

João Carlos Figueira Pujol

Orientador: Arnaldo de Albuquerque Araújo

Aplicação de Redes Neurais no Processamento Digital de Imagens

Dissertação apresentada ao Departamento de Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação

Belo Horizonte

Março de 1994

*Meus Agradecimentos ao Professor Arnaldo de Albuquerque
Araújo pelas sugestões e críticas, e aos meus colegas pela ajuda na
revisão do texto.*

ÍNDICE

1	Introdução	1
1.1	Objetivo	1
1.2	Organização do trabalho	3
2	Redes neuronais	4
2.1	Conceitos básicos	4
2.1.1	Neurônio	5
2.1.2	Conexões	8
2.1.3	Funcionamento	10
2.1.4	Determinação dos pesos	10
2.2	<i>Perceptron</i> multicamadas	14
2.2.1	Topologia	14
2.2.2	Algoritmo de treinamento <i>backpropagation</i>	16
2.3	Mapa auto-organizativo	18
2.3.1	Mapa auto-organizativo de Kohonen	19
2.3.2	Mapa para aprendizado competitivo sensível à frequência	22
2.4	Rede de Hopfield	23
2.4.1	Condições de estabilidade Cohen-Grossberg	23
2.4.2	O modelo de Hopfield	23
2.5	Conclusão	27
3	Processamento digital de imagens	28
3.1	Conceitos básicos e terminologia	28
3.2	Compressão	29
3.3	Detecção de bordas	31
3.4	Restauração	32
3.5	Conclusão	35

4	Aplicação de redes neuronais em PDI	36
4.1	Compressão de imagens por <i>perceptron</i> multicamadas	36
4.1.1	Compressão de imagens em níveis de cinza	36
4.1.2	Compressão de imagens coloridas	39
4.2	Quantização vetorial	40
4.3	Detecção de bordas por <i>perceptron</i> multicamadas	41
4.4	Detecção de bordas por rede de Hopfield	43
4.5	Restauração de imagens por rede de Hopfield	45
4.5.1	Representação da imagem	45
4.5.2	Estimativa dos parâmetros do modelo	46
4.5.3	Restauração	49
4.5.4	Algoritmo prático	51
4.6	Conclusão	52
5	Implementação dos algoritmos	57
5.1	Introdução	57
5.2	<i>Perceptron</i> multicamadas para compressão de imagens	58
5.2.1	Implementação	58
5.2.2	Resultados	62
5.3	Mapa auto-organizativo	71
5.3.1	Implementação	71
5.3.2	Resultados	73
5.4	Detecção de bordas por <i>perceptron</i> multicamadas	78
5.4.1	Implementação	78
5.4.2	Resultados	79
5.5	Conclusão	83
6	Conclusão	85
	Referências bibliográficas	87

LISTA DE FIGURAS E TABELAS

FIGURAS

Estados da rede de Hopfield para detecção de borda	44
Função sigmóide típica	7
Imagem degradada por turvamento uniforme e ruído de quantização	54
Imagem original	53
Imagem real colorida em 24 bits e mapa de bordas gerado pela rede neuronal	82
Imagem real em tons de cinza e mapa de bordas gerado pela rede neuronal	79
Imagem restaurada pela rede neuronal	56
Imagem restaurada por filtro inverso	55
Imagem sintética em tons de cinza e mapa de bordas gerado pela rede neuronal	80
Imagens coloridas decodificadas por quantização vetorial	77
Imagens coloridas descomprimidas por <i>perceptron</i> multicamadas	70
Imagens em tons de cinza decodificadas por quantização vetorial	76
Imagens em tons de cinza descomprimidas por <i>perceptron</i> multicamadas	69
Imagens coloridas tratadas em 24 <i>bits</i> utilizadas nos treinamentos	63
Imagens em tons de cinza utilizadas nos treinamentos	64
Imagens coloridas em 24 bits utilizadas nos testes de compressão/descompressão	65
Imagens em tons de cinza utilizadas nos testes de compressão/descompressão	66
Mapa auto-organizativo de Kohonen	21
Modelo do neurônio	7
Padrões de borda e padrões que não representam borda	42
<i>Perceptron</i> de camada única separa um vetor de entrada em duas classes A e B	13
<i>Perceptron</i> multicamadas, uma rede com realimentação para frente	9
Reconhecimento de caracteres pela rede de Hopfield	26
Rede de compressão e rede de descompressão	38
Rede de Hopfield, uma topologia com realimentação total	9
Tipos de regiões que podem ser separadas por <i>perceptrons</i>	15

TABELAS

Resultados da descompressão por <i>peceptron</i> multicamadas e algoritmos convencionais, referentes à Figura 19a	67
Resultados da descompressão por <i>peceptron</i> multicamadas e algoritmos convencionais, referentes à Figura 19b	67
Resultados da descompressão por <i>peceptron</i> multicamadas e algoritmos convencionais, referentes à Figura 18a	68
Resultados da descompressão por <i>peceptron</i> multicamadas e algoritmos convencionais, referentes à Figura 18b	68
Resultados da descompressão por mapa auto-organizativo, referentes à Figura 19a	74
Resultados da descompressão por mapa auto-organizativo, referentes à Figura 19b	74
Resultados da descompressão por mapa auto-organizativo, referentes à Figura 18a	75
Resultados da descompressão por mapa auto-organizativo, referentes à Figura 18b	75

RESUMO

Neste trabalho, é discutida a aplicação de redes neuronais no processamento digital de imagens, mais especificamente, em compressão de imagens, quantização vetorial, restauração de imagens e detecção de bordas. São apresentados os conceitos fundamentais de redes neuronais e algumas das topologias básicas, tendo em vista as aplicações mencionadas. São discutidos os modelos de *perceptron* multicamadas com algoritmo de aprendizado supervisionado *backpropagation*, uma topologia para aprendizado competitivo e a rede Hopfield. São apresentadas as aplicações destes modelos em processamento digital de imagens, bem como a implementação dos algoritmos e os resultados obtidos.

ABSTRACT

In this work applications of neural networks to image processing are presented, more specifically, to image coding, vector quantization, image restoration and edge detection. The fundamentals of neural networks and some topologies are discussed concerning the mentioned applications. The multilayered perceptron model with the backpropagation supervised learning rule, a topology for competitive learning, and the Hopfield network are discussed. The application of these models to digital image processing, the implementation of the algorithms and the results obtained are presented

1

INTRODUÇÃO

1.1 Objetivo

Os computadores digitais convencionais são muito bons em executar tarefas bem definidas, com grande velocidade e precisão muito acima da capacidade humana como, por exemplo, inverter uma matriz. As instruções são executadas seqüencialmente uma de cada vez. Esta abordagem tem sido estendida pela introdução de processamento paralelo [HUSSAIN91].

O cérebro humano, por outro lado, tem excelente performance em tarefas como visão, percepção da comunicação oral e auditiva, além do reconhecimento de complexos padrões espaciais e temporais, inclusive na presença de ruído, de dados distorcidos ou incompletos [WIDROW90].

Curiosamente, os circuitos eletrônicos são várias ordens de grandeza mais rápidos do que os neurônios do cérebro. Contudo, o cérebro humano utiliza paralelismo em larga escala. Milhões e até mesmo bilhões de neurônios são usados simultaneamente para resolver problemas complexos [SIMPSON90].

Definir o conceito da letra **A**, nas suas mais diversas variações, a partir de conceitos de alto nível como tangentes, ângulos, número de buracos, momentos e outros, é uma tarefa difícil e de resultados limitados. No entanto, qualquer pessoa é capaz de reconhecer facilmente a letra **A**, mesmo quando esta está incompleta. Isto porque as pessoas foram naturalmente treinadas desde crianças, a verem e classificarem os diversos tipos de letras nas mais diversas formas e contextos. As pessoas têm formado internamente um conceito do que seja a letra **A**, conceito este que não é facilmente quantitativamente representado.

Técnicas de inteligência artificial se utilizam da manipulação simbólica, da lógica e da base de conhecimento. Os sistemas especialistas são um exemplo deste tipo de abordagem [CHARNIAK86].

Há grande interesse na utilização de ferramentas matemáticas como fractais, redes neuronais e lógica nebulosa em processamento de imagens. Em particular, rede neuronal é uma ferramenta que tem sido aplicada em áreas tão diversas quanto [SIMPSON90], [BRESSLOF91]:

- **medicina**

- análise de radiografias, diagnóstico de doenças, identificação de células, simulação de funções cerebrais;

- **geologia**
caracterização de rochas, prospecção mineral, sensoriamento remoto;
- **transporte**
escalonamento de horários e rotas;
- **defesa**
detecção de alvos, classificação de sinais de radar;
- **telecomunicações**
compressão de dados, reconhecimento de voz ;
- **esportes**
previsão de resultados;
- **economia**
previsão de mercado acionário, taxas de juros, análise de crédito;
- **entretenimento**
efeitos especiais, animação;
- **robótica**
visão computacional, controle de manipuladores, análise de situações;
- **controle de processos**
simulação de processos químicos, processamento de sinal, diagnóstico de falhas;
- **ciência dos materiais**
análise de texturas e fases.

Pode-se observar por estes exemplos que redes neuronais representam um campo multidisciplinar envolvendo pesquisadores de diversas áreas como: neurociência, física, matemática, psicologia e ciência da computação.

Isto vem demonstrar a enorme importância do estudo de redes neuronais, o que motivou o desenvolvimento deste trabalho sobre a sua aplicação em processamento digital de imagens e a implementação de alguns algoritmos como ponto de partida para pesquisas futuras.

1.2 Organização do trabalho

No capítulo 2, são apresentados os conceitos básicos de redes neuronais (RNs) e algumas topologias e paradigmas de RNs, tendo em vista as aplicações apresentadas neste trabalho. Em primeiro lugar, é apresentado o paradigma do neurônio artificial. A seguir, são discutidas as topologias do *perceptron* multicamadas (para aprendizado supervisionado), do mapa auto-organizativo (para aprendizado competitivo) e da rede de Hopfield (para problemas de otimização).

No Capítulo 3, são apresentados alguns conceitos básicos de processamento digital de imagens (PDI), detecção de bordas, compressão e restauração de imagens.

No Capítulo 4, são discutidas as aplicações do *perceptron* multicamadas em compressão de imagens e detecção de bordas, do mapa auto-organizativo em compressão de imagens e da rede de Hopfield em restauração de imagens e detecção de bordas.

No Capítulo 5, são apresentadas as implementações de alguns dos modelos discutidos e os resultados obtidos nas aplicações. De modo a permitir a avaliação dos resultados, estes são comparados com os obtidos por métodos convencionais de PDI.

No Capítulo 6, são discutidas perspectivas de aplicações futuras e novos desenvolvimentos.

As referências bibliográficas incluem livros e artigos de revista que direta ou indiretamente contribuíram com idéias para este trabalho.

Ao longo de todo o texto procurou-se manter uma notação uniforme, mesmo em detrimento das usuais da literatura ou das originalmente presentes nas referências, de modo a facilitar o seu entendimento como um todo.

O trabalho não é uma simples reunião de métodos e resultados, procurou-se fazer um desenvolvimento lógico para se atingir o objetivo de demonstrar as aplicações de redes neuronais em processamento digital de imagens.

2

Redes neuronais

2.1 Conceitos básicos

Redes neuronais artificiais são estruturas computacionais que consistem em um grande número de unidades processadoras simples, chamadas neurônios ou nodos, interconectados numa forma inspirada nos neurônios do cérebro e suas ligações. Alguns autores preferem o termo neural [GORNI93] no lugar de neuronal [CARVALHO91]. Neste trabalho, é utilizado o segundo termo.

Numa rede neuronal, um grupo de neurônios de entrada recebe informações do mundo exterior. Estes neurônios estão conectados a outros escondidos que, por sua vez, estão conectados a outros neurônios de saída. Cada conexão, também chamada de sinapse, tem um peso associado. Dependendo do valor destes pesos, os neurônios de saída são mais ativados ou não. Através de um processo de treinamento, estes pesos são ajustados de modo que a rede aprende a responder adequadamente a estímulos de entrada.

O conhecimento pode ser representado na rede como padrões de atividade distribuída entre muitas unidades, e uma unidade pode tomar parte em diversos padrões diferentes. Em outras palavras, redes neuronais combinam uma arquitetura altamente paralela com uma forma distribuída de representação de conhecimento. São formas de se extrair informação difusa ou indeterminada, a partir de um conjunto de dados previamente escolhido. São modelos indicados para tratar de sistemas abertos ou complexos, pouco conhecidos e que não podem ser adequadamente descritos por um conjunto de regras ou equações. São indicadas para aplicações que requeiram tolerância a falhas, detecção de padrões, diagnóstico, abstração ou generalização, onde haja dados com ruído, incompletos ou distorcidos [WIDROW90].

O conhecimento numa rede neuronal não está armazenado em locais determinados. Não é possível olhar num endereço de memória e verificar o conteúdo de uma variável. O conhecimento está armazenado na topologia e nos pesos.

Não há, em tese, necessidade de se determinar *a priori* quais variáveis ou parâmetros são importantes, pois as redes neuronais tendem a extrair informações da estatística dos dados.

Sempre que possível, é conveniente comparar os resultados obtidos pelas redes neuronais e pelos modelos convencionais.

As redes neurais podem ser implementadas em *software* ou *hardware*. Existem placas comerciais para PCs ou estações de trabalho. A perspectiva atual é da sua implementação em circuitos digitais de integração em larga escala, visando a velocidade de processamento e o uso de arquiteturas paralelas [HUSSAIN91].

É comum a utilização dos termos computação paralela distribuída ou *connectionist computing* para denominar o tipo de abordagem das redes neurais [HERTZ91], [JONES87], [LIPPMANN87].

Há vários tipos de redes neurais, cada uma indicada para certo tipo de aplicação. Contudo todas têm três elementos em comum: elementos processadores distribuídos (neurônios), as conexões entre eles (topologia) e a técnica de determinação dos pesos.

As redes neurais operam em duas fases:

- a fase de determinação dos pesos - que pode ser demorada, quando houver necessidade de um treinamento;
- a fase de produção - uma vez determinados os pesos a rede pode ser usada para classificar ou armazenar padrões e resolver problemas de otimização.

2.1.1 Neurônio

O neurônio é o elemento processador da rede neuronal (ver Figura 1) [LIPPMANN87]. Cada neurônio gera uma saída a partir da combinação de sinais de entrada recebidos de outros neurônios com os quais está conectado, ou a partir de sinais externos. O sinal de saída é o resultado da aplicação da função de transferência [HERTZ91] (também chamada de função de ativação) [ZURADA92] sobre a combinação dos sinais de entrada. O estado do neurônio é representado pelo seu sinal de saída.

O sinal de entrada total do neurônio é obtido pela combinação linear dos sinais recebidos:

$$u_i = \sum_j w_{ij} v_j + \lambda_i, \quad (2.1.1-1)$$

onde u_i = entrada total do neurônio i ;

v_j = saída do neurônio j ;

w_{ij} = peso da conexão entre os neurônios i e j ;

λ_i = termo de *bias* do neurônio i .

O peso da conexão representa uma forma de ponderação do sinal recebido. Em alguns modelos, um termo de *bias* é introduzido como mais um sinal recebido. Isto pode representar um limiar de estímulo. De qualquer modo pode ser tratado como mais um sinal de entrada constante com valor unitário e peso igual a λ . Este modelo representa o paradigma do neurônio artificial.

A função de transferência transforma o sinal de entrada em sinal de saída. Assim, a saída de um neurônio qualquer é dada por:

$$v = f(u),$$

onde f é a função de transferência.

Existem diversas formas desta função. Duas das mais comuns são a função degrau e a sigmóide.

A função degrau produz apenas dois valores de saída, em geral 0 e 1. É uma função descontínua, em que a soma ponderada dos sinais de entrada é comparada com um limiar. Se a soma for maior que este limiar o neurônio dispara, isto é, gera um sinal de saída, caso contrário não. Ou seja, o neurônio terá apenas dois estados.

A função sigmóide (ver Figura 2) é contínua, não decrescente e suave, produzindo valores num intervalo limitado. Em geral este intervalo é $(0,1)$ ou $(-1,+1)$. Tangente hiperbólica é um exemplo. É utilizada em processamento de sinais contínuos.

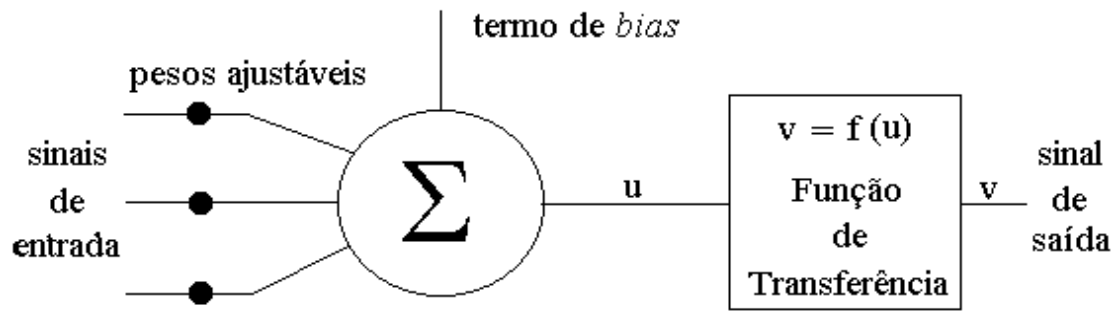


Figura 1. Modelo do neurônio.(Fonte : [LIPPMANN87])

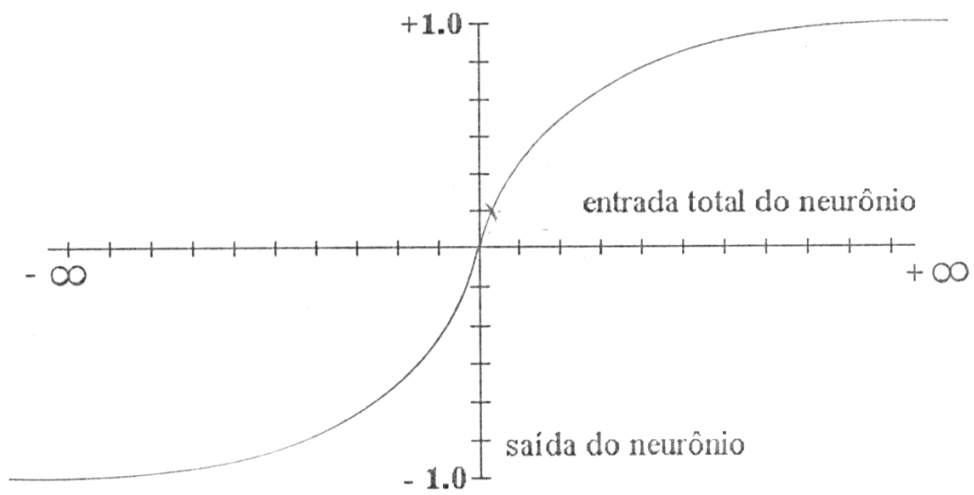


Figura 2. Função sigmóide típica.

