 1 (exp1). A figura 10.5 exibe a classificação com os dados do experimento 2 (exp2). A figura 10.6 mostra o resultado da classificação, usando-se os dados do experimento 3 (exp3).

Na figura 10.4 as classes de interesse estão atribuídas da seguinte forma: Água - cor azul; Vegetação - cor verde; Área Construída - cor laranja; e Solo Exposto - Vermelho.

Na figura 10.5 as classes são assim distribuídas: Água - cor azul; Vegetação - cor verde; Área Construída - cor ciano; e Solo Exposto - Vermelho.

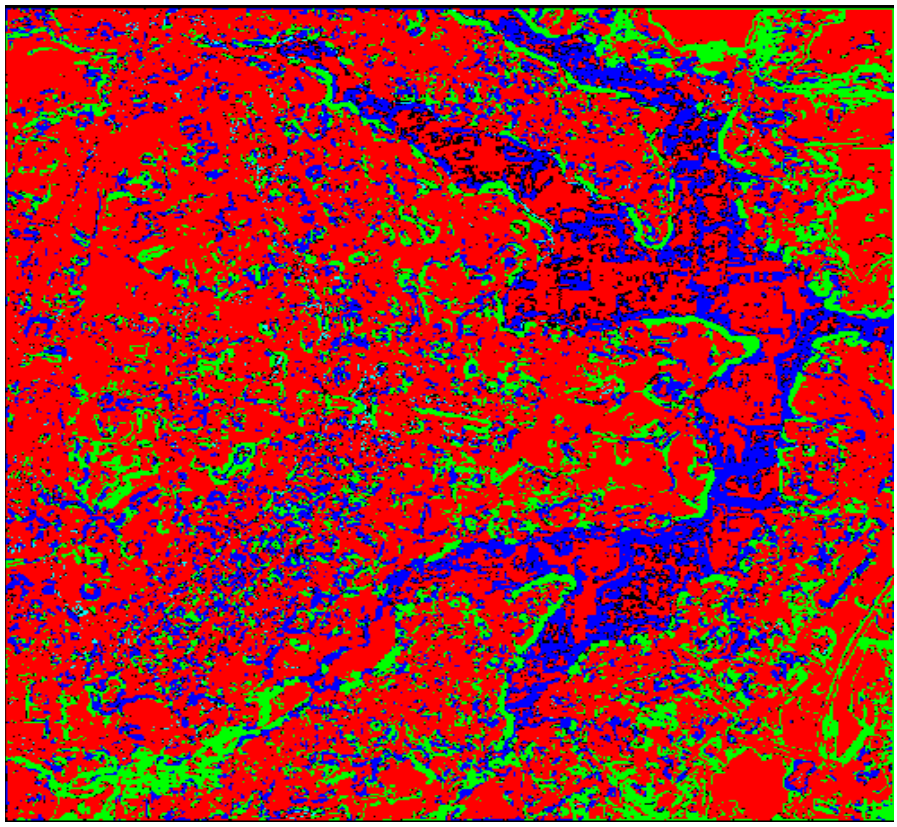


FIGURA 10.5 - Resultado da classificação obtida com a rede neural treinada no experimento 2 (abordagem estatística, com janela de 3x3).

Na figura 10.6 as classes têm as seguintes cores associadas: Água - cor azul; Vegetação - cor verde; Área Construída - cor laranja; e Solo Exposto - Vermelho.