2.1.2 Conexões

Os neurônios podem estar conectados de forma total ou parcial. As conexões podem ser bi ou unidirecionais, isto é, um neurônio a pode receber sinal de um neurônio b e vice-versa (ou não).

No caso de conexão parcial, em geral há um sentido de fluxo de dados e os neurônios estão divididos em camadas. São as redes de alimentação para frente, as quais são muito utilizadas em problemas classificatórios. O *perceptron* multicamadas [ZURADA92] é um exemplo deste tipo de rede (ver Figura 3). Este modelo é discutido na Seção 2.2.

No caso da conexão total, não há sentido de propagação dos sinais, uma vez que todos os neurônios estão conectados entre si em ambos os sentidos. São em geral utilizadas em problemas de otimização. A rede de Hopfield [ZURADA92] é um exemplo de modelo com conexão total (ver Figura 4). Este modelo será discutido na Seção 2.4.

Os neurônios podem estar divididos em camadas, com conexão total unidirecional (da entrada para a saída da rede) entre as camadas. Pode haver ou não conexão dentro de uma mesma camada ou mesmo de um neurônio consigo próprio, gerando um auto-estímulo. Pode haver uma, duas ou mais camadas. No caso de mais de duas, as camadas entre as de entrada e de saída são chamadas de escondidas.

Cada conexão tem um peso. Quando o peso é positivo a conexão é dita excitativa, quando for negativo é dita inibitória.

Há modelos que apresentam realimentação da saída para a entrada da rede, são as chamadas redes recorrentes.

Em outros modelos, neurônios vizinhos competem através de interações laterais mútuas e se adaptam progressivamente como classificadores de diferentes padrões de sinais de entrada. É o caso dos mapas auto-organizativos. Este modelo é discutido na Seção 2.3.

Enfim, as conexões definem a topologia da rede.

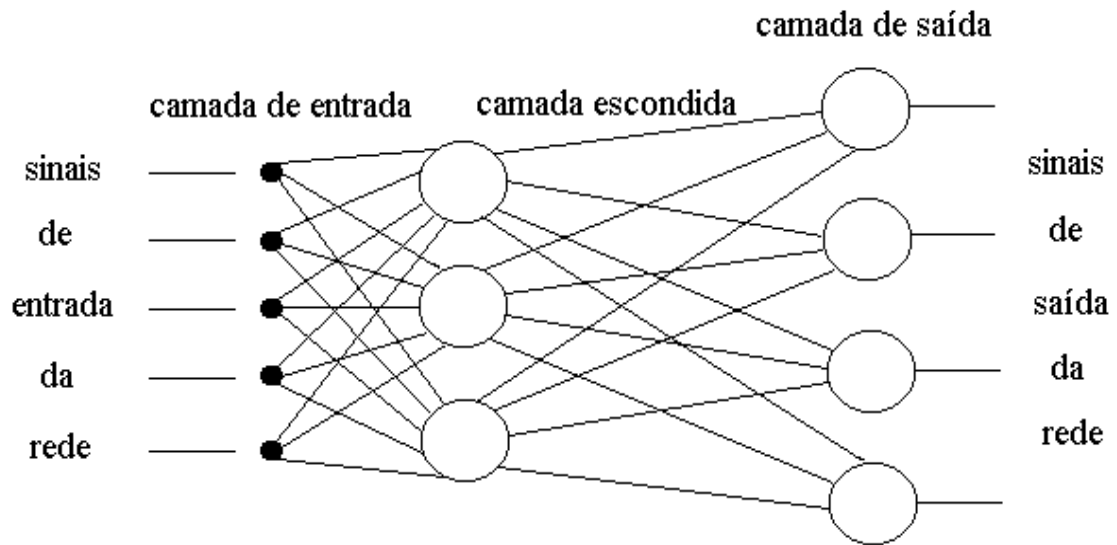


Figura 3. O *perceptron* multicamadas, uma rede com alimentação para frente. (Fonte: [TAGLIARINI91])

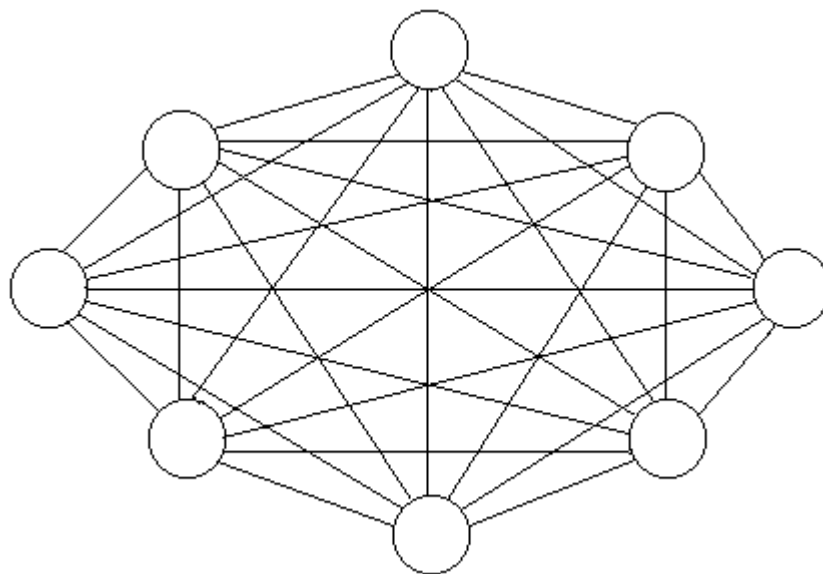


Figura 4. Rede de Hopfield, uma topologia com realimentação total. (Fonte: [GORNI93])

2.1.3 Funcionamento

Além dos neurônios e da topologia, a rede deve ter entrada e saída. Os dados de entrada podem ser apenas a intensidade dos *pixels* como, por exemplo, em reconhecimento de caracteres, ou ainda características de objetos.

Na rede totalmente conectada, todos os neurônios são de entrada e saída. Isto significa que a saída da rede é representada pelo estado de todos os neurônios. Normalmente, os neurônios são inicializados e a rede percorre em ciclos interativos o seu espaço de estados até atingir o equilíbrio. Na rede de Hopfield, este equilíbrio é definido a partir de uma função não-crescente dos estados dos neurônios (uma função de Lyapunov do sistema [ESGOLTS70]), a qual deve ser minimizada.

No caso da rede parcialmente conectada, alguns neurônios são de entrada e outros de saída. Os de entrada recebem os sinais do mundo exterior, os de saída transmitem sinais para o exterior. Isto pode ser comparado com o cérebro, que recebe estímulos externos através dos sentidos, processá-os e devolve uma resposta.

As redes podem operar paralela e assincronamente [LIPPMANN87]: paralelamente, uma vez que alguns neurônios podem depender apenas da saída de apenas alguns outros (na conexão parcial cada camada pode ser processada em paralelo) e assincronamente, pois pode não haver seqüência determinada de processamento (o caso da conexão total, por exemplo) e pode haver processamento aleatório dos neurônios (no caso da conexão total, em cada ciclo iterativo um número aleatório de neurônios pode ser atualizado em ordem também aleatória).

O resultado do processamento pode ser a própria saída dos neurônios, ou esta pode ser mapeada ou classificada em categorias, através de uma tabela de conversão.

Implementações em *software* ou *hardware* podem explorar o paralelismo e aumentar a velocidade de processamento.

2.1.4 Determinação dos pesos

Os pesos das conexões precisam ser determinados para que a rede responda corretamente à dados de entrada. Isto pode ser feito através de um processo de treinamento (também chamado de aprendizado) ou não.

As técnicas de treinamento utilizadas para modificar os valores dos pesos podem ser classificadas em supervisionadas e não-supervisionadas [BRESSLOF91].

Os algoritmos supervisionados trabalham em função de uma saída desejada. Um erro é calculado a partir da saída desejada e daquela gerada pela rede. O erro quadrático médio é uma medida muito usada. Este erro é propagado para trás na rede e os pesos são ajustados para diminuí-lo. É um processo iterativo, em geral bastante demorado. O algoritmo deste tipo mais usado atualmente é o de *backpropagation* [ZURADA92], o qual é discutido na Seção 2.2.

Os algoritmos não-supervisionados trabalham em função de erros calculados internamente, sem considerar uma saída desejada. A rede não tem indicação do resultado esperado. Estes algoritmos tendem a agrupar os dados de entrada em grupos representativos ou categorias. Algoritmos de aprendizado competitivo pertencem a esta classe de algoritmos e são indicados para quantização vetorial. O mapa auto-organizativo de Kohonen [LIPPMANN87] é um exemplo deste tipo de algoritmo e é discutido na Seção 2.3.

No caso da rede de Hopfield [ZURADA92], não há um processo de treinamento, os pesos são calculados a partir de dados de entrada e permanecem fixos. Este caso é discutido na Seção 2.4.

Para ilustrar o processo de treinamento supervisionado, considere-se o caso do *perceptron* de uma camada [LIPPMANN87]. Este *perceptron* deve decidir se um padrão de entrada pertence a uma classe A ou B .

O único neurônio calcula a soma ponderada dos sinais recebidos, subtrai um limiar λ e passa o resultado por uma função de transferência discreta com valores -1 e $+1$. A decisão a ser tomada é associar a resposta $+1$ com sinais de entrada da classe A e -1 com sinais de entrada da classe B .

Uma maneira de visualizar o comportamento da rede é construir um mapa de decisão no hiperespaço dos sinais de entrada. Estas regiões de decisão determinam quais sinais de entrada correspondem à classe A e quais à classe B (ver Figura 5).

O *perceptron* forma duas regiões separadas por um hiperplano. Estas regiões aparecem na Figura 5 como uma linha reta quando há apenas dois sinais de entrada. Neste caso, sinais acima da linha divisória correspondem à classe A e aqueles abaixo à classe B . Naturalmente, a equação da reta depende dos valores dos pesos, os quais precisam ser determinados em função de uma boa separabilidade das regiões.

O procedimento original para ajustes dos pesos do *perceptron* foi desenvolvido por Rosenblatt [ROSENBLATT62]. Este procedimento pode ser descrito como:

passo 1. inicializar pesos $w_j(t)$, $0 \leq j \leq N$ e limiar λ com valores aleatórios pequenos
 $w_j(t)$, λ e N são o peso do sinal j na iteração t , o limiar do neurônio e o número de pesos, respectivamente;

passo 2. entrar com novo conjunto $x(t)$ de sinais de entrada e saída $r(t)$ desejada

$$\mathbf{x}(t) = (x_1(t), x_2(t), \dots, x_N(t));$$

passo 3. calcular a saída do neurônio

$$\begin{aligned} u(t) &= \sum_j w_j(t) x_j(t) + \lambda; \\ v(t) &= f(u(t)); \end{aligned} \quad (2.1.4-1)$$

passo 4. ajustar os pesos

$$\begin{aligned} desvio(t) &= r(t) - v(t); \\ w_j(t+1) &= w_j(t) + \eta [desvio(t)] x_j(t), \text{ para todos os } N \text{ pesos}; \end{aligned} \quad (2.1.4-2)$$

$$r(t) = \begin{cases} +1 & \text{para sinais de entrada da classe A} \\ -1 & \text{para sinais de entrada da classe B} \end{cases}$$

η é um fator de correção menor do que 1;

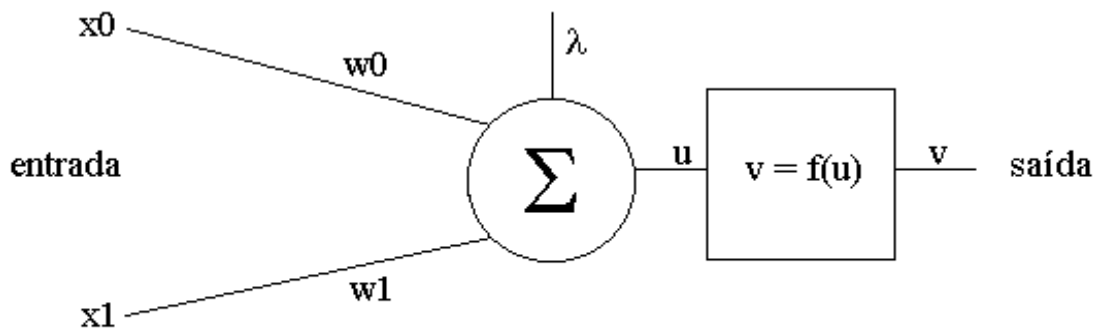
passo 5. retornar ao passo 2.

Obviamente, os pesos não são ajustados quando a saída calculada for igual à desejada. A iteração deve continuar até que o desvio seja suficientemente pequeno para todos os conjuntos de dados de entrada.

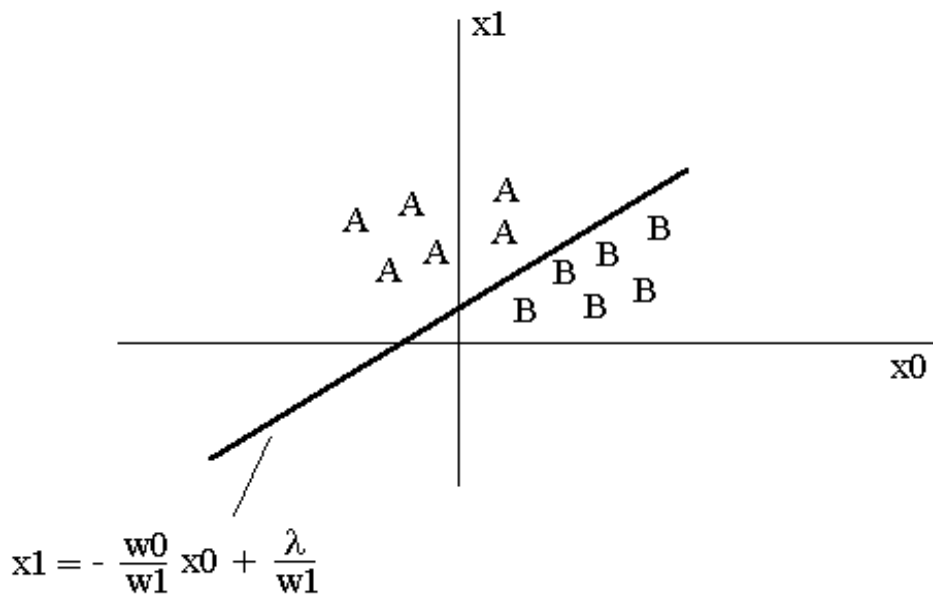
O valor do fator de correção deve ser ajustado em função da velocidade de convergência e, quando necessário, deve ser feito um reajuste para evitar oscilações.

Rosenblatt demonstrou que se os sinais de entrada de ambas as classes são separáveis, ou seja, elas caem em lados opostos de algum hiperplano, então o procedimento descrito converge e posiciona o hiperplano de decisão entre as duas classes.

Naturalmente, é muito importante a escolha do conjunto de dados para treinamento, de modo que ele seja representativo do universo a ser encontrado. Dados de classes diferentes devem ser fornecidos em ordem alternada para evitar tendências no aprendizado.



$$v = \begin{cases} +1 & \text{corresponde a classe B} \\ -1 & \text{corresponde a classe A} \end{cases}$$



equação da fronteira de decisão

Figura 5. Um *perceptron* de camada única separa um vetor de entrada em duas classes A e B. No caso bidimensional, o hiperplano reduz-se a uma reta. (Fonte: [LIPPMANN87])

2.2 *Perceptron* multicamadas

2.2.1 Topologia

O *perceptron* multicamadas [HERTZ91] é uma rede neuronal com uma ou mais camadas escondidas. As camadas escondidas contêm neurônios que não estão diretamente conectados à entrada ou à saída da rede. Na Figura 3, está representado um *perceptron* de duas camadas, modelo utilizado nas implementações que são abordadas no Capítulo 3.

Nessa figura, há duas camadas de neurônios [LIPPMANN87]. Alguns autores [JANSSON91] preferem dizer que este modelo tem três camadas, levando em conta as unidades de entrada. Neste trabalho, é utilizada a nomenclatura de duas camadas.

O fluxo de dados é da camada de entrada para a de saída. Observa-se que os neurônios de uma camada estão completamente conectados apenas aos neurônios da camada seguinte. Não há conexão dentro das camadas.

O *perceptron* multicamadas é uma rede neuronal muito utilizada atualmente. A sua capacidade de resolver problemas baseia-se na não linearidade dos seus neurônios. Se os neurônios fossem elementos lineares, uma camada apenas, com pesos apropriados, poderia reproduzir o mesmo efeito de mais de uma camada [LIPPMANN87].

A Figura 6 apresenta *perceptrons* de uma, duas e três camadas, mostrando exemplos de regiões de decisão que podem ser formadas [LIPPMANN87].

Pode-se notar que um *perceptron* de duas camadas pode formar regiões convexas no espaço de sinais de entrada. Convexa significa que qualquer linha unindo dois pontos de uma região está inteiramente contida na região. Estas regiões têm tantos lados quanto são os neurônios da primeira camada. Na verdade, a capacidade do *perceptron* de duas camadas ainda não está plenamente compreendida, sendo ainda objeto de estudos [GIBSON90].

Isto indica que o número de neurônios deve ser tão grande que possa formar uma região de decisão tão complexa quanto a requerida pelo problema. No entanto, o total de neurônios não deve ser tal que os pesos não possam ser determinados adequadamente a partir do conjunto de dados de treinamento. Por exemplo, dois neurônios num *perceptron* de duas camadas são suficientes para resolver a questão do operador *XOR* (ou exclusivo). Por outro lado, um *perceptron* de uma camada não é suficiente para resolver a questão do operador *XOR*, uma vez que este tipo de rede só consegue definir um hiperplano (ver Figura 6).

Um *perceptron* de três camadas pode formar regiões de decisão complexas, como mostram as classes da Figura 6. Portanto, um número maior de camadas é desnecessário [LIPPMANN87].

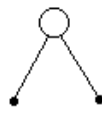
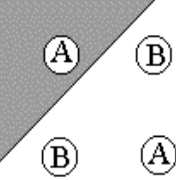
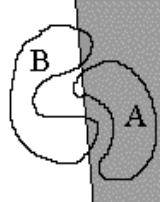
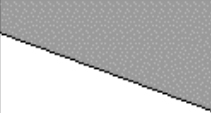
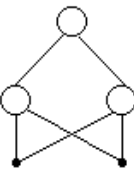
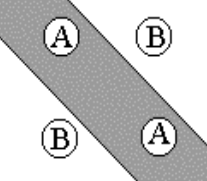
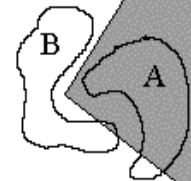
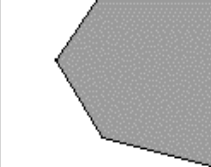
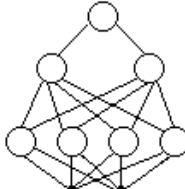
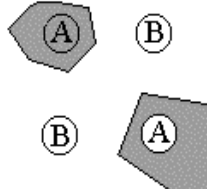
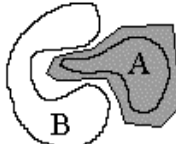
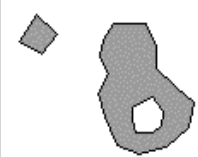
Rede	Regiões de decisão	Operador XOR	Definição de classes	Regiões mais complexas
	regiões separadas por hiperplano			
	Regiões convexas abertas ou fechadas			
	Arbitrárias: complexidade depende do número de neurônios			

Figura 6. Tipos de regiões que podem ser separadas por *perceptrons*. (Fonte: [LIPPMANN87])

2.2.2 O algoritmo de treinamento *backpropagation*

O algoritmo de *backpropagation* [ZURADA92], também conhecido por retropropagação em português, é uma generalização do algoritmo de erro quadrático médio ou regra delta. É usada uma técnica de pesquisa no sentido e direção do gradiente descendente para uma função erro igual à diferença quadrática média entre as saídas esperadas e as geradas pela rede, para todos os padrões do conjunto de treinamento.

O erro para cada padrão pode ser calculado por:

$$E_p = 0,5 \sum_{k=1}^{NS} (r_{pk} - v_{pk})^2; \quad (2.2.2-1)$$

$$v_{pk} = f(u_{pk}(w_l)),$$

onde E_p = erro quadrático para o padrão p ;

r_{pk} = saída esperada do neurônio k da camada de saída para o padrão p ;

v_{pk} = saída calculada para o neurônio k da camada de saída para o padrão p ;

u_{pk} = entrada total do neurônio k da camada de saída para o padrão p ;

w_l = peso da conexão l da rede, com $l=0,1,2 \dots$ número total de conexões;

NS = número de neurônios da camada de saída.

Os pesos das conexões representam as variáveis independentes. A partir do cálculo do gradiente de E_p , resultam derivadas parciais em relação a cada w_l e a partir destas são calculados os incrementos para os pesos. Esta dedução está apresentada em detalhe em [ZURADA92] e o algoritmo resultante [FREEMAN92] para duas camadas de neurônios é descrito a seguir:

passo 1. inicialização

inicializar os pesos das conexões e os termos de *bias* com valores aleatórios pequenos (-0,5 a +0,5); fornecer os parâmetros de aprendizado; inicializar erro total $E = 0$;

passo 2. entrar com novo vetor de dados de entrada $\mathbf{x}_p = (x_{p1}, x_{p2}, \dots, x_{pN})$; se não há mais vetores, ir para passo 12;

passo 3. calcular o somatório de entrada dos neurônios da camada escondida

$$u_{pj}^e = \sum_{i=1}^{NI} w_{ji}^e x_{pi} + \lambda_j^e; \quad NI = \text{número de neurônios da camada de}$$

entrada;

passo 4. calcular a saída da camada escondida

$$v_{pj}^e = f(u_{pj}^e);$$

passo 5. calcular o somatório de entrada para cada neurônio k da camada de saída

$$u_{pk}^s = \sum_{j=1}^{NE} w_{kj}^s v_{pj}^e + \lambda_k^s; \text{ NE} = \text{número de neurônios da camada escondida};$$

passo 6. calcular a saída de cada neurônio da camada de saída

$$v_{pk}^s = f(u_{pk}^s);$$

passo 7. calcular E_p e componentes do erro associados a cada neurônio k da camada de saída

$$\delta_{pk} = (r_{pk} - v_{pk}^s) f'_{pk}(v_{pk}^s);$$

passo 8. calcular componentes do erro para cada neurônio j da camada escondida

$$\delta_{pj} = f'_{pj}(v_{pj}^s) \sum_k \delta_{pk} w_{kj}^s;$$

passo 9. atualizar os pesos das conexões entre a camada de saída e a escondida

$$w_{kj}^s(t+1) = w_{kj}^s(t) + \eta \delta_{pk} u_{pj}^e;$$

passo 10. atualizar os pesos das conexões entre a camada escondida e a entrada da rede

$$w_{ji}^e(t+1) = w_{ji}^e(t) + \eta \delta_{pj} x_{pj};$$

passo 11. incrementar erro total $E = E + E_p$; voltar ao passo 2;

passo 12. se E for menor do que erro mínimo finalizar, caso contrário, reinicializar lista de vetores de padrões e retornar ao passo 2.

Esta iteração deve ser feita até que o erro E seja aceitável. Na verdade, a forma original deste algoritmo [RUMELHART86] prevê a atualização dos pesos apenas quando todos os padrões de entrada tiverem sido introduzidos e um erro quadrático total tiver sido calculado, de modo a minimizar o erro total. Na prática, isto não faz diferença alguma [ZURADA92].

O valor da taxa de aprendizado η afeta o desempenho na fase de treinamento. Em geral, η deve ser um valor positivo pequeno, da ordem de 0,05 a 0,25, de modo a assegurar a convergência. Quanto menor η maior será o número de iterações. η pode ser ajustado durante o treinamento, de modo a acelerar o processo de convergência e evitar oscilações.

Outra forma de acelerar a convergência é fazer uso da técnica de *momentum* (em analogia ao conceito da Física). Ao atualizar-se os pesos, uma fração do ajuste anterior é acrescentada. Este acréscimo tende a manter o ajuste no mesmo sentido [ZURADA92]. As equações de ajuste dos pesos das conexões entre a camada de saída e a escondida passam a ser, então:

$$w_{kj}^s(t+1) = w_{kj}^s(t) + \eta \delta_{pk} X_j + \alpha \Delta w_{kj}^s(t-1), \quad (2.2.2-2)$$

onde

$$\begin{aligned} \Delta w_{kj}^s(t-1) &= \text{variação do peso na iteração } t-1; \\ \alpha &= \text{coeficiente do termo de momentum.} \end{aligned}$$

Em geral, tem-se $\alpha < 1$. Equação semelhante aplica-se para a camada escondida. Esta é a equação de atualização utilizada neste trabalho.

O processo de treinamento pode convergir para um mínimo local em lugar do global. Isto depende do valor inicial dos pesos, do número de neurônios utilizados e dos parâmetros de aprendizado. Se a rede converge para um mínimo local inaceitável em termos do erro, o processo de aprendizado deve ser reiniciado com a alteração de alguns dos parâmetros citados. Na prática, isto não ocorre com muita frequência. Por outro lado, se o processo converge para um mínimo aceitável sob o ponto de vista do erro obtido, não faz diferença se o mínimo é local ou não [FREEMAN92].

2.3 Mapa auto-organizativo

Na classificação de padrões, dados de entrada de um conjunto possivelmente infinito são agrupados em subconjuntos, grupos ou aglomerados, de acordo com algum critério. Frequentemente, tal critério não é conhecido *a priori*. Isto torna impossível o aprendizado supervisionado. O aprendizado não-supervisionado ajusta os pesos sem qualquer medida externa de erro. A estrutura de rede resultante reflete características importantes dos dados de entrada. Isto pode, então, ser usado como base para um esquema de classificação. Desta forma, através de um processo de auto-organização, a rede aprende a formar representações internas dos dados de entrada de modo a codificar tais características. As representações adequadas não são conhecidas de antemão, mas resultam do processo de aprendizado. Como exemplo de aprendizado auto-organizativo, é descrito o mapa auto-organizativo de Kohonen [KRISHNAMURTHY90]. Uma variação deste modelo, o mapa com aprendizado sensível à frequência [KRISHNAMURTHY90], também é apresentado.

2.3.1 Mapa auto-organizativo de Kohonen

O mapa auto-organizativo de Kohonen, KSFM (*Kohonen self-organizing feature map*) é um algoritmo competitivo proposto por Kohonen como um modelo para a formação de mapas topológicos no cérebro [KOHONEN91].

As unidades, isto é os neurônios, são tradicionalmente organizados num arranjo bi-dimensional (ver Figura 7). Cada neurônio tem um vetor de pesos \mathbf{w} associado. Durante o treinamento, cada neurônio do arranjo recebe sinais de entrada. Considerando cada um destes sinais como componentes de um vetor \mathbf{x} , na iteração t , este vetor é denotado por $\mathbf{x}(t)$. O sinal de saída de um neurônio i é calculado por:

$$D_i(t) = f(\|\mathbf{x}(t) - \mathbf{w}_i(t)\|),$$

onde f é uma função real suave de tipo gaussiano em torno de zero. $\|\dots\|$ representa o módulo da diferença entre o vetor de pesos \mathbf{w} e o vetor de entrada \mathbf{x} . Os neurônios competem de acordo com uma estratégia do tipo o *vencedor leva tudo*, na qual o neurônio de melhor resposta é selecionado, isto é, o neurônio de menor desvio. Em outras palavras, a unidade escolhida é aquela cujo vetor $\mathbf{w}(t)$ de pesos está mais próximo a $\mathbf{x}(t)$.

Durante o processo de treinamento, não só o neurônio vencedor tem seus pesos ajustados, mas também os neurônios dentro de uma vizinhança em torno do vencedor, correspondente a uma distância d_{max} . O tamanho desta vizinhança é gradativamente reduzido.

Inicialmente, o vetor de pesos \mathbf{w} está distribuído aleatoriamente no universo de dados de entrada. O algoritmo de ajuste pode ser resumido por:

passo 1. inicializar os vetores de pesos \mathbf{w} com valores aleatórios pequenos e d_{max} com

valor suficientemente grande para incluir todas as unidades;

passo 3. entrar com novo vetor de sinais $\mathbf{x}(t)$;

passo 4. em cada iteração t , selecionar a unidade j de menor desvio $D_j(t)$;

passo 5. a unidade j vencedora tem seus pesos ajustados por

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \alpha(0) [\mathbf{x}(t) - \mathbf{w}_j(t)]$$

e cada unidade i , a uma distância $d(t)$ da unidade j menor do que d_{max} , tem seus pesos ajustados por

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(d(t)) [\mathbf{x}(t) - \mathbf{w}_i(t)], \quad (2.3.1-1)$$

onde $0 < \alpha(d(t)) < 1$ é uma função decrescente com a distância $d(t)$;

passo 6 diminuir d_{max} ;

passo 7 repetir a partir do passo 2 para todos os vetores de entrada.

A distância $d(t)$ num arranjo bidimensional é apenas a dimensão de uma região em torno de um neurônio, região esta que é gradativamente reduzida.

Desta forma, todas as unidades dentro de uma vizinhança da unidade escolhida têm seus pesos ajustados na direção do vetor $x(t)$. Tem-se verificado que para obter-se bons resultados, d_{max} deve ser reduzido gradativamente. O procedimento de ajuste deve ser feito de modo que diferentes regiões do arranjo convirjam para diferentes vetores de sinais de entrada. Assim, é produzido um mapeamento topográfico bidimensional do espaço de dados de entrada para a rede, no qual cada unidade é identificada com o vetor sinal de entrada que obteve melhor resposta no treinamento. Isto permite que estas unidades sejam agrupadas em regiões ou grupos em que a resposta para um dado vetor de entrada é máxima.

Diz-se, então, que o vetor de entrada ativa o grupo que lhe está associado. Cada aglomerado pode ser rotulado, identificado por um índice, para servir de código de representação dos vetores. Além disto, o padrão de resposta da rede reflete relações entre os dados de entrada, as quais são determinadas pela regra de aprendizado. Em particular, sinais de entrada semelhantes ativam regiões vizinhas, de modo que relações de distância entre vetores de entrada são preservadas no mapeamento da rede. Em outras palavras, características semelhantes são agrupadas próximas e características dessemelhantes permanecem separadas.

Após o treinamento, qualquer vetor de entrada, não necessariamente utilizado no treinamento, conduzirá a seleção da unidade cujo vetor de pesos lhe for mais próximo. A saída da rede é tomada como a do aglomerado a que a unidade pertence. Desta forma, a rede funciona como um classificador.

Observa-se este detalhe em relação ao aproveitamento em processamento digital de imagens (PDI), quando sinais de entrada poderão representar a intensidade dos *pixels* da imagem, uma vez que regiões semelhantes de uma imagem poderão ser mapeadas para a mesma unidade no arranjo bidimensional.

O algoritmo descrito pode conduzir à subutilização de neurônios, isto é, alguns neurônios podem sair vencedores muitas vezes enquanto outros não. Isto pode acarretar uma má distribuição das categorias. Na próxima seção é apresentado um outro algoritmo para contornar este problema.

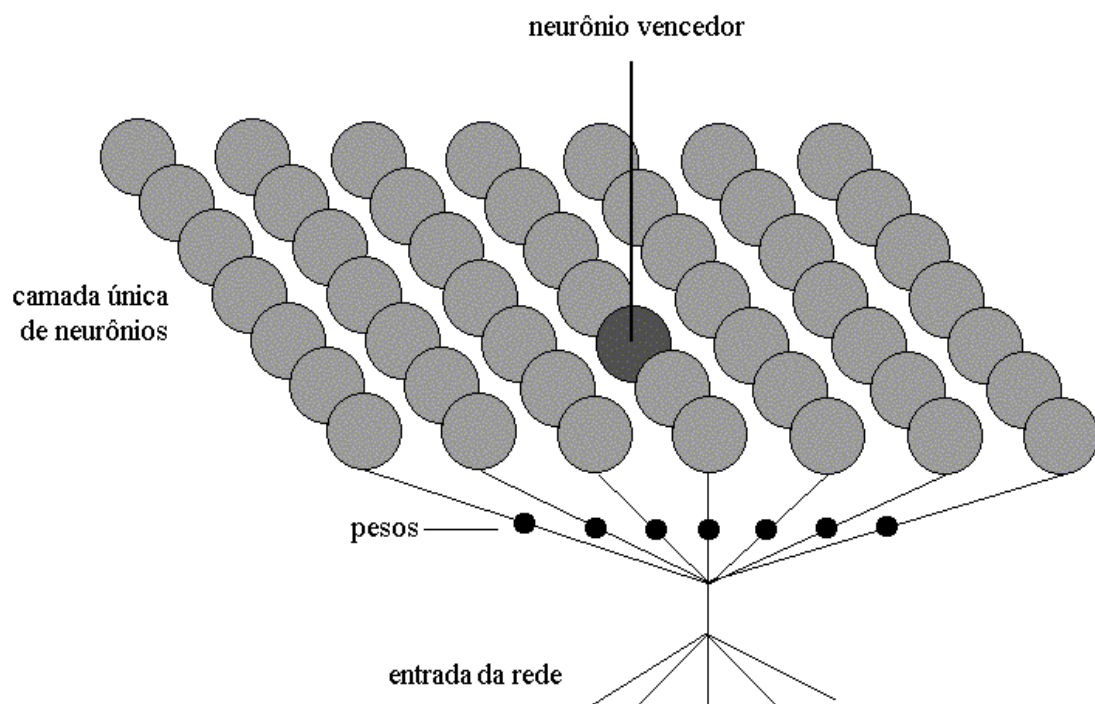


Figura 7. Mapa auto-organizativo de Kohonen. (Fonte: [CAUDILL93])

2.3.2 Mapa para aprendizado competitivo sensível à frequência

O aprendizado competitivo sensível à frequência [KRISHNAMURTHY90], FSCL (*frequency sensitive competitive learning*), trata o problema da subutilização de algumas unidades pela incorporação de uma medida da frequência de sucesso de uma unidade no processo de aprendizado. Cada neurônio mantém um contador do número de vezes em que foi a unidade vencedora. A medida utilizada para indicar a unidade vencedora passa a ser, então

$$D_i(t) = F(u_i) f(\|\mathbf{x}(t) - \mathbf{w}_i(t)\|), \quad (2.3.2-1)$$

onde $F(u_i)$ é uma função não decrescente chamada de função de ajuste. Esta é uma maneira de introduzir uma ponderação em função do contador. Quanto mais frequentemente um unidade vencer, maior será o valor do seu contador e, conseqüentemente, maior será o desvio. Isto permite que outras unidades vençam a competição.

O algoritmo FSCL pode ser resumido como:

passo 1. inicializar os contadores u_i de todas as unidades em zero;

passo 2. inicializar todos os pesos w_i com valores aleatórios pequenos;

passo 3. calcular o desvio $D_i(t)$ para cada unidade i ;

passo 4. selecionar a unidade j de menor desvio e rotulá-la como vencedora;

passo 5. ajustar os pesos da unidade selecionada por

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \alpha(t) [\mathbf{x}(t) - \mathbf{w}_j(t)], \quad (2.3.2-2)$$

onde $0 < \alpha(t) < 1$ é uma função decrescente com a iteração;

passo 6. repetir, a partir do passo 3, para todos os vetores de entrada.

A função de ajuste fornece uma maneira simples de controlar o comportamento do processo de aprendizado. Por exemplo, escolher $F(u) = 1$ reduz o algoritmo ao modelo clássico de aprendizado competitivo [ZURADA92]. Escolher $F(u) = u$, também é uma maneira simples de considerar a influência do número de vitórias. Esta segunda forma foi utilizada neste trabalho [KRISHNAMURTHY90].

Para $\alpha(t)$ foi utilizada a função $\beta e^{-(u(t)/k)}$, com $0 < \beta < 1$ e k um inteiro positivo grande [KRISHNAMURTHY90].