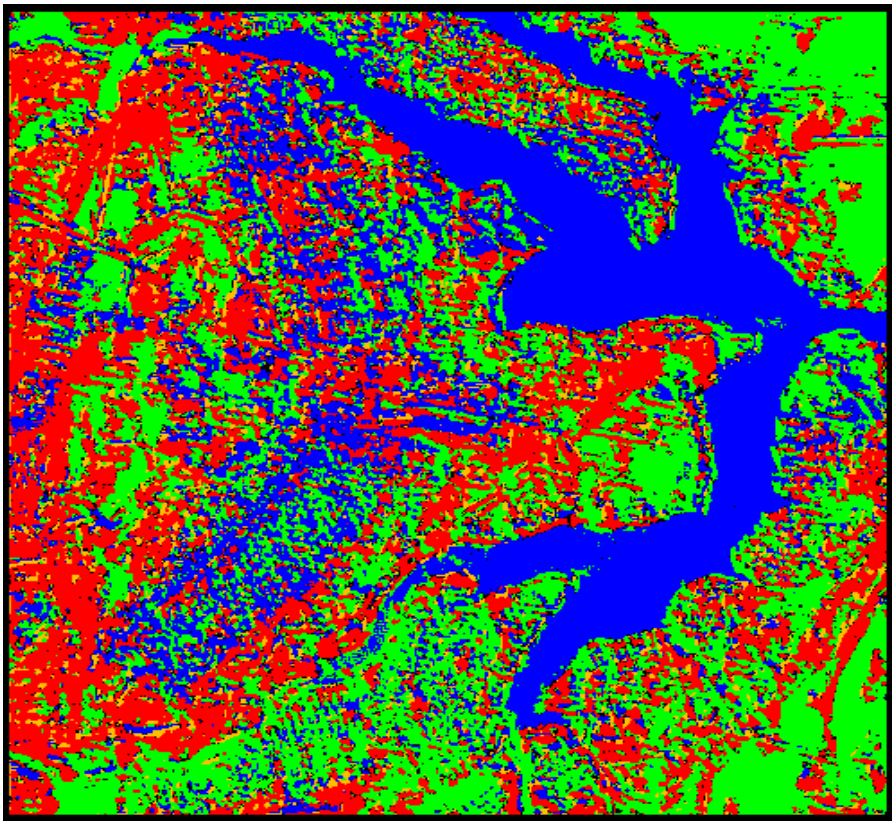


FIGURA 10.6 - Resultado da classificação obtida com a rede neural treinada no experimento 3 (abordagem não estatística, com janela de 9x9; Amostras obtidas com janela adaptativa).

As figuras 10.4 e 10.5 exibem uma grande confusão entre as classes Área Construída e Solo Exposto, como foi constatado no teste de generalização e através da matriz de confusão (ver tabelas 11.1, 11.2 e 11.3). Este problema pode ser solucionado se as amostras forem melhor selecionadas durante o processo de aquisição de amostras e definição de classes.

A figura 10.6 exibe um melhor resultado, devido ao fato da janela (campo de visada) do algoritmo classificador ser de grande dimensão (9x9), o que implica em um maior número de pixels vizinhos interagindo na classificação de um pixel.

Etapa 11. Especificação de medidas de desempenho com aplicação às implementações.

O desempenho de um classificador é medido em função do erro de classificação. Neste trabalho o desempenho da rede neural como classificadora foi medido através do erro de generalização. A generalização consiste na avaliação da rede neural treinada sobre dados não utilizados no treinamento. Neste sentido, como comentado na seção referente a etapa 10 dos 'Testes dos algoritmos de redes neurais em tarefas bem conhecidas de análise de imagens', utilizou-se conjuntos de dados das mesmas classes (samples3 e samples5) tomadas como alvos para teste da generalização da rede.

O teste consistiu em apresentar as amostras à rede e medir o erro em relação ao esperado. A tabela 10.4 exibe os resultados do erro de classificação.

Como podemos observar o erro de generalização para o caso do treinamento com amostras de tamanho fixo (3x3), o desempenho de generalização da rede foi baixo, se compararmos os erros obtidos com o erro estabelecido como alvo no treinamento.

Entretanto, para o caso de amostras obtidas com janelas adaptativas, o erro de generalização aproximou-se do erro alvo.

Para se ter uma idéia do que causou os baixos desempenhos, podemos observar as matrizes de confusão para os dois casos de testes de generalização. As tabelas abaixo mostram as matrizes com a confusão entre as amostras.