2.4. Rede de Hopfield

2.4.1 Condições de estabilidade Cohen-Grossberg

Grossberg [TAGLIARINI91] desenvolveu um modelo matemático que engloba uma grande variedade de modelos de redes neuronais. A análise deste modelo por Cohen e Grossberg [TAGLIARINI91] resultou nas condições para que os sistemas de equações diferenciais usados para caracterizar uma série de modelos de redes neuronais converjam para estados estáveis. O modelo analisado é um sistema dinâmico de equações diferenciais acopladas da forma:

$$\frac{du_i}{dt} = a_i(u_i) \left[b_i(u_i) - \sum_{j=1}^N C_{ij} f_j(u_j) \right]. \quad (2.4.1-1)$$

Pode-se mostrar que existe uma função de Lyapunov [ESGOLTS70] para este sistema se a matriz $[C_{ij}]$ e as funções a_i , b_i e f_j satisfizerem às seguintes condições:

- a matriz deve ser simétrica, isto é, $C_{ij} = C_{ji}$;
- as funções a_i e b_i devem ser contínuas com a_i não negativas e
- as funções f_j devem ser não decrescentes.

Uma função de Lyapunov para um sistema dinâmico estabelece condições para o comportamento coletivo do sistema composto pelas equações. A idéia básica é que o sistema sempre evolui de modo a não aumentar o valor desta função. A sua existência garante, então, que o sistema seguirá uma trajetória no seu hiper-espaço de estados até atingir um estado estável, independente do estado inicial, desde que as condições mencionadas sejam satisfeitas. A demonstração de Cohen-Grossberg estabelece a estabilidade de sistemas descritos pela Equação (2.4.1-1). Um destes casos é o modelo de Hopfield discutido a seguir [ZURADA92].

2.4.2 O modelo de Hopfield

O primeiro modelo introduzido por Hopfield empregava neurônios de apenas dois estados. Este modelo foi usado para desenvolver memórias enderaçáveis pelo conteúdo [HOPFIELD86]. Posteriormente, Hopfield introduziu uma versão modificada deste modelo que empregava uma função de transferência não-linear para os neurônios. É o modelo contínuo que corresponde a um caso particular do modelo de Grossberg para redes neuronais aditivas [TAGLIARINI91].

No modelo contínuo, o comportamento de um neurônio i é caracterizado pelo seu nível de ativação u_i , que satisfaz à seguinte equação diferencial:

$$\frac{du_i}{dt} = -\gamma_i u_i + \sum_{j=1}^N w_{ij} f_j(u_j) + \lambda_i, \quad (2.4.2-1)$$

onde $-\gamma_i u_i$ é um termo de decaimento, w_{ij} é o peso da conexão entre os neurônios i e j , $f_j(u_j)$ é a função de transferência para o neurônio j e λ_i é um sinal de entrada externo para o neurônio i . Na ausência de sinais externos e de outros neurônios, o termo de decaimento faz com que a atividade de um neurônio tenda para zero. A saída v_i de um neurônio i é dada pela sua função de transferência $v_i = f_i(u_i)$. A função de transferência é tipicamente uma função suave não decrescente do tipo sigmóide, uma tangente hiperbólica, por exemplo.

Como $f_i(u_i)$ é não decrescente, ela satisfaz à condição de Cohen-Grossberg para estabilidade. Desta forma, se estímulos externos forem mantidos constantes, uma rede neuronal modelada pela Equação (2.4.2-1) eventualmente atingirá o equilíbrio.

Hopfield [TAGLIARINI91] descobriu uma função de Lyapunov para uma rede neuronal caracterizada pela Equação (2.4.2-1), que pode ser expressa por:

$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} v_i v_j - \sum_{i=1}^N \lambda_i v_i. \quad (2.4.2-2)$$

A esta expressão, Hopfield refere-se como energia computacional da rede, ou simplesmente função de energia. Este termo é uma analogia entre o comportamento da rede e o de certos sistemas físicos. Da mesma forma que sistemas físicos podem evoluir na direção de um estado de energia mínima, uma rede neuronal também pode evoluir na direção de um mínimo da função de energia. Os estados estáveis da rede correspondem a mínimos locais desta função.

Hopfield [HOPFIELD85] observou que é possível usar a função de energia em problemas de otimização. Uma vez que a rede neuronal minimizará a função de energia, pode-se modelar um problema associando-se variáveis do problema a variáveis desta função. Desenvolver uma rede neuronal para resolver problemas de otimização implica em escolher valores apropriados para os pesos w_{ij} das conexões e para os sinais externos λ_j . Hopfield e Tank utilizaram este conceito para configurar redes neuronais para diversas aplicações, por exemplo o clássico problema do caixeiro viajante [HOPFIELD85].

A função de energia forma o embasamento teórico para problemas de otimização utilizando-se redes neuronais.

Uma função de transferência discreta binária pode ser aproximada pelo caso contínuo quando o ganho proporcionado por $v_i = f_i(u_i)$ for muito grande, resultando sempre nos valores saturados. Relembrando a Equação (2.1.1-1), uma função de transferência do tipo degrau é um caso discreto binário:

$$v_i = \begin{cases} 1 & \lambda_i > \text{limiar;} \\ 0 & \text{caso contrário;} \end{cases}$$

Os pesos das conexões precisam ser determinados em função de estados já conhecidos da rede e de alguma condição de contorno.

No caso de utilização deste modelo para armazenar um conjunto de padrões é utilizada a seguinte equação [JOSIN87] para os pesos:

$$w_{ij} = \sum_{k=1}^N (2 v_i(k) - 1)(2 v_j(k) - 1), \quad (2.4.2-3)$$

onde $v_i(k)$ é a saída (estado) do neurônio i no padrão k , $i \neq j$ e $w_{ii} = 0$.

Não havendo sinais externos, a função de energia reduz-se a:

$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} v_i v_j. \quad (2.4.2-4)$$

Considerando que um conjunto de padrões tenha sido previamente armazenado na rede, esta é inicializada com os neurônios em um novo estado não pertencente a este conjunto.

Como foi dito anteriormente, a Equação (2.4.2-4) é uma função de Lyapunov e define uma superfície num hiperespaço definido por v_i . Desta forma, a rede percorrerá uma trajetória nesta superfície de modo a atingir um mínimo local mais próximo, representado pelos estados armazenados. Isto significa que a rede "recordará" o estado armazenado mais próximo.

Um exemplo bem simples de aplicação destes conceitos é o reconhecimento de caracteres representados em uma matriz binária [SCHALKOFF89]. Ver Figura 8.

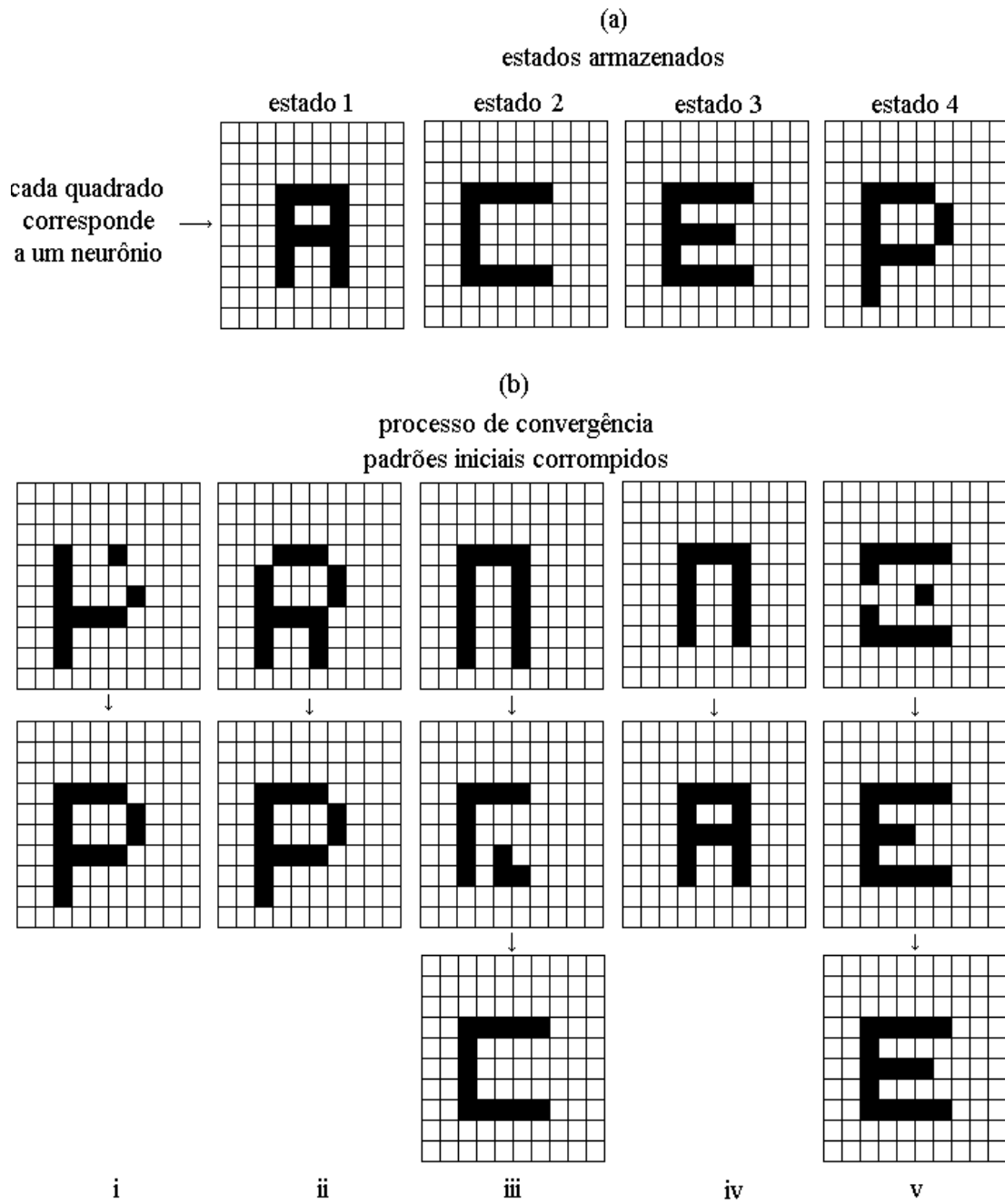


Figura 8. Reconhecimento de caracteres pela rede de Hopfield. (a) Padrões armazenados. (b) Processo de convergência para cinco padrões iniciais corrompidos. (Fonte: [SCHALKOFF89])

O procedimento consiste em:

passo 1. mapear uma matriz binária $N \times N$ como uma rede neuronal de N^2 neurônios;

passo 2. armazenar cada caractere a ser reconhecido como um estado estável utilizando a Equação (2.4.2-3). O estado de cada neurônio num estado estável corresponde ao seu valor binário na matriz que representa o caractere;

passo 3. introduzir um caractere desconhecido na forma de um estado inicial da rede;

passo 4. fazer a rede evoluir assíncrona e aleatoriamente no seu espaço de estados até atingir o estado estável mais próximo. Na Figura 8(b), está representado um exemplo do processo de convergência [SCHALKOFF89].

No Capítulo 4, são discutidas aplicações do modelo de Hopfield em detecção de bordas e restauração de imagens.

2.5 Conclusão

Neste capítulo, foram apresentados conceitos básicos de redes neuronais. Os modelos foram apresentados tendo em vista as aplicações que serão discutidas nos capítulos subsequentes.

Foi apresentado o modelo do *perceptron* multicamadas, cuja aplicação em compressão de imagens e detecção de bordas é apresentada no Capítulo 4.

O mapa auto-organizativo também foi discutido e uma implementação para quantização vetorial é mostrada no Capítulo 4.

Finalmente, foi apresentado o modelo de Hopfield, cuja aplicação em detecção de bordas e restauração de imagens é discutida no Capítulo 4.

Ao final deste trabalho, uma série de referências é indicada aos interessados nos fundamentos e aplicações de redes neuronais.

3

Processamento digital de imagens

3.1. Conceitos básicos e terminologia

Processamento digital de imagens significa processamento numérico de uma representação matemática do estímulo luminoso percebido pelo sistema visual humano.

Neste trabalho, imagem significa uma função $I(x,y)$ de intensidade luminosa no ponto (x,y) percebida pelo sistema visual humano. É uma quantidade finita, real e não-negativa [GONZALEZ87].

A energia de um determinado comprimento de onda recebida pelo sistema visual pode ser representada por [JAIN89]

$$f(\lambda) = \rho(\lambda) L(\lambda),$$

onde $\rho(\lambda)$ = coeficiente de reflexão do objeto para o comprimento de onda λ e
 $L(\lambda)$ = densidade de energia incidente para o comprimento de onda λ .

Desta forma,
$$I(x,y) = \int f(x,y,\lambda) V(\lambda) d\lambda,$$

onde $V(\lambda)$ é a função de eficiência luminosa do sistema visual humano.

Uma imagem pode ser acromática (também chamada de imagem em tons ou níveis de cinza) ou colorida.

Imagem colorida, no modelo RGB (**R**ed, **G**reen, **B**lue) [GALBIATI90], significa uma representação em três componentes, ou bandas, da distribuição de energia que um objeto ou uma cena produz e que o sistema visual capta. A imagem colorida terá, então, três componentes de intensidade: $I_r(x,y)$ para o vermelho, $I_g(x,y)$ para o verde e $I_b(x,y)$ para o azul.

Imagem digital é uma imagem que sofreu um processo de discretização tanto nas coordenadas espaciais, quanto na intensidade dos componentes do modelo RGB. Cada ponto da imagem é chamado de *pixel* (*picture element*). O tamanho físico da área representada por um *pixel* é chamado de resolução espacial do *pixel*.

A representação numérica da intensidade luminosa da imagem discretizada em uma matriz bidimensional é chamada de imagem digital. Cada ponto da matriz representa um *pixel* da imagem. Assim, pode-se dizer que as imagens digitais são representadas por linhas e colunas.

O valor mínimo de cada componente de intensidade é zero e o máximo depende do formato de armazenamento no computador. Todas as imagens digitais neste trabalho serão representadas no modelo RGB de 24 *bits* por *pixel*, um *byte* por componente. As imagens em nível de cinza são codificadas com os três componentes iguais, permitindo até 256 níveis de cinza. Todos os valores são inteiros. Quando algum processamento resultar em valores fracionários, estes devem sofrer algum tipo de arredondamento.

Vizinhança, ou janela em torno de um *pixel*, é o conjunto de pontos imediatamente adjacentes ao mesmo. Por exemplo, uma janela 3 x 3 em torno do *pixel* (x,y) significa os pontos $(x-1,y-1)$, $(x-1,y)$, $(x-1,y+1)$, $(x,y-1)$, (x,y) , $(x,y+1)$, $(x+1,y-1)$, $(x+1,y)$, $(x+1,y+1)$ ou, em forma matricial

$(x-1, y-1)$	$(x-1, y)$	$(x-1, y+1)$
$(x, y-1)$	(x, y)	$(x, y+1)$
$(x+1, y-1)$	$(x+1, y)$	$(x+1, y+1)$

De forma semelhante, vizinhanças maiores tais como: 4 x 4, 5 x 5 ou $N \times N$ podem ser definidas.

3.2 Compressão

Compressão de imagens digitais refere-se à redução do número de *bits* para armazenar e/ou transmitir uma imagem [JAIN89]. Imagens típicas de televisão têm resolução espacial de aproximadamente 512 x 512 *pixels* por quadro. A 8 *bits* por componente de cor e 30 quadros por segundo, isto resulta em cerca de 190×10^6 *bits/s*. Isto representa um enorme volume de dados. Este fato justifica plenamente o enorme interesse no desenvolvimento de técnicas de compressão de dados, em particular compressão de imagens.

Aplicações para compressão na transmissão de imagens encontram-se em transmissões de televisão, sensoriamento remoto, comunicação militar, radar e sonar, teleconferência, comunicação por computador, facsímile, etc. O armazenamento de imagens é necessário para documentos, imagens médicas geradas por tomografia computadorizada ou NMR (*Nuclear Magnetic Resonance*), radiografia digital, películas cinematográficas, imagens de satélite, cartas meteorológicas, pesquisas geológicas, etc. [JAIN89].

Os métodos de compressão de imagens se dividem em duas categorias [JAIN89]. Na primeira, encontram-se os métodos chamados de codificação sem perda de informação, estes métodos em geral exploram o fato de haver repetições nas imagens. O método de *run-length encoding* é desta categoria [JAIN89]. Na segunda, encontram-se os métodos com perda de informação. A codificação por transformada de Fourier e codificação por quantização vetorial são dois exemplos desta categoria [JAIN89].

A compressão de imagens por redes neuronais enquadra-se na segunda categoria de compressão de imagens.

Em geral, a compressão com perda de informação é feita por janelas, através de uma redução do número de *bits* para representar os *pixels* da janela. Isto é obtido através de dados estatísticos dos *pixels*. A perda de informação nem sempre constitui um problema, muitas vezes o usuário fica plenamente satisfeito com uma pequena perda de "qualidade" da imagem codificada em relação à original.

Naturalmente, há um compromisso entre a redução do número de *bits* por *pixel* e a qualidade da imagem reproduzida após a descompressão. Em geral, há perda de qualidade pela introdução de ruído ou de artefatos.

Para medir a qualidade de uma imagem descomprimida em relação à original pode ser utilizada a relação sinal/ruído definida por [KOSKO92-1]:

$$SNR = 10 \log_{10} \left(\frac{SM}{MSE} \right), \quad (3.2-1)$$

$$\text{com } SM = \frac{1}{N \cdot M} \sum_{i=1}^N \sum_{j=1}^M \sum_{k=1}^3 (IO_{i,j,k})^2.$$

$$MSE = \frac{1}{N \cdot M} \sum_{i=1}^N \sum_{j=1}^M \sum_{k=1}^3 (ID_{i,j,k} - IO_{i,j,k})^2,$$

onde $ID_{i,j,k}$ = intensidade no ponto (i,j) no componente k da imagem descomprimida;

$IO_{i,j,k}$ = intensidade no ponto (i,j) no componente k da imagem original;

N = número de linhas da imagem;

M = número de colunas da imagem;

MSE = erro quadrático médio (*Mean Square Error*);

SNR = relação sinal/ruído (*Signal to Noise Ratio*).

Esta medida é um valor numérico. No entanto, deve-se considerar que, às vezes, resultados numericamente inferiores podem ser qualitativamente superiores, pois pode haver eliminação de detalhes, introdução de ruído e artefatos. Ou seja, a inspeção visual humana é muito importante.

O valor da taxa de compressão depende da imagem. Uma imagem com apenas um nível de cinza pode ser inteiramente definida por 1 *byte* (correspondendo a 256 possíveis níveis de cinza) e pelas suas dimensões. Por outro lado, uma imagem com variações nos níveis de cinza ao longo da imagem necessita de um maior número de *bytes* para sua representação.

Outra possibilidade é utilizar a razão entre o número de *bits* necessários para representar os *pixels* de uma janela na imagem comprimida e o número de *pixels* da janela original. Esta razão pode ser fixada pelo usuário e independe da imagem. O fato de haver janelas repetidas na imagem não é levado em consideração. Esta medida é adotada neste trabalho para medir a compressão da imagem [COTTRELL87].

$$NBPP = \text{número de bits por pixel da janela} = \frac{N_{\text{bits}}}{N_{\text{pixels}}}, \quad (3.2-2)$$

onde N_{bits} = número de *bits* para representar os *pixels* da janela na imagem comprimida;

N_{pixels} = número de *pixels* da janela original.

Por exemplo, uma janela 3 x 3 tem 9 *pixels* e necessita de 72 *bits* (1 *byte* = 8 *bits* para cada *pixel*) para ser armazenada, resultando 72/9 *bits* por *pixel*. Se um método de compressão armazenar a mesma janela, utilizando-se de apenas 24 *bits*, isto resulta em $NBPP = 24/9$ *bits* por *pixel* da janela. Ou seja, há uma compressão.

3.3 Detecção de bordas

Uma questão fundamental em processamento digital de imagens é a detecção de bordas [GONZALEZ87]. Elas caracterizam fronteiras de objetos e, portanto, são úteis em segmentação e identificação de objetos em cenas. Idealmente, as bordas podem ser vistas como pontos em torno dos quais há variações abruptas de níveis de cinza. Os algoritmos de detecção de bordas se baseiam neste conceito.

Uma classe destes algoritmos é a dos operadores gradientes [JAIN89]. Estes algoritmos se baseiam no cálculo da derivada do nível de cinza em direções ortogonais ou na derivada direcional. Na sua forma discreta são representados por máscaras de convolução espacial [JAIN89]. Na verdade, eles são casos particulares de máscaras de convolução espacial do tipo:

<i>a</i>	<i>b</i>	<i>c</i>
<i>d</i>	<i>e</i>	<i>f</i>
<i>g</i>	<i>h</i>	<i>i</i>

Fazendo-se a convolução desta máscara com uma janela 3 x 3 tem-se, dependendo dos valores dos coeficientes *a, b, c, d, e, f, g, h, i*, os operadores de Roberts, Sobel ou Laplaciano ou, ainda, operadores de suavização como os filtros da média e gaussiano [GONZALEZ87].

Em geral, após a aplicação da máscara, o resultado é comparado com um limiar de forma a definir-se se o *pixel* central da janela 3 x 3 representa uma borda ou não. Em caso positivo, é atribuído ao *pixel* o máximo valor de nível de cinza (255 no caso deste trabalho), caso contrário é atribuído o valor zero. A imagem é sequencialmente percorrida de modo que no final está formado o chamado mapa de bordas.

Outra possibilidade é o realce de borda [JAIN89]. Neste caso, se o valor da operação resultou menor do que o limiar, o *pixel* central da máscara é zerado, caso contrário seu valor é mantido.

Os operadores de detecção de borda são filtros passa-altas e, portanto, têm muitas vezes o inconveniente de realçar ruído presente na imagem. Em geral, este problema é tratado através da introdução de limiares para o valor da derivada ou pela aplicação de um filtro de suavização (filtro gaussiano por exemplo) em conjunto com um operador para detecção de bordas.

Quando a imagem é colorida, o problema fica mais complicado. Em geral, faz-se um processamento por componente de cor RGB. No entanto, o critério de limiar neste caso pode ser falacioso, uma vez que pequenas variações de intensidade nos componentes de cor podem significar uma nítida mudança de região aos olhos de uma pessoa. Uma outra alternativa é transformar a imagem colorida em tons de cinza antes do processamento. Seja qual for o método utilizado, o processamento de imagens coloridas é bem mais complicado.

Uma outra alternativa é considerar a vizinhança do *pixel*. Deve-se lembrar que bordas em geral são parte de contornos, isto é, os *pixels* das bordas estão conectados formando um contorno, a menos que sejam ruído. A aplicação de redes neuronais, a ser discutida no Capítulo 4, leva em conta este fato, através da comparação com padrões de linhas.

3.4 Restauração de imagens

O objetivo da restauração de imagens é melhorar de alguma forma uma dada imagem que tenha sido degradada [GONZALEZ87]. Isto é feito através de algum conhecimento do processo de degradação. Ou seja, os processos de restauração de imagem de alguma forma tentam modelar o processo de degradação e invertê-lo para recuperar a imagem original através da filtragem.

A degradação pode aparecer na forma de ruído do sensor, turvamento devido ao foco da câmera ou ao movimento relativo entre objeto e câmera, ruído atmosférico, etc.

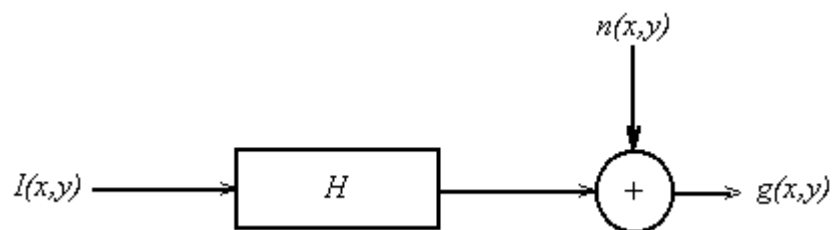
A restauração de imagem difere do realce de imagem no sentido de que o realce procura acentuar ou destacar características da imagem ao invés de recuperar alguma deterioração.

Ao longo dos anos, vários métodos foram desenvolvidos, como: filtro inverso [ANDREWS77], filtro de Wiener [ANDREWS77], filtro Kalman [WOODS81] decomposição pseudoinversa em valor singular [ANDREWS77] e muitas outras. As primeiras técnicas de restauração são baseadas no domínio da frequência. Mais modernamente o problema é tratado algebricamente através de sistemas de equações. Esta será a abordagem adotada neste trabalho.

O modelo a ser descrito a seguir, é discutido em detalhe em [GONZALEZ87] e corresponde a degradação por função de turvamento espacialmente invariante e ruído.

O processo de degradação pode ser modelado como um operador H que, em conjunto com um termo de ruído $n(x,y)$ opera sobre uma imagem $I(x,y)$ para produzir uma imagem degradada $g(x,y)$. O problema da restauração da imagem consiste, então, em obter uma aproximação de $I(x,y)$, conhecida a imagem degradada $g(x,y)$ e o processo de degradação na forma do operador H . Supõe-se que o conhecimento sobre o ruído $n(x,y)$ seja apenas de natureza estatística.

Este modelo pode ser representado esquematicamente por



A relação representada neste diagrama pode ser expressa por

$$g(x,y) = H I(x,y) + n(x,y); \quad (3.4.2-1)$$

Por enquanto, suponha-se $n(x,y) = 0$, de modo que $g(x,y) = H I(x,y)$. H é definido como linear se

$$H [k_1 I_1(x,y) + k_2 I_2(x,y)] = k_1 H [I_1(x,y)] + k_2 H [I_2(x,y)], \quad (3.4.2-2)$$

onde k_1 e k_2 são constantes e $I_1(x,y)$ e $I_2(x,y)$ são duas imagens quaisquer. Esta propriedade é chamada de aditiva.

Para $I_2(x,y) = 0$, a Equação (3.4.2-2) reduz-se à $H [k_1 I_1(x,y)] = k_1 H [I_1(x,y)]$.

Isto equivale a dizer que a aplicação de um operador linear sobre o produto de uma imagem por uma constante é igual à constante multiplicada pelo resultado da operação. Esta propriedade é chamada de homogeneidade.

O operador é dito espacialmente invariante se $H [I(x-\alpha, y-\beta)] = g(x-\alpha, y-\beta)$ para quaisquer (x, y) , $I(x, y)$, α e β . Isto significa que o resultado da operação em um ponto depende apenas do valor do *pixel* e não da posição na imagem.

Pode-se expressar uma imagem $I(x, y)$ como uma integral da função delta de Dirac

$$I(x, y) = \iint I(\alpha, \beta) \delta(x-\alpha, y-\beta) d\alpha d\beta.$$

Para $n(x, y) = 0$ a Equação (3.4.2-1) \Rightarrow

$$g(x, y) = H I(x, y) = H \iint I(\alpha, \beta) \delta(x-\alpha, y-\beta) d\alpha d\beta.$$

Como H é linear e supondo que a propriedade aditiva se aplique à integração, vem:

$$g(x, y) = H I(x, y) = \iint H [I(\alpha, \beta)] \delta(x-\alpha, y-\beta) d\alpha d\beta.$$

Como $I(\alpha, \beta)$ não depende de x e y segue-se, da propriedade de homogeneidade, que

$$g(x, y) = \iint I(\alpha, \beta) H \delta(x-\alpha, y-\beta) d\alpha d\beta.$$

O termo $h(x, \alpha, y, \beta) = H \delta(x-\alpha, y-\beta)$ é chamado de resposta impulsiva do operador H ,

$$\text{ou seja, } g(x, y) = \iint I(\alpha, \beta) h(x, \alpha, y, \beta) d\alpha d\beta. \quad (3.4.2-3)$$

Esta equação é muito importante, pois estabelece que se a resposta impulsiva de um operador H for conhecida, então a resposta para uma função pode ser calculada pela Equação (3.4.2-3). Isto significa que um sistema linear pode ser caracterizado pela sua resposta impulsiva.

Como H é suposto ser espacialmente invariante, tem-se que:

$$H \delta(x-\alpha, y-\beta) = h(x-\alpha, y-\beta).$$

Assim, a Equação (3.4.2-3) reduz-se à:

$$g(x,y) = \iint I(\alpha,\beta) h(x-\alpha,y-\beta) d\alpha d\beta.$$

No caso da presença de ruído aditivo,

$$g(x,y) = \iint I(\alpha,\beta) h(x-\alpha,y-\beta) d\alpha d\beta + n(x,y). \quad (3.4.2-4)$$

Muitos tipos de degradação podem ser aproximadas por processos lineares e espacialmente invariantes. Numa formulação discreta, a Equação (3.2.4-4) resulta em [GONZALEZ87]

$$g(x,y) = \sum_{i=1}^N \sum_{j=1}^M I(i,j) h(x-i,y-j) + n(x,y), \quad (3.2.4-5)$$

ou em forma matricial, $\mathbf{G} = \mathbf{I}\mathbf{H} + \mathbf{N}$.

Isto significa que, uma vez que o processo de degradação possa ser descrito por uma matriz \mathbf{H} , a inversão deste processo pode ser feita a partir da Equação (3.2.4-5), restaurando-se a matriz da imagem \mathbf{I} a partir da matriz \mathbf{G} da imagem degradada e de uma matriz ruído \mathbf{N} . A formulação discreta deste modelo está apresentada em detalhe em [GONZALEZ87].

A solução deste sistema de equações em termos de um modelo de rede neuronal é apresentada no Capítulo 4.

3.5 Conclusão

Neste capítulo, foram introduzidos os conceitos básicos de processamento digital de imagens necessários às aplicações neste trabalho. Mais especificamente, foram discutidos conceitos relativos à compressão, detecção de bordas e restauração de imagens. As definições que são utilizadas nos próximos capítulos e a motivação para aplicação de redes neuronais como uma ferramenta para solução destes problemas também foram colocadas.

Aplicação de redes neurais em PDI

São apresentadas aplicações do *perceptron* multicamadas na compressão de imagens e na detecção de bordas, do mapa auto-organizativo em quantização vetorial e da rede de Hopfield na detecção de bordas e na restauração de imagens.

4.1 Compressão de imagens por *perceptron* multicamadas

Inicialmente, é apresentado um modelo de compressão por *perceptron* multicamadas que está baseado em [COTTRELL87] e se aplica a imagens em tons de cinza. A seguir, é apresentada uma proposta do autor para compressão de imagens coloridas.

4.1.1 Compressão de imagens em níveis de cinza.

Como foi visto no Capítulo 3, a idéia da compressão de imagens resume-se em fazer uma transformação do espaço de *pixels* para um outro em que o número de *bytes* para armazenar as informações seja menor. Naturalmente, isto é feito com alguma perda. Como encontrar esta transformação é a questão fundamental. Esta tarefa fica a cargo da rede neuronal. São utilizadas duas redes neurais, uma rede de compressão e outra de descompressão.

A rede de compressão precisa ser treinada. Para isto, uma imagem é dividida em janelas quadradas de $K \times K$ *pixels*. Os valores dos níveis de cinza dos $K \cdot K$ *pixels* são introduzidos como dados de entrada para um *perceptron* de duas camadas (ver Figura 9a).

A entrada e a saída da rede têm o mesmo número de neurônios. A idéia é que os pesos das conexões da rede estejam ajustados de tal modo que a rede produza uma saída igual à entrada, isto é, o valor do sinal de saída v_j do neurônio j da camada de saída deve corresponder ao valor u_j do sinal de entrada do neurônio j da camada de entrada. Este sinal é o valor do nível de cinza do *pixel* correspondente na janela da imagem. Uma diferença entre a entrada e a saída da rede gera uma medida de erro que é retropropagada na rede, através do algoritmo de *backpropagation* (descrito no Capítulo 3). Esta propagação do erro implica num ajuste dos pesos. À medida que os pesos se ajustam, a rede aprende a reproduzir uma saída igual à entrada.

A compressão de imagens é um tipo de mapeamento idêntico. O mapeamento é feito através de um canal estreito, forçando a rede a desenvolver uma codificação

eficiente. Este canal estreito é uma camada escondida menor do que as camadas de saída e de entrada.

O ajuste dos pesos é feito para todas as janelas da imagem. Uma janela é introduzida e um pequeno ajuste é feito em função desta janela. Na seqüência, uma nova janela é apresentada e um novo ajuste é feito. Isto é feito para uma série de imagens. Percorrida a série uma vez, a seqüência é reiniciada. Esta é a fase de treinamento da rede.

Uma série de imagens deve ser apresentada à rede, de modo que ao final do treinamento a rede tenha aprendido a reproduzir uma saída igual à entrada para uma determinada classe de imagens.

Uma vez treinada, a rede pode ser utilizada na compressão, pois os valores dos pesos das suas conexões já estão ajustados e armazenados.

Para comprimir uma imagem, basta passá-la em janelas através da rede. Cada janela gera um conjunto de valores de saída nos neurônios da camada escondida. Estes valores representam a janela comprimida, sendo gerados e armazenados para todas as janelas da imagem.

Na descompressão, é utilizado um *perceptron* de uma camada de mesmo número de neurônios que a camada de saída da rede utilizada na compressão. São utilizadas tantas conexões por neurônio quantos são os neurônios da camada escondida da rede de compressão (ver Figura 9b). Os valores dos pesos destas conexões são os valores correspondentes na rede de compressão e que foram armazenados.

Os dados de entrada para a rede de descompressão são as saídas dos neurônios da camada escondida da rede de compressão.

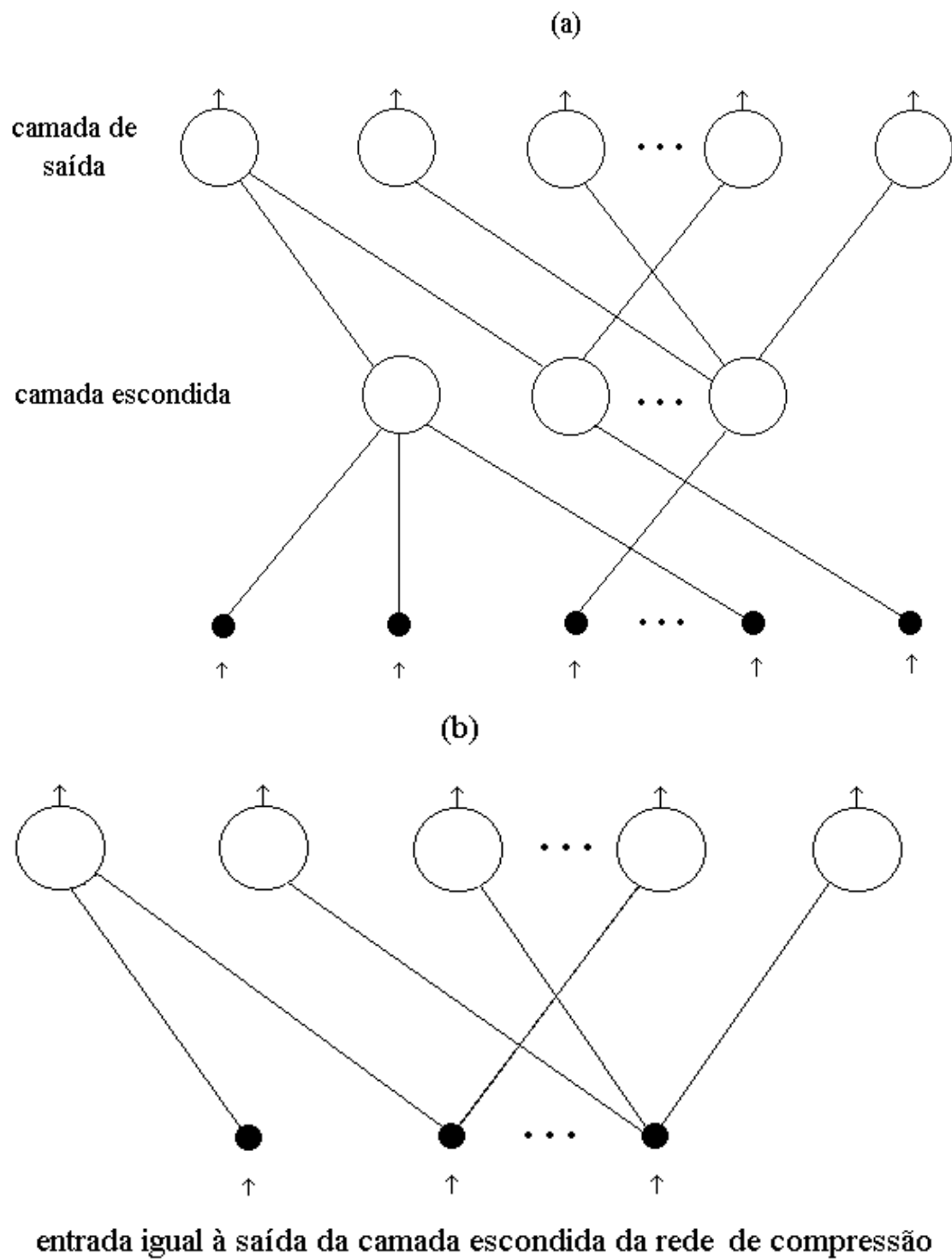


Figura 9. (a) Rede de compressão. (b) Rede de decompressão.(Fonte: [COTTRELL87])

4.1.1 Compressão de imagens coloridas

No caso de imagens coloridas, há três componentes de cores RGB. Uma alternativa é tratar cada componente independentemente, isto é, como se fosse nível de cinza. Mas neste caso, a compressão seria menor. Outra possibilidade é considerar os componentes conjuntamente. Os valores dos três componentes dos *pixels* de uma janela $K \times K$ são introduzidos em uma rede neuronal de $3 \cdot K \cdot K$ neurônios na entrada e $3 \cdot K \cdot K$ neurônios na saída. A segunda alternativa foi utilizada neste trabalho.