Todos os dados nas tabelas 11.1 a 11.3 são expressos em porcentagens, em relação ao total de amostras por classes. Para cada classes foram tomadas 50 amostras. Nas três tabelas verifica-se que as classes Área Construída e Solo Exposto apresentaram maior confusão. Entretanto, o classificador que usou dados obtidos com janela adaptativa apresentou melhor desempenho.

TABELA 11.1 - Matriz de confusão para o classificador do experimento 1.

	Água	Vegetação	Área Construída	Solo Exposto
Água	92	8	0	0
Vegetação	10	86	0	4
Área Construída	0	0	66	34
Solo Exposto	0	10	38	52

TABELA 11.2 - Matriz de confusão para o classificador do experimento 2.

	Água	Vegetação	Área Construída	Solo Exposto
Água	90	10	0	0
Vegetação	8	84	0	10
Área Construída	0	0	66	34
Solo Exposto	2	12	36	50

TABELA 11.3 - Matriz de confusão para o classificador do experimento 3.

	Água	Vegetação	Área Construída	Solo Exposto
Água	100	0	0	0
Vegetação	14	76	0	10
Área Construída	0	0	96	4
Solo Exposto	0	4	34	62

Conclusão

Este trabalho teve como objetivo a familiarização com as redes neurais artificiais, principalmente voltadas para aplicações em Visão Computacional e Processamento de Imagens. Apesar dos algoritmos de redes neurais serem cada vez mais utilizados em diversas áreas de aplicação, ainda há muito trabalho a ser realizado nas duas áreas mencionadas anteriormente, devido à necessidade de adaptação dos algoritmos aos problemas destas áreas.

A adaptabilidade dos modelos de redes neurais envolve estudos e pesquisas do paradigma de redes neurais e das áreas de aplicação. Os modelos de redes neurais disponíveis na literatura precisam ser construídos de forma a modelar os aspectos envolvidos nas tarefas, o que implica na necessidade de pesquisas. Além disso, existem muitos aspectos inerentes às redes neurais que necessitam de pesquisas, para tornar os algoritmos de redes neurais artificiais existentes e os que surgirem, mais eficientes no tocante a arquitetura utilizada e a aprendizagem conseguida.

Durante o estágio foram feitos estudos e implementações de algoritmos de redes neurais, bem como o acompanhamento da sistemática do uso de redes neurais em tarefas de Processamento de Sinais e Imagens desenvolvidos no âmbito do Departamento onde o estágio foi desenvolvido.

Os experimentos propostos realizados consistiram em classificar imagens de satélite. O objetivo principal foi aplicar a metodologia de trabalho de aplicação de redes neurais em

tarefas desta natureza, ou seja, como adaptar uma rede neural para executar a tarefa de classificação supervisionada de imagens de satélite.

Portanto, os resultados encontrados, como visto nas figuras 10.4, 10.5 e 10.6, não são resultados ditos precisos devido a dois fatores: primeiro, a imagem utilizada é uma imagem predominantemente de zona urbana, com uma alta densidade construída. Isso introduz um grau de complexidade na obtenção das amostras para as classes estabelecidas como alvo, por a área apresentar um uso de solo muito diversificado. Em segundo, como o problema de sensoriamento remoto abordado não era o objetivo principal do trabalho, a seleção das amostras não envolveu passos criteriosos para melhorar o desempenho do classificador.

Por outro lado o trabalho mostrou o poder das ferramentas de redes neurais, apesar da complexidade existente quanto à necessidade de treinamento. Em geral, a maioria do tempo utilizado nos experimentos foi dedicada ao treinamento das redes neurais.

A especificação da rede neural é uma tarefa iterativa que envolve a especificação de várias tentativas para se atingir uma topologia que resolva o problema. A resolução do problema é inferida se há o treinamento da rede, ou seja, se a rede convergir para o erro alvo. Entretanto, na literatura existem muitos trabalhos que sugerem o uso de um determinado número de neurônios na camada interna, em função do número de padrões no conjunto de treinamento. Essa abordagem pode não funcionar sempre em função da natureza dos dados no conjunto de treinamento.