Os resultados obtidos inicialmente não foram satisfatórios, tanto em relação à qualidade da imagem quanto à taxa de compressão. Foram utilizadas janelas 3×3 , 4×4 e diversas variações no número de neurônios da camada escondida, mas os resultados obtidos foram pobres. Isto é consequência de que, nas imagens coloridas, o nível de informação é muito grande, sendo que pequenas variações em um componente podem significar uma grande variação a nível de detalhamento da imagem.

A alternativa encontrada pelo autor deste trabalho foi fazer um pré-processamento da imagem. Para isto, é calculada a média aritmética de cada componente RGB na janela $K \times K$. A seguir, para cada *pixel* da janela são calculadas as diferenças entre os seus componentes RGB e as respectivas médias. Estas diferenças são utilizadas como dados de entrada da rede neuronal.

Na descompressão, da mesma forma que nos níveis de cinza, é utilizado um *perceptron* de apenas uma camada. Os dados de entrada para esta rede são os valores de saída dos neurônios da camada escondida do *perceptron* de duas camadas utilizado na compressão. Os pesos das conexões são os correspondentes no *perceptron* de duas camadas.

Os valores são introduzidos nesta segunda rede e a saída obtida é uma aproximação para as diferenças entre os valores dos componentes dos pixels e os respectivos valores médios na janela. Para reproduzir o valor dos *pixels* são necessárias as médias dos componentes RGB. Estas, portanto, também precisam ser armazenadas.

Desta forma, a compressão de imagens coloridas passa a ser um refinamento do algoritmo de compressão por blocos que utiliza apenas o valor médio [JAIN89].

4.2. Quantização vetorial

Quantização vetorial é uma forma de codificação estatística [KRISHNAMURTHY90]. Um vetor \mathbf{x} de dimensão k que deve ser codificado é representado por um dos símbolos de um conjunto de M símbolos. Associado a cada símbolo está um vetor \mathbf{c} de dimensão k chamado de código. O conjunto completo de M códigos é chamado de livro código. Os k valores podem ser, por exemplo, amostras de um sinal ou parâmetros extraídos do sinal. O livro código $C = \{ \mathbf{c}_i, i = 1, \dots, M \}$ é usualmente obtido através de um processo de treinamento usando-se um grande conjunto de dados de treinamento que sejam representativos daqueles que serão encontrados na prática.

Durante o processo de codificação, o vetor \mathbf{x} de dimensão k é comparado a cada um dos M códigos e o desvio $d(\mathbf{x}, \mathbf{c}_i)$, $i = 1, \dots, M$ entre o vetor e o código é calculado. O vetor \mathbf{x} é então codificado como o índice do código de menor desvio. De posse de uma cópia do livro código, é possível decodificar o vetor \mathbf{x} original a partir do índice armazenado e do código correspondente.

Formular o problema de quantização vetorial em termos de uma rede neuronal é bastante simples. Considere-se uma rede neuronal com M neurônios (ver Figura 7 na Seção 2.3) e associado com cada neurônio i um vetor de pesos que seja o i -ésimo código, isto é, $\mathbf{w}_i = \mathbf{c}_i$. Qualquer vetor \mathbf{x} a ser codificado é introduzido em paralelo para os M neurônios. Cada um calcula o desvio $d(\mathbf{x}, \mathbf{w}_i)$. A unidade com menor desvio é declarada vencedora e \mathbf{x} é codificado pelo índice desta unidade. Interpretando-se este desvio como a distância descrita na Seção 2.3, as ferramentas ali apresentadas podem ser usadas para o treinamento.

Assim, podem ser utilizados os algoritmos para aprendizado do mapa organizativo de Kohonen ou da rede para aprendizado competitivo sensível à frequência. Ambos os algoritmos partem de valores aleatórios pequenos para os pesos. Os algoritmos ajustam os pesos a cada novo vetor \mathbf{x} de entrada introduzido.

No caso de imagens em tons de cinza, os valores dos componentes do vetor \mathbf{x} consistem nos níveis de cinza dos *pixels* de uma janela $K \times K$. No caso de imagens coloridas, da mesma forma que na compressão por *perceptron* multicamadas, utilizou-se o artifício de subtrair o valor médio por componente de cor RGB. O vetor \mathbf{x} , neste caso, terá $3 \cdot K \cdot K$ componentes por janela, correspondendo aos três componentes de cores RGB.

Novamente, poderia ser feito um tratamento por componente separadamente, isto é, para cada janela haveria três índices no livro código. Cada componente geraria um desvio e um índice. O livro código teria três conjuntos de pesos e códigos, um para cada componente.

A alternativa adotada neste trabalho foi calcular um desvio total dos três componentes. O livro código continua, então, a ter apenas um índice por janela.

4.3 Detecção de bordas por *perceptron* multicamadas

A idéia básica do algoritmo é definir o que seja uma borda através de padrões de borda e treinar uma rede neuronal para identificar estes padrões. Este é o tipo de estratégia adotada em reconhecimento de caracteres apresentada na Seção 2.4.

Na verdade, o que normalmente é chamado de detecção de borda é apenas detecção de variação de nível de cinza através de um limiar. Seria mais correto denominar o modelo aqui proposto de detecção de contorno, uma vez que o contorno está relacionado com a idéia de continuidade e também com a variação do nível de cinza. Com a estratégia aqui proposta pretende-se um efeito de eliminação de ruído.

Neste trabalho, foi adotado um conceito de borda baseado numa janela 3×3 e padrões de *bits* 0 e 1 nestas janelas. Estes padrões são mostrados na Figura 10 e representam o que seja uma borda binária. Eles, juntamente com padrões binários do que não seja uma borda, foram utilizados para treinar uma rede a reconhecer o que seja uma borda. Esta representação de padrões de borda é inspirada na representação de caracteres presente em [SCHALKOFF89] e que foi discutida na Seção 2.4.

Uma janela 3×3 binarizada passa pela rede neuronal. Caso a saída do neurônio da camada de saída seja 1, o *pixel* central da janela pertence a uma borda, caso contrário não.

Como as imagens neste trabalho são coloridas de 24 *bits*, é preciso que cada janela 3×3 da imagem gere uma janela 3×3 binária para que a rede neuronal a identifique como sendo ou não uma borda.

No caso de imagens em níveis de cinza, isto foi feito utilizando-se novamente o recurso da média. Numa janela 3×3 em níveis de cinza, o valor médio dos *pixels* é calculado. Os *pixels* cujas diferenças em relação à média estejam acima de um limiar assumem o valor 1, aqueles abaixo da média assumem o valor 0. Assim, a janela está binarizada.

No caso de imagens coloridas, isto é feito por componente. Se em pelo menos um componente a diferença for maior do que o limiar, o valor atribuído é 1, caso contrário é 0.

A rede neuronal utilizada tem 9 neurônios na camada de entrada (correspondendo aos *pixels* da janela 3×3), 3 na escondida e 1 na camada de saída.

Uma imagem deve ser percorrida em janelas 3×3 *pixel a pixel*. Estas janelas são binarizadas e passadas pela rede neuronal. O reconhecimento de borda é feito janela por janela e o valor do *pixel* central é feito igual a 255 ou 0 conforme o caso. Isto produz o conhecido mapa de bordas.

Observa-se que o importante é corresponder ou não a um padrão de borda, não importa qual.

Uma alternativa seria fazer com que a rede neuronal indentificasse não só uma borda mas o seu tipo. Isto pode ser feito atribuindo-se outros valores à saída do neurônio da camada de saída e fazendo-se a correspondência entre eles e o tipo da borda: borda diagonal para direita, diagonal para esquerda, horizontal, etc. Uma vez feito isto, o mapa de bordas poderia ser gerado através de segmentos de borda retilíneos, que seriam traçados na janela conforme o tipo da borda.

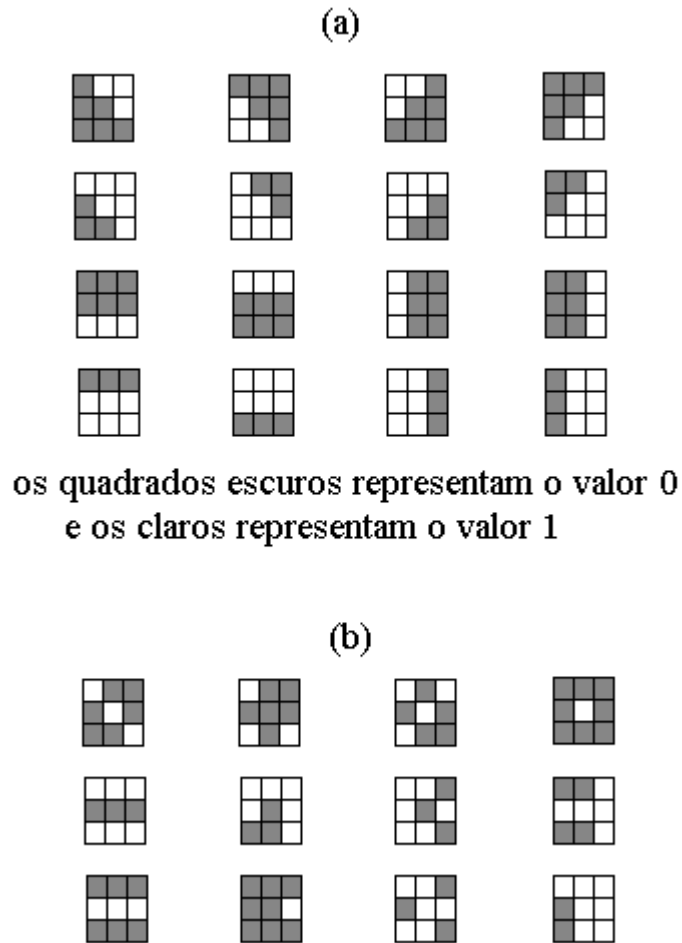


Figura 10. (a) Padrões de borda. (b) Exemplos de padrões que não representam borda.

4.4 Detecção de bordas por rede de Hopfield

Outra forma para identificação de bordas utilizando-se o modelo de Hopfield é discutida a seguir. Este modelo não foi implementado, apenas os seus conceitos são apresentados.

O modelo baseia-se numa aplicação visual de memória associativa [JOSIN87]. Considere-se o conjunto de padrões da Figura 10(a). São janelas 3x3 que representam padrões de bordas. Pode-se associar cada *pixel* da janela a um neurônio e ainda acrescentar-se mais 16 neurônios para representar cada categoria de borda. Isto pode ser visto na Figura 11.

Na Figura 11, a linha número 4 categoriza o padrão. O padrão no topo à esquerda tem o valor 1 na primeira coluna da linha número 4, enquanto que o padrão imediatamente ao lado tem o valor 1 na segunda coluna da mesma linha. Este primeiro padrão caracteriza uma borda diagonal de baixo para cima, da direita para esquerda. O segundo caracteriza uma borda da esquerda para direita, de baixo para cima. Os demais padrões estão categorizados de forma similar. Os padrões que não representam borda têm a última linha toda nula.

O modelo de memória associativa descrito na Seção 2.4.2 pode ser usado para codificar e decodificar estes padrões. Se cada neurônio assumir valor 0 ou 1, uma rede de Hopfield de 25 neurônios pode mapear este conjunto particular de padrões na forma de uma matriz de pesos. Assim, quando um novo padrão for apresentado, a rede recordará o padrão mais próximo. Se o padrão recordado tiver 1 em alguma coluna da linha número 4, o padrão corresponderá a uma borda (qualquer borda).

Recordando o Capítulo 2, quando a rede é totalmente interconectada, um neurônio recebe e envia sinais de e para todos os demais.

Os padrões da Figura 10a devem ser introduzidos de modo a permitir o cálculo da matriz de pesos. Padrões corrompidos, isto é, que não correspondem a bordas (linha número 4 toda nula), também devem ser fornecidos. A partir da Equação (2.4.2-3) os pesos das conexões são determinados. Feito isto, a rede está apta a detectar uma borda.

Novamente, deve-se transformar uma janela de uma imagem em padrão binário pelo mesmo processo descrito na Seção 4.3. Este padrão binário é introduzido na rede. Esta entra em iteração assíncrona até atingir um estado estável. Os valores da linha número 4 são então observados para verificar se a janela corresponde ou não a uma borda. Em caso positivo, o *pixel* central da janela é feito igual a 255, caso contrário é feito igual a 0.

A imagem deve ser percorrida *pixel a pixel* com janelas 3x3 definidas ao redor de cada um. Este procedimento também gera um mapa de bordas. O resultado a ser obtido deve ser equivalente ao da detecção de bordas por *perceptron* multicamadas, uma vez que o conceito de borda é o mesmo.

Da mesma forma que no caso anterior, o mapa de bordas poderia ser formado a partir de segmentos de bordas retilíneas, através da identificação não só de uma borda mas do tipo da borda. Isto poderia ser feito pelo número da coluna em que o valor do neurônio da linha 4 assume o valor 1.

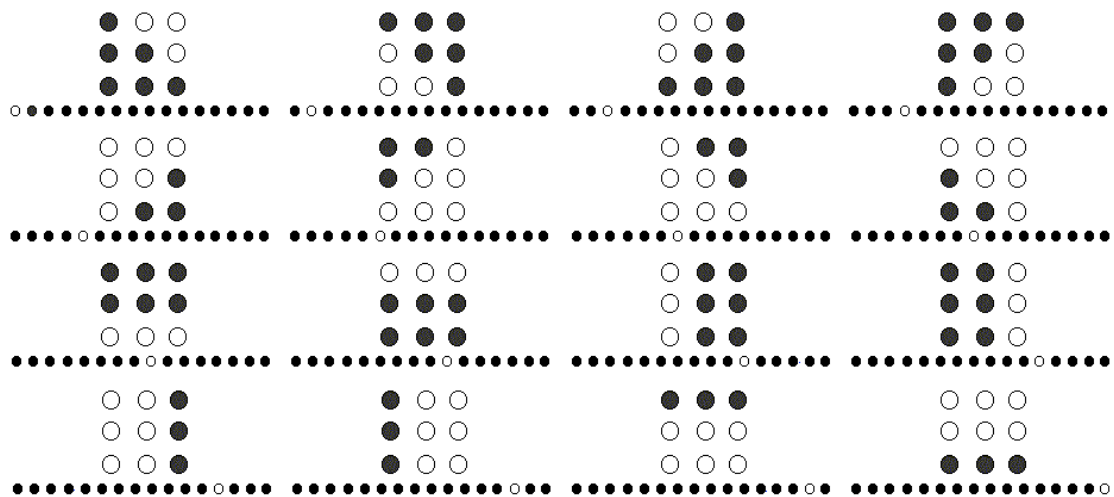


Figura 11. Estados da rede de Hopfield para detecção de borda. Os círculos representam neurônios, os quais estão inteiramente conectados da forma mostrada na Figura 4. (Fonte: [JOSIN87])

4.5 Restauração de imagens por rede de Hopfield

O modelo discutido aqui está descrito em [KOSKO92-1] e se aplica a imagens em tons de cinza. Este modelo não foi implementado, apenas os seus conceitos são apresentados, de modo a demonstrar a determinação dos pesos das conexões em um problema de otimização.

4.5.1 Representação da imagem

O modelo a ser utilizado é o modelo de Hopfield discreto, descrito na Seção 2.4. Desta forma, é necessário fazer uma representação binária de uma imagem em níveis de cinza.

Para isto, são usados neurônios agrupados para representar os níveis de cinza. Naturalmente, isto exigirá um número enorme de neurônios. O modelo consiste em $L \cdot L \cdot M$ neurônios totalmente interconectados, onde $L \cdot L$ representa o tamanho da imagem (supondo-se uma imagem quadrada) e M o valor máximo do nível de cinza (255 por exemplo). A imagem é descrita como um conjunto de funções de nível de cinza $\{x(i,j)$, onde $i,j \leq L\}$, com $x(i,j)$ (inteiros positivos) representando o nível de cinza do *pixel* (i,j) . Seja $V = \{v_{ik} \mid 1 \leq i \leq L^2, 0 \leq k \leq M\}$ um estado binário da rede neuronal com v representando o estado do neurônio (i,k) . A função de nível de cinza pode ser representada por uma soma dos estados dos neurônios como

$$x(i,j) = \sum_{k=0}^M v_{m,k} \quad (4.5.1-1)$$

onde $m = (i-1)L + j$.

As funções de níveis de cinza têm repetições. Por exemplo, se uma função de nível de cinza é representada por M neurônios e o valor é 10, então qualquer conjunto de 10 neurônios que assuma o valor 1 satisfaz. Isto significa que há $M!/(10!(M-10)!)$ configurações possíveis.

É interessante notar que o fato de um neurônio ter um valor errado não ocasiona um erro muito grande. Uma rede neuronal usando tal representação tem

estados estáveis para uma imagem $L \times L$. Um número tão grande de estados possibilita à rede mais facilidade de estabilizar numa solução correta.

Neste modelo, cada neurônio (i,k) recebe sinais aleatória e assincronamente de todos os demais e um sinal de *bias*.

$$u_{i,k} = \sum_{j=1}^{L:L} \sum_{l=0}^M w_{i,k;j,l} v_{j,l} + \lambda_{i,k} \quad (4.5.1-2)$$

onde $w_{i,k;j,l}$ corresponde ao peso da conexão entre os neurônios (i,k) e (j,l) e $\lambda_{i,k}$ é o termo de *bias*. Os pesos são simétricos, isto é,

$$w_{i,k;j,l} = w_{i,l;i,k} \quad \text{e} \quad w_{i,k;i,k} \neq 0.$$

Isto significa que as conexões são simétricas e os neurônios têm realimentação. Cada $u_{i,k}$ é realimentado ao neurônio correspondente após truncamento por limiar

$$v_{i,k} = f(u_{i,k}), \quad (4.5.1-3)$$

onde $f(u_{i,k})$ é uma função não linear da forma

$$f(u) = \begin{cases} 1 & \text{se } u \geq 0 \\ 0 & \text{se } u < 0 \end{cases}$$

O estado de um neurônio é atualizado utilizando o último estado disponível dos demais neurônios.

4.5.2 Estimativa dos parâmetros do modelo

Os parâmetros do modelo neuronal, os pesos das conexões e termos de *bias*, podem ser determinados em termos da função de energia da rede neuronal. Como foi visto no Capítulo 2, a partir da Equação (2.4.2-1) pode-se escrever esta função como:

$$E = -\frac{1}{2} \sum_{i=1}^{L:L} \sum_{j=1}^{L:L} \sum_{k=0}^M \sum_{l=0}^M w_{i,k;j,l} v_{i,k} v_{j,l} - \sum_{i=1}^{L:L} \sum_{l=0}^M \lambda_{i,k} v_{i,k} \quad (4.5.2-1)$$

Com a finalidade de minimizar a energia da rede neuronal, o problema de restauração de imagem é reformulado como sendo um problema de minimização de uma função erro com restrições, definida por [KOSKO92-1]

$$E = \frac{1}{2} \|Y - H X\theta\|^2 + \frac{1}{2} \mu \|DX\theta\|^2, \quad (4.5.2-2)$$

onde $\|\dots\|$ é o módulo num espaço $L \times L$;

μ é uma constante;

Y é a imagem degradada;

$X\theta$ é a aproximação para imagem original que se deseja obter;

H é o operador do processo de degradação, conforme descrito na Seção 3.4;.

D é um filtro de suavização.

Uma medida de erro desta forma é largamente usada em problemas de restauração de imagens [GONZALEZ87]. O primeiro termo corresponde a procurar $X\theta$ tal que $HX\theta$ seja uma aproximação por mínimos quadrados de Y . O segundo é um termo de suavização. A constante μ determina a importância relativa de ambos.

Em geral, se H é uma distorção passa-baixas, D é um filtro passa-altas. Uma escolha bastante comum para D é um operador diferencial de segunda ordem que pode ser aproximado por um operador de vizinhança no caso discreto. Por exemplo, um operador Laplaciano

$$\frac{1}{6} \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

pode ser aproximado por operador de vizinhança [KOSKO92-1]

$$\text{isto é, } d(k,l) = \begin{cases} -\frac{10}{3} & \text{se } (k,l) = (0,0); \\ \frac{2}{3} & \text{se } (k,l) = \{(-1,0), (0,-1), (0,1), (1,0)\}; \\ \frac{1}{6} & \text{se } (k,l) = \{(-1,-1), (-1,1), (1,-1), (1,1)\}. \end{cases}$$

Expandindo a Equação (4.5.2-2) e substituindo x_i pela Equação (4.5.1-1) vem:

$$\begin{aligned} E &= \frac{1}{2} \sum_{p=1}^{L:L} (y_p - \sum_{i=1}^{L:L} h_{p,i} x_i)^2 + \frac{1}{2} \mu \sum_{p=1}^{L:L} (\sum_{i=1}^{L:L} d_{p,i} x_i)^2 \\ &= \frac{1}{2} h_{p,i} h_{p,j} v_{i,k} v_{j,l} + \frac{1}{2} \mu d_{p,i} d_{p,j} v_{i,k} v_{j,l} \\ &\quad - \sum_{i=1}^{L:L} \sum_{k=0}^M \sum_{p=1}^{L:L} y_p h_{p,i} v_{i,k} + \frac{1}{2} \sum_{p=1}^{L:L} y_p^2. \end{aligned} \quad (4.5.2-3)$$

Comparando-se os termos da Equação (4.5.2-3) com os correspondentes da Equação (4.5.2-1) e ignorando o termo constante

$$\frac{1}{2} \sum_{p=1}^{L:L} y_p^2,$$

pode-se determinar os pesos das conexões e os termos de *bias* por

$$w_{i,k;j,l} = - \sum_{p=1}^{L:L} h_{p,i} h_{p,j} - \mu \sum_{p=1}^{L:L} d_{p,j} d_{p,i} \quad (4.5.2-4)$$

e

$$\lambda_{i,k} = \sum_{p=1}^{L:L} y_p h_{p,i} \quad (4.5.2-5)$$

onde $h_{i,j}$ e $d_{i,j}$ são os elementos das matrizes H e D , respectivamente.

Dois aspectos interessantes das Equações (4.5.2-4) e (4.5.2-5) devem ser destacados: (1) os pesos das conexões são independentes dos subscritos k e l e os termos de *bias* são independentes de k ; (2) há realimentação dos neurônios.

Da Equação (4.5.2-4), pode-se ver que os pesos das conexões são determinados pela função de turvamento espacialmente invariante, pelo operador diferencial e pela constante μ . Desta forma, os pesos podem ser calculados sem erro, desde que a função de turvamento seja conhecida. Por outro lado, os termos de *bias* são função da imagem degradada. Se a imagem for degradada por uma função de turvamento espacialmente invariante, $\lambda_{i,k}$ pode ser perfeitamente estimado. Caso contrário, $\lambda_{i,k}$ é afetado pelo ruído. Isto pode ser compreendido a partir da Equação (3.2.4-5), em que a matriz da imagem degradada \mathbf{G} linearizada passa a ser o vetor \mathbf{Y} e a matriz da imagem original \mathbf{I} linearizada passa a ser o vetor \mathbf{X} . Substituindo-se y_p por

$$\sum_{j=1}^{L:L} h_{p,j} x_j + n_p,$$

tem-se:

$$\sum_{p=1}^{L:L} \left(\sum_{j=1}^{L:L} h_{p,j} x_j + n_p \right) h_{p,i} = \sum_{p=1}^{L:L} \sum_{j=1}^{L:L} h_{p,j} x_j h_p + \sum_{p=1}^{L:L} n_p h_{p,i} \quad (4.5.2-6)$$

O segundo termo da Equação (4.5.2-5) representa o efeito do ruído. Se a relação sinal/ruído SNR (definida Seção 3.2) for baixa, deve-se escolher um valor alto para μ . Na ausência de ruído, os parâmetros podem ser estimados perfeitamente, assegurando a restituição exata da imagem à medida que a energia E tende a zero. No entanto, o problema não é tão simples, já que o desempenho da restauração depende tanto do parâmetro μ quanto da função de turvamento, quando um erro quadrático como o dado pela Equação (4.5.2-2) é usado. Este efeito será discutido a seguir.

4.5.3 Restauração

A restauração da imagem é feita pela avaliação do estado dos neurônios e pela reconstituição da imagem a partir deles. Uma vez que $w_{i,k;j,l}$ e $\lambda_{i,k}$ sejam calculados a partir das Equações (4.5.2-4) e (4.5.2-5), cada neurônio pode avaliar aleatoriamente e assincronamente o seu estado e reajustá-lo de acordo com as Equações (4.5.1-2) e (4.5.1-3). Quando um ponto de energia quase mínima for atingido, a imagem pode ser reconstituída pela Equação (4.5.1-1).

Contudo, esta rede neuronal tem realimentação, isto é, $w_{i,k;i,k} \neq 0$. Em consequência, a energia E nem sempre decresce monotonicamente. Isto pode ser visto definindo-se a mudança de estado do neurônio (i,k) como $\Delta v_{i,k}$ e a variação na energia ΔE como

$$\Delta v_{i,k} = v2_{i,k} - v1_{i,k} \quad \text{e} \quad \Delta E = E2 - E1.$$

A partir da Equação (4.5.2-1), a variação ΔE devido à $\Delta v_{i,k}$ pode ser calculada por

$$\Delta E = - \left(\sum_{i=1}^{L:L} \sum_{l=0}^M w_{i,k;j,l} v_{j,l} + I_{i,k} \right) \Delta v_{i,k} - w_{i,k;i,k} v_{i,k} \Delta(v_{i,k})^2, \quad (4.5.3-1)$$

que nem sempre é negativa. Por exemplo, se

$$v_{i,k} = 0; u_{i,k} = \sum_{i=1}^{L:L} \sum_{l=0}^M w_{i,k;j,l} v_{j,l} + I_{i,k} > 0$$

e, considerando-se a função de limiar dada por (4.5.1-3), que $v_{i,k} = 1$ e $\Delta v_{i,k} > 0$, o primeiro termo na Equação (4.5.3-1) é negativo. Entretanto,

$$w_{i,k;i,k} = - \sum_{p=1}^{L:L} (h_{p,i} h_{p,i}) - \lambda \sum_{p=1}^{L:L} (d_{p,i} d_{p,i}) < 0.$$

Para $\mu > 0$ isto conduz à $-w_{i,k;i,k} v_{i,k} \Delta(v_{i,k})^2 > 0$.

Quando o primeiro termo for menor do que o segundo em (4.5.3-1), tem-se $\Delta E > 0$, o que significa que E não é uma função de Lyapunov (definida no Capítulo 2). Ou seja, a convergência da rede não está garantida.

Uma regra de decisão para atualização do estado dos neurônios deve então ser arbitrada. Por exemplo, uma regra determinística de se atualizar o estado de um neurônio (i,k) de $v_{i,k}$ para $v_{i,k}$ apenas se a variação ΔE na energia E devido à $\Delta v_{i,k}$ for negativa; se for positiva mantém-se o estado. Pode-se, também, adotar um procedimento estocástico [KOSKO92-1].

Em resumo, o algoritmo de restauração pode ser descrito como:

passo 1. inicializar os neurônios;

passo 2. atualizar os neurônios aleatória e assincronamente pela regra de decisão;

passo 3. se variação da função de energia for suficientemente pequena, ir para passo 4, caso contrário retornar ao passo 2;

passo 4. construir uma imagem a partir da Equação (4.5.1-1).

4.5.4 Algoritmo prático

O algoritmo descrito na Seção anterior é de difícil implementação, devido à complexidade, mesmo para imagens de tamanho razoável. Por exemplo, uma imagem $L \times L$ com M níveis de cinza precisa de $L^2 \cdot M$ neurônios e $0,5 L^4 \cdot M^2$ conexões. Além disso são necessárias da ordem de $L^4 \cdot M^2$ operações de soma e multiplicação em cada iteração. Portanto, a complexidade é da ordem de $O(L^4 \cdot M^2)$ e $O(L^4 \cdot M^2 \cdot K)$, sendo K o número de iterações. Frequentemente, L e M são da ordem de 512 e 256, respectivamente. Contudo, é possível simplificar o algoritmo desde que os neurônios sejam atualizados seqüencialmente.

De modo a simplificar o algoritmo, considere-se as Equações (4.5.1-2) e (4.5.2-3). Como já foi destacado, os pesos $w_{i,k;j,l}$ na Equação (4.5.2-4) são independentes dos índices k e l e os termos de *bias* $\lambda_{i,k}$ são independentes do índice k , uma vez que os M neurônios usados para representar o mesmo nível de cinza têm os mesmos termos de *bias*. Isto significa que as dimensões das matrizes de pesos e de *bias* podem ser reduzidas de um fator M^2 .

A partir da Equação (4.5.1-2), todos os sinais de entrada recebidos por um neurônio (i,k) podem ser escritos por

$$u_{i,k} = \sum_{j=1}^{L:L} (w_{i,-;j,-} \sum_{j=1}^{L:L} \sum_{l=0}^M v_{j,l}) + I_{i,-} = \sum_{j=1}^{L:L} w_{i,-;j,-} + x_j I_{i,-}, \quad (4.5.4-1)$$

onde x_j é o nível de cinza do j -ésimo *pixel* da imagem. O símbolo "-" nos índices de $w_{i,-;j,-}$ e $I_{i,-}$ significa que estes parâmetros são independentes de k . Se os neurônios forem atualizados seqüencialmente, o algoritmo de restauração pode ser reformulado como:

passo 1. inicializar os neurônios;

passo 2. visitar seqüencialmente os *pixels* da imagem; para cada *pixel* i atualizar seqüencialmente os neurônios (i,k) correspondentes aos seus níveis de cinza;

passo 3. verificar a função de energia, se a sua variação for suficientemente pequena ir para passo 4, caso contrário retornar ao passo 2;

passo 4. construir a imagem a partir da Equação (4.5.1-1).

As Figuras 12,13 e 15 ilustram a aplicação deste algoritmo. A Figura 12 apresenta a imagem original, enquanto a Figura 13 apresenta a imagem degradada por uma função de turvamento uniforme $h(i+k,j+l) = 1 / 25$, para $|k|$ e $|l| \leq 2$, aplicada em torno de cada *pixel* (i,j) da imagem. Na Figura 15, está a imagem restaurada pelo filtro neuronal e na Figura 14, está a imagem restaurada por um filtro inverso [KOSKO92-1]. Esta figuras foram retiradas de [KOSKO92-1].

O procedimento apresentado não foi implementado, e tem apenas a finalidade de ilustrar a abordagem do processo de restauração digital de imagens como um processo de otimização, expresso na Equação (4.5.2-2), e sua solução através de uma rede neuronal do tipo Hopfield. Este procedimento de restauração se aplica a imagens em níveis de cinza, degradadas por uma função de turvamento espacialmente invariante e ruído aditivo. É um procedimento que não envolve inversão de matrizes e pode ser aplicado em imagens de tamanho razoável.

Deve-se destacar, que o mínimo local atingido pode não ser o melhor, em função do problema que está sendo tratado. Neste caso, pode-se tentar encontrar outros mínimos locais na superfície definida pela energia E . Para isto, deve-se deslocar a rede neuronal do estado atingido e continuar a iteração.

4.6 Conclusão

Neste capítulo, foram apresentados três paradigmas de redes neuronais aplicados em PDI:

- *perceptron* multicamadas em compressão de imagens e detecção de bordas;
- mapa auto-organizativo em compressão de imagens por quantização vetorial;
- rede de Hopfield em detecção de bordas e restauração de imagens.

O *perceptron* multicamadas é um exemplo de aplicação de aprendizado supervisionado. Neste capítulo, foi apresentada a formulação descrita em [COTTRELL87] para compressão de imagens em níveis de cinza. O autor deste trabalho propôs uma extensão para imagens coloridas de 24 *bits*. A implementação deste algoritmo para compressão de imagens é discutida no próximo capítulo.

Foi discutida uma aplicação do *perceptron* multicamadas para detecção de bordas através de um modelo de borda. A sua implementação é apresentada no próximo capítulo.

O mapa auto-organizativo é um exemplo de aprendizado não supervisionado. Foi introduzida a formulação para compressão de imagens em níveis de cinza presente em [KRISHNAMURTHY90] e uma extensão para compressão de imagens coloridas de 24 *bits*. A implementação deste algoritmo é apresentada no próximo capítulo.

Finalmente, foi discutida a aplicação de redes com realimentação total na forma da aplicação do paradigma de Hopfield em restauração de imagens em níveis de cinza, segundo o modelo apresentado em [KOSKO92-1]. A sua aplicação em detecção de bordas é uma extensão de um modelo de memória associativa presente em [JOSIN87].



Figura 12. Imagem original. (Fonte: [KOSKO92-1])



Figura 13. Imagem degradada por turvamento uniforme e ruído de quantização.
(Fonte: [KOSKO92-1])



Figura 14. Imagem restaurada por filtro inverso. (Fonte: [KOSKO92-1])



Figura 15. Imagem restaurada pela rede neuronal. (Fonte: [KOSKO92-1])

Implementação dos algoritmos

5.1 Introdução

A título de demonstração, foram implementados o *perceptron* multicamadas para compressão de imagens e detecção de bordas e o mapa auto-organizativo para compressão de imagens.

Foi utilizada a linguagem C, versão Borland C 3.1, em ambiente DOS 5.0. Foi utilizado um PC-AT 486 de 50MHz com placa de vídeo de 1Mb, monitor SVGA colorido de 1024x768 e 256 níveis de cinza, 8Mb de memória RAM e disco rígido de 200MB. Estes recursos estão disponíveis no Centro de Desenvolvimento da Tecnologia Nuclear/Comissão Nacional de Energia Nuclear.

Os algoritmos foram implementados usando-se apenas recursos padrões da linguagem C, evitando-se ao máximo extensões da linguagem, tendo em vista a sua portabilidade.

Os algoritmos processam a imagem a partir do vídeo. Uma imagem é carregada e trabalhada a partir da tela como se fosse uma matriz. Para isto, foi utilizada a biblioteca SVGA256.LIB do sistema Pixelware [DAVIS92] para processamento das imagens.

Os algoritmos foram implementados para processar imagens coloridas de 24 *bits* RGB, mesmo que os recursos disponíveis só trabalhem com imagens em cores mapeadas por *palettes*. As imagens em tons de cinza são simplesmente imagens de 24 *bits* com os três componentes iguais.

As imagens são processadas por janelas, e o acerto das janelas nas bordas da imagem é feito acrescentando-se janelas adicionais superpostas para completar a imagem, quando isto for necessário.

A seguir, é descrita a implementação destes algoritmos e são apresentados alguns resultados da sua utilização.

[

5.2 *Perceptron* multicamadas para compressão de imagens

5.2.1 Implementação

O *perceptron* multicamadas foi implementado com duas camadas, uma intermediária ou escondida e outra de saída [COTTRELL87]. O número de neurônios por camadas é arbitrário, sendo fornecido como dados de entrada. No entanto, como já foi visto no Capítulo 3, as camadas de entrada e saída devem ter o mesmo número de neurônios, e estes correspondem diretamente aos *pixels* de uma janela da imagem a ser processada. A imagem é percorrida por janelas não superpostas, e os sinais de entrada da rede são os valores dos *pixels*.

Conforme já foi visto no Capítulo 3, uma imagem colorida utiliza três vezes mais neurônios nas camadas de entrada e de saída do que uma imagem em tons de cinza.

Os dados de entrada da rede são os valores dos componentes RGB da imagem. Estes valores variam de 0 a 255. A rede trabalha internamente em ponto flutuante com valores entre -1,0 e +1,0. Assim, os valores dos *pixels* são convertidos linearmente para este intervalo através de um fator de escala. Na saída da rede, os valores são novamente reconvertidos para o intervalo [0, 255].

No caso de imagens em tons de cinza, apenas um componente é processado. No caso de imagens coloridas, os três componentes RGB são efetivamente processados.