Observou-se que muitos problemas podem ser resolvidos utilizando-se as redes neurais. Entretanto, a maioria dos problemas resolvidos por redes neurais relatados na literatura, já foram resolvidos por outras abordagens também. Em geral, a abordagem com redes neurais tentam simplificar os problemas no sentido de que as restrições inerentes a maioria dos problemas, é substituída pela tarefa menos difícil de treinar a rede neural, apesar de se tomar bastante tempo de treinamento dependendo do algoritmo de aprendizagem usado.

Há a necessidade entretanto, de se pesquisar novos modelos ou procedimentos de pré-processamento, que sejam mais adequados para determinadas áreas de aplicação.

O estágio permitiu a familiarização com ferramentas de simulação de redes neurais adequadas e voltadas para a modelagem do problema em mãos.

Referências Bibliográficas

Accoto, P.; Arena, F.; Storti-Gajani, G. Image Texture analysis with neural networks. World Congress on Neural Networks, vol III, pp. III-776 - III-779, July 11-15, 1993.

Aloimonos, Y. Visual Shape Computation. Proceedings of the IEEE, 76(8):899-916, August 1988.

Aloimonos, Y. Active Perception. Editor Yiannis Aloimonos, Lawrence Erlbaum Associates, New Jersey, 1993.

Bajcsy, R. Active Perception. Proceedings of the IEEE, 76(8):996-1005, August 1988.

Bajcsy, R.; Campos, M. Active and Exploratory Perception. CVGIP: Image Understanding, 56(1):31-40, July 1992.

Bebis, G.N.; Papadourakis, G.M. Object Recognition using Invariant Object Boundary Representations and Neural Network Models. Pattern Recognition, 25(1):25-44, 1992.

Ballard, D.H.; Brown, C.M. Principles of Animate Vision. CVGIP: Image Understanding, 56(1):3-21, July 1992.

Beale, R.; Jackson, T. Neural Computing: An Introduction. IOP Publishing Ltd. 1990.

Bertero, M.; Poggio, T.; Torre, V. III-Posed Problems in Early Vision. Proceedings of the IEEE, 76(8): 869-889, August 1988.

Bhuiyan, S.; Matsuo, H.; Iwata, A.; Fujimoto, H.; Sato, M. An improved Neural Network based Edge Detection Method. Proceedings of the International Conference on Neural Information Processing, Seoul, Korea, 1:620-625, October 17-20, 1994.

Boden, M.A. Computer models of mind: Computational approaches in theoretical psychology. Cambridge University Press, 1991, p. 1 - 87.

Canny, J. A Computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8(6):679-698, 1986.

Choi, J.J.; Araabshashi, P.; Marks, R.J.; Caudell, T.P. Fuzzy Parameter Adaptation in Neural Systems. International Joint Conference on Neural Networks, Vol.1, pp. 232-238, Baltimore, MD, 1992.

Cruz, Jesús M.; Pajares, G.; Aranda, J. A Neural Network Model in Stereovision Matching. Neural Networks, 8(5):805-813, 1995.

Duda, R.O.; Hart, P.E. Pattern Classification and Scene Analysis. Stanford Research Institute, Menlo Park, California, John Wiley & Sons, 1973.

Eberhart, R.C.; Dobbins, R.W. Neural Networks PC Tools: A Practical Guide. The John Hopkins University Applied Physics Laboratory, Laurel, Maryland, 1990.

Fahlman, S.E. and Lebiere, C. The Cascade-Correlation Learning Architecture. Report CMU-CS-90-100, Carnegie Mellon University, Pittsburgh, February 1990

Feldman, J.A. Connectionist Models and Parallelism in High Level Vision. Computer Vision, Graphics, and Image Processing, 31:178-200, 1985.

Finkel. L.H.; Sajda, P. Constructing Visual Perception. American Scientist, 82:224-237, May - June 1994.

Fukushima, K. A Neural Network for Visual Pattern Recognition. Computer, 65-75, March 1988.

Fukushima, K.; Miyake, S. Neocognitron: A New Algorithm for Pattern Recognition Tolerant of Deformations and Shifts in Position. Pattern Recognition, 15(6):455-469, 1982.

Haykin, Simon. Neural Networks: A Comprehensive Foundation. Macmillan, New York, 1994

Haykin, Simon. Neural Networks Expand SP's Horizons. IEEE Signal Processing Magazine, 24-49, March 1996.

Haring, S; Viergever, M.A.; Kok, J.N. Kohonen networks for multiscale image segmentation. Image and Vision Computing, 12(6):339-344, July/August 1994.

Hines, W. Matlab Supplement of Fuzzy and Neural Approaches in Engineering. (Under preparation) John Wiley & Sons, 1997.

Hecht-Nielsen B. Neurocomputing. Addison-Wesly, 1990.

Hertz, J.; Krogh, A.; Palmer, R.G. Introduction to the Theory of Neural Computing. Santa Fe Institute. Addison-Wesly, 1991.

Horn, B.K.P. Robot Vision. New York, NY, McGraw-Hill, 1986.

Jolion, J.M. Computer Vision Methodologies. CVGIP: Image Understanding, 59(1):53-71, January 1994.

Joshi, A.; Lee, Chia-Hoang, Stereo Correspondence and Missing Points. World Congress on Neural Networks, vol II, pp. III-731 - III-734, July 11-15, 1993.

Khanna, T. Foundations of Neural Networks. Addison-Wesley Publishing Company, Inc., 1990.

Khotanzad, A.; Lu, J.H. Object Recognition Using a Neural Network and Invariant Zernike Features. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 200-205, San Diego, CA, June 4-8, 1989.

Koch, C.; Segev, I. Methods in neuronal Modeling: From Synapses to Networks. Mit Press, Cambridge, 1989.

Koenderink, J.J The Structure of Images. Biological Cybernetics, 50:363-370, 1984.

Lee, J.J.; Shim, J.C.; Ho Ha Y. Stereo Correspondence Using the Hopfield Neural Network of a New Energy Function. Pattern Recognition, 27(11):1513-1522, 1994.

Lin, Chin-Leng and Lee, C.S. G. Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems. Prentice Hall, New Jersey, 1996

Lindeberg, T. Scale-Space: A Basis for Early Vision. Report from Computational Vision and Active Perception Laboratory (CVAP). Royal Stockholm University, Institute of Technology, Stockholm, Sweden. 1993.

Lu, Y.; Jain, R.C. Reasoning about Edges in Scale Space. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(4):450-468, April 1992.

Mambelli, E. Ncomp: Application of the Holpfield Neural Network Model to the Stereo Matching Problem. World Congress on Neural Networks, vol II, pp. III-735 - III-739, July 11-15, 1993.

March, R. Computation of stereo disparity using regularization. Pattern Recognition Letters, 8:181-187, October 1988.

Marr, D. Vision. Freeman. San Francisco, CA. 1982.

Mathworks Inc. http://www.mathworks.com. 1997.

Mousavi, M.S.; Schalkoff, R.J. ANN Implementation of Stereo Vision Using A Multi-Layer Feedback Architecture. IEEE Transactions on Systems, Man, and Cybernetics, 24(8):1220-1238, August 1994.

Nasrabadi, N.M.; Choo, C.Y. Hopfield Network for Stereo Vision Correspondence. IEEE Transactions on Neural Networks, 3(1):5-13, January 1992.

Paik, J.K.; Brailean, J.C.; Katsaggelos, A.K. An Edge Detection Algorithm Using Multi-State ADALINES. Pattern Recognition, 25(12):1495-1504, 1992.