A rede neuronal pode trabalhar em três modos diferentes:

- aprendizado;
- compressão;
- descompressão.

Toda rede neuronal de aprendizado supervisionado precisa sofrer um processo de treinamento para ajustar os pesos das suas conexões. Este processo de treinamento pode ser resumido nos seguintes passos:

passo 1. dimensionar a rede neuronal, introduzir constantes de aprendizado, fator de escala dos pesos e coeficiente da função de transferência;

passo 2. inicializar os pesos com valores aleatórios entre -1,0 e +1,0;

passo 3. inicializar erro total com zero. Se necessário, atualizar constantes de aprendizado;

passo 4. carregar nova imagem, se não houver mais imagens, ir ao passo 11;

- passo 5.** ler janela da imagem; ao chegar ao fim da imagem, retornar ao passo 4;
- passo 6.** passar valores pela rede neuronal;
- passo 7.** calcular erro de saída, conforme descrito na Seção 2.1;
- passo 8.** propagar erro na rede conforme descrito na Seção 2.1, ajustando-se os pesos;
- passo 9.** calcular erro de saída acumulado para todas as janelas e para todas as imagens;
- passo 10.** retornar ao passo 4;
- passo 11.** se erro total acumulado for suficientemente pequeno, guardar dados da rede e encerrar o processamento, caso contrário reinicializar lista de imagens e retornar ao passo 3.

No dimensionamento da rede neuronal, o usuário indica o tamanho da janela a ser utilizada, o que automaticamente define o número de neurônios das camadas de entrada e de saída. A camada intermediária deve também ser definida. Este dimensionamento é muito importante, pois é ele que define a taxa de compressão a ser obtida. A rede neuronal deve aprender a mapear a entrada na saída, passando por um canal estreito que é a camada intermediária. Isto significa que o número de neurônios desta camada deve ser bem menor do que nas camadas de entrada e saída.

A taxa de aprendizado e o coeficiente de *momentum* (definido na Seção 2.2.2) podem ser atualizados a cada etapa do treinamento, de modo a acelerá-lo ou contornar problemas de convergência.

Os valores dos pesos podem ser dimensionados através de um fator de escala de modo que assumam valores menores, entre $-0,1$ e $+0,1$, por exemplo. Isto pode ter influência na convergência.

A função de transferência que foi utilizada é uma função exponencial da forma [COTTRELL87]:

$$v = f(u) = \frac{1}{1 + e^{-\alpha u}},$$

onde u = entrada total do neurônio;

v = saída do neurônio;

α = coeficiente dado pelo usuário como dado de entrada.

Para servir de referência para o usuário, existem valores *default* para todos os dados de entrada.

Uma vez completado o treinamento, a rede está apta para as etapas de compressão e descompressão de imagens. Os dados da rede, como dimensões e valores dos pesos, são guardados em arquivo de dados para posterior utilização.

O procedimento de compressão pode ser resumido por:

passo 1. inicializar a rede neuronal a partir de arquivo gerado no treinamento;

passo 2. carregar imagem;

passo 3. ler nova janela; se for fim da imagem, ir para passo 6;

passo 4. passar janela pela rede neuronal;

passo 5. guardar em arquivo as saída dos neurônios da camada escondida e retornar ao passo 3;

passo 6. fim.

No passo 5, a saída dos neurônios da camada intermediária são guardadas em arquivo. No final da compressão, o arquivo gerado desta forma representa a imagem comprimida. Isto decorre do fato de que os neurônios da camada escondida são em menor número.

No entanto, isto apenas não basta. Como foi dito, internamente a rede neuronal trabalha em ponto flutuante no intervalo $-1,0$ a $1,0$, ao passo que os pontos da imagem são representados por números inteiros que ocupam um *byte* cada. Desta forma, foi necessário fazer um mapeamento do intervalo $-1,0$ a $+1,0$ para o intervalo -128 a $+127$, através da divisão em 256 intervalos iguais [COTTRELL87].

No procedimento de descompressão, é utilizada uma rede neuronal de apenas duas camadas. A camada de entrada é a camada intermediária da rede de compressão e a camada de saída é a correspondente da rede de compressão. Este procedimento compreende os seguintes passos:

passo 1. inicializar a rede neuronal a partir do arquivo de treinamento;

passo 2. ler janela comprimida no arquivo comprimido; se for fim de arquivo, ir para passo 6;

passo 3. fazer mapeamento inverso dos valores da janela do intervalo -128 a $+127$ para o intervalo $-1,0$ a $+1,0$;

passo 4. passar valores pela rede neuronal;

passo 5. converter valores de saída da rede do intervalo $-1,0$ a $+1,0$ para o intervalo 0 a 255 através do fator de escala; mostrar janela no vídeo e retornar ao passo 2;

passo 6. fim.

A medida da compressão obtida é o número de *bits* por *pixel* utilizado no arquivo comprimido (definido na Seção 3.2). Este valor é obtido por

$$NBPP = \frac{\text{número de neurônios da camada escondida}}{\text{número de pixels da janela}} \times 8 \text{ bits por pixel.}$$

Observa-se que esta medida depende apenas das dimensões da rede neuronal e não da imagem.

Naturalmente, há um compromisso entre a qualidade da imagem comprimida e a taxa de compressão obtida. Várias redes neuronais podem ser treinadas para permitir diversas taxas de compressão.

O que foi descrito até aqui funcionou bem para imagens em tons de cinza, tanto a nível de inspeção visual das imagens descomprimidas, quanto a nível da relação sinal/ruído. Para imagens coloridas, este procedimento simples não funcionou de maneira satisfatória. Os três componentes RGB variando independentemente de 0 a 255 geram um conjunto muito grande de valores e a rede neuronal não consegue separar adequadamente as categorias neste hiperespaço.

Neste caso, é feito um pré-processamento utilizando-se o valor médio de cada componente RGB na janela, conforme descrito na Seção 4.1.1. Como estes valores médios (1 *byte* cada um dos três componentes) também precisam ser armazenados no arquivo comprimido, para imagens coloridas o número de *bits* por *pixel* é dado por:

$$NBPP = \frac{\text{número de neurônios da camada escondida} + 3}{\text{número de pixels da janela}} \times 8 \text{ bits por pixel.}$$

5.2.2 Resultados

Algumas configurações (variações no número de neurônios por camada) de *perceptrons* de duas camadas foram escolhidas e treinadas para se determinar os pesos das conexões. O treinamento foi feito a partir de imagens em tons de cinza e coloridas. Este foi um processo exaustivo (várias horas de CPU), resultando em arquivos de dados de redes para serem utilizadas na compressão de imagens. A Figura 16 mostra as imagens coloridas tratadas em 24 *bits* utilizadas neste processo.

No treinamento para compressão em tons de cinza, as imagens coloridas da Figura 16 foram transformadas para 256 níveis de cinza, resultando nas imagens da Figura 17.

Para as imagens em tons de cinza foram utilizadas janelas 4 x 4, com 4 e 8 neurônios na camada escondida. As imagens coloridas foram processadas com janelas 3 x 3, 4 x 4 e 6 x 6, com 3, 4 e 6 neurônios na camada escondida, respectivamente.

Em seguida, foram feitos alguns testes de compressão/descompressão com as imagens coloridas tratadas em 24 *bits* das Figuras 18a e 18b, que não haviam sido utilizadas no treinamento. Para compressão em tons de cinza, estas imagens foram igualmente transformadas para 256 níveis de cinza, resultando nas imagens das Figuras 19a e 19b.

As tabelas 1 e 2 apresentam os resultados da compressão em tons de cinza das imagens das Figuras 19a e 19b, respectivamente. As tabelas 3 e 4 mostram os resultados da compressão das versões coloridas originais destas imagens.

Os resultados numéricos são razoáveis e para efeito de comparação, dois métodos de compressão convencionais DPCM (*Differential Pulse Code Modulation*) e BTC (*Block Truncation Coding*) [JAIN89] foram aplicados nas imagens em tons de cinza. Foram utilizadas implementações destes algoritmos feitas por [SILVA93]. Infelizmente não havia, no momento, *software* disponível para compressão de imagens coloridas. Os resultados da aplicação dos métodos são mostrados em conjunto com os resultados da rede neuronal nas tabelas 1 e 2.

Nas Figuras 20a e 20b estão as imagens descomprimidas correspondentes à compressão em tons de cinza das imagens das Figuras 19a e 19b, respectivamente. As imagens descomprimidas resultantes da compressão colorida são mostradas nas Figuras 21a e 21b. Pode-se observar que o resultado da descompressão é bastante satisfatório, sendo imperceptível o efeito de *blocking* devido ao tratamento por janelas. Em comparação ao processo de treinamento, os procedimentos de compressão e descompressão são bastante rápidos (da ordem de segundos para as imagens das Figuras 19a e 19b).

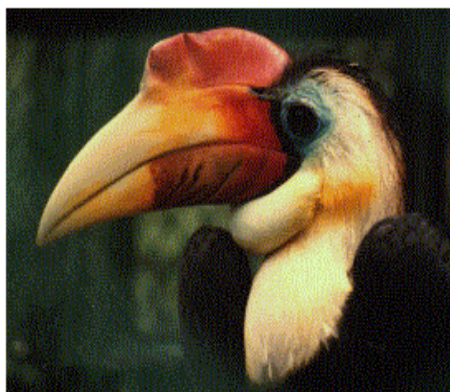
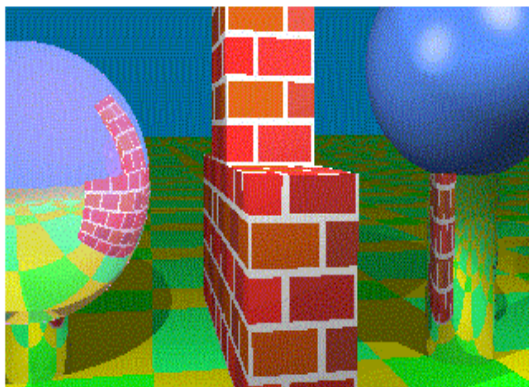


Figura 16. Imagens coloridas utilizadas nos treinamentos.

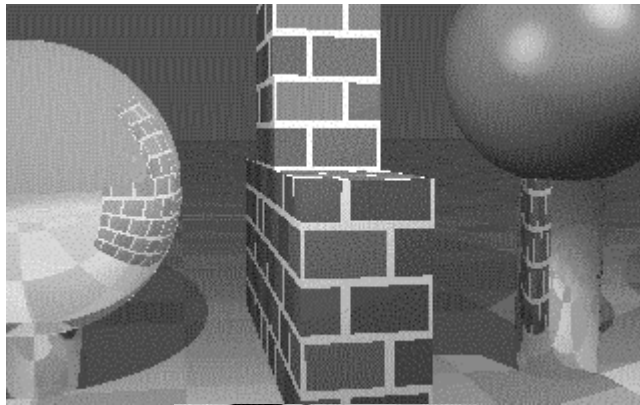
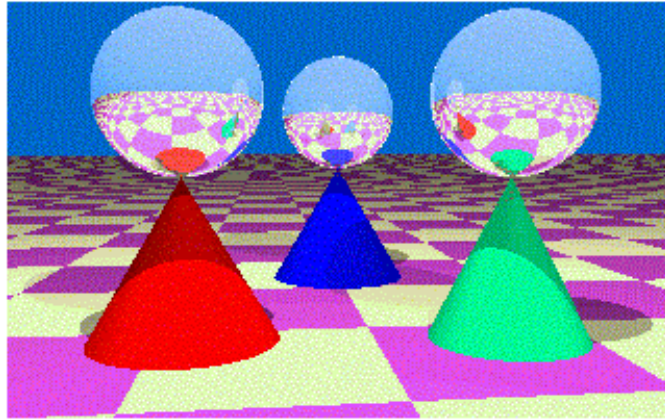


Figura 17. Imagens em tons de cinza utilizadas nos treinamentos.

(a)



(b)

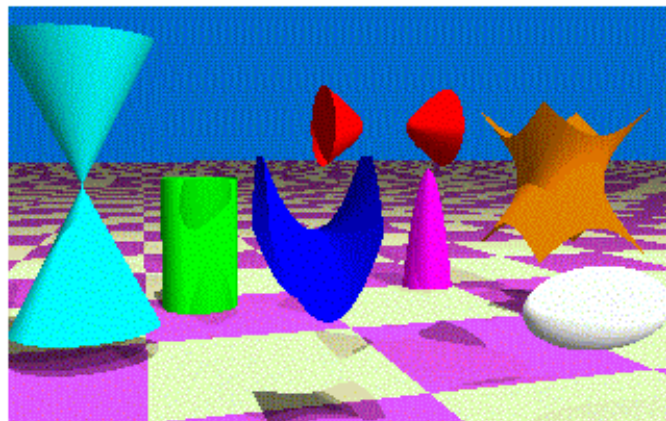
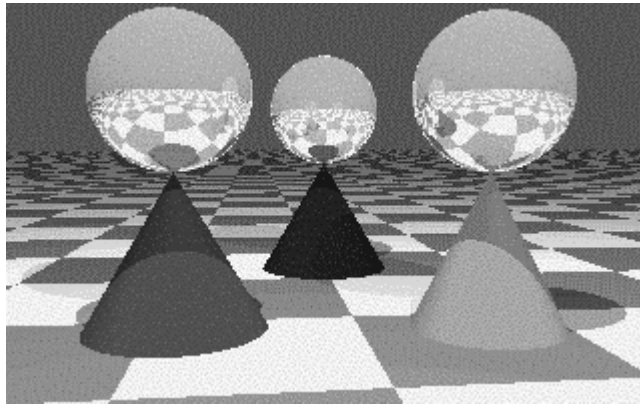


Figura 18. Imagens coloridas utilizadas nos testes de compressão/descompressão.

(a)



(b)

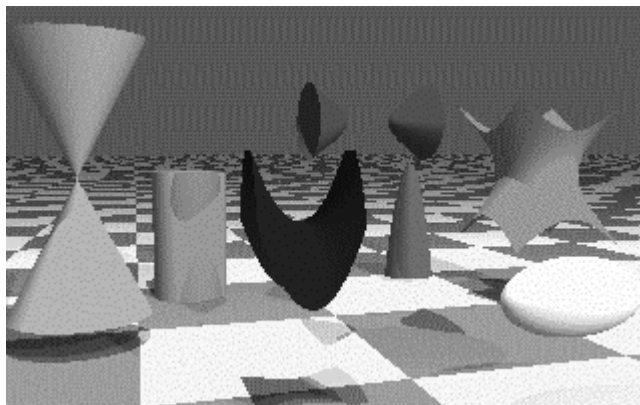


Figura 19. Imagens em tons de cinza utilizadas nos testes de compressão/descompressão.

Tabela 1

Resultados correspondentes à imagem da Figura 19a processada em tons de cinza

	número de <i>bits/pixel</i>	SNR	taxa de compressão
BTC	2,0	23,0	4,0:1,0
DPCM	4,4	23,0	1,8:1,0
<i>PERCEPTRON MULTICAMADAS</i>			
janela 4 x 4; 4 neurônios na camada escondida	2,0	15,0	4,0:1,0
janela 4 x 4; 8 neurônios na camada escondida	4,0	17,0	2,0:1,0

Tabela 2

Resultados correspondentes à imagem da Figura 19b processada em tons de cinza

	número de <i>bits/pixel</i>	SNR	taxa de compressão
BTC	2,0	25,0	4,0:1,0
DPCM	4,4	24,0	1,9:1,0
<i>PERCEPTRON MULTICAMADAS</i>			
janela 4 x 4; 4 neurônios na camada escondida	2,0	15,0	4,0:1,0
janela 4 x 4; 8 neurônios na camada escondida	4,0	16,0	2,0:1,0

Tabela 3

Resultados correspondentes à imagem da Figura 18a colorida tratada em 24 *bits*

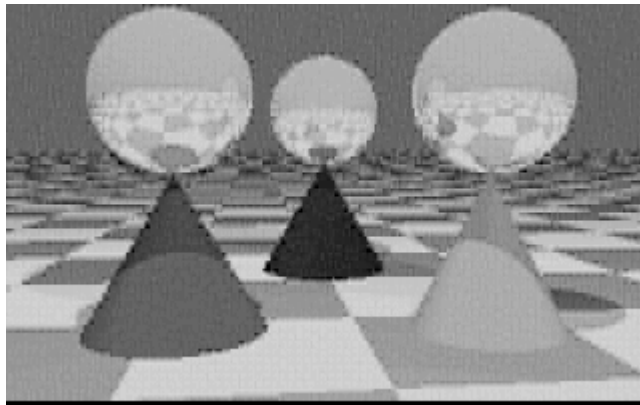
<i>PERCEPTRON MULTICAMADAS</i>			
	número de <i>bits/pixel</i>	SNR	taxa de compressão
janela 3 x 3; 3 neurônios na camada escondida	5,3	15,8	4,5:1,0
janela 4 x 4; 4 neurônios na camada escondida	3,5	15,3	6,8:1,0
janela 6 x 6; 6 neurônios na camada escondida	2,0	14,4	12,0:1,0

Tabela 4

Resultados correspondentes à imagem da Figura 18b colorida tratada em 24 *bits*

<i>PERCEPTRON MULTICAMADAS</i>			
	número de <i>bits/pixel</i>	SNR	taxa de compressão
janela 3 x 3; 3 neurônios na camada escondida	5,3	16,2	4,5:1,0
janela 4 x 4; 4 neurônios na camada escondida	3,5	15,5	6,8:1,0
janela 6 x 6; 6 neurônios na camada escondida	2,0	14,5	12,0:1,0

(a)



(b)

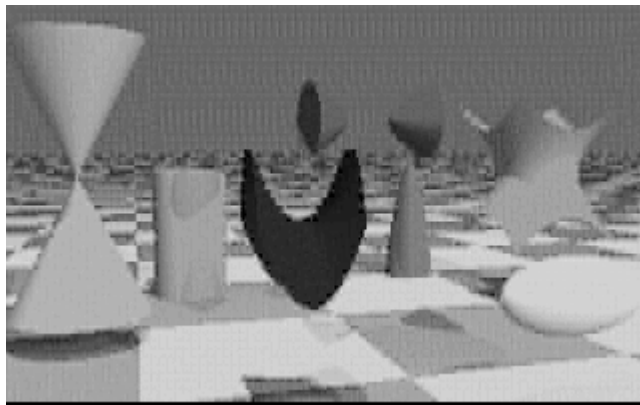