Park, D.J.; Park, R.-H.; Lee, S.U.; Choi, J.S. Hierarchical edge detection using the bi-directional information in edge pyramids. Pattern Recognition Letters, 15:65-75, January 1994.

Pavlidis, T. Why progress in machine vision is so slow. Pattern Recognition Letters, 13:221-225, 1992.

Pentland, A.P. Perceptual Organization and the Representation of Natural Form. Artificial Intelligence, 28:293-331, 1986.

Poggio, T. Early Vision: From Computational Structure to Algorithms and Parallel Hardware. Computer Vision, Graphics, and Image Processing, 31:139-155, 1985.

Poggio, T.; Torre, V.; Koch, C. Computational vision and regularization theory. Nature 317:314-319, 1985.

Rao, Rajesh P.N.; Ballard, Dana H. A Class of Stochastic Models for Invariant Recognition, Motion, and Stereo. Submitted to NIPS 96.

Rattarangsi, A.; Chin, R.T. Scale-Based Detection of Corners of Planar Curves. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(4):430-449, April 1992.

Rosenfeld, A. Computer Vision: Basic Principles. Proceedings of the IEEE, 76(8):863-868, August 1988.

Santos, N.M; Oliveira, E.C. A neurophysiological neural network model for vehicle identification. Anais do II Congresso Brasileiro de Redes Neurais, Curitiba, 29/10-01/11/1995, p. 133-137.

Sarkar, S.; Boyer, K.L. Perceptual Organization in Computer Vision: A Review and a Proposal for a Classificatory Structure. IEEE Transactions on Systems, Man, and Cybernetics, 23(2):382-398, March/April 1993.

Schalkoff, R.J. Digital Image Processing and Computer Vision, John Wiley & Sons, New York, 1989.

Schalkoff, R.J. Pattern Recognition: Statistical, Structural and Neural Approaches, John Wiley & Sons, New York, 1992.

Silva, J. D. S. The Cascade-Correlation using the Matlab environment. http://www.lac.inpe.br/~demisio/casc.html. 1997.

Simpson, P.K. Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations. Pergamon Press. San Diego, 1990.

Skoneczny, S.; Foltyniewicz, R. An Efficient Method of Blur Identification by Neural network for Image Restoration Purpose. World Congress on Neural Networks, vol II, pp. III-665 - III-668, July 11-15, 1993.

Somers, D.C.; Nelson, S.B.; Sur, M. An Emergent Model of Orientation Selectivity in Cat Visual Cortical Simple Cells. Journal of Neuroscience, in press, march 2, 1995

Srinivasan, V.; Bhatia, P.; Ong, S.H. Edge Detection Using a Neural Network. Pattern Recognition, 27(12):1653-1662, 1994.

Torre, V.; Poggio, T. On Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(2):147-163, March 1986.

Tsoukalas, L.H.; Uhrig, R.E. Fuzzy and Neural Approaches in Engineering. John Wiley & Sons, 1997.

Uhrig, R.E. Tutotial: Artificial Neural Networks and Potential Applications to Nuclear Power Plants. DOE-EPRI Workshop on Cost-Effective Instrumentation and Control Technologies Upgrades, Nashivlle, Tennessee, March 22-24, 1995.

Wechsler, H. Computational Vision. Academic Press, Boston, 1990, p. 183 - 192.

Wenzel, B.C.; Gimblett, R.H. An Approach to Image Compression and Reconstruction Using Neural Networks. World Congress on Neural Networks, vol IV, pp. IV-71 - IV-74, July 11-15, 1993.

Wehmeier, U.; Dong, D.; Koch, C.; Van Essen, D. Modeling the Mammalian Visual System. Em Methods in neuronal Modeling: From Synapses to Networks. Editores: Christof Koch e Idan Segev. MIT Press, Cambridge, 1989, p. 335-359.

You, S. D.; Ford, G. E. Network Model for Invariant Object Recognition. World Congress on Neural Networks, vol II, pp. III-718 - III-721, July 11-15, 1993.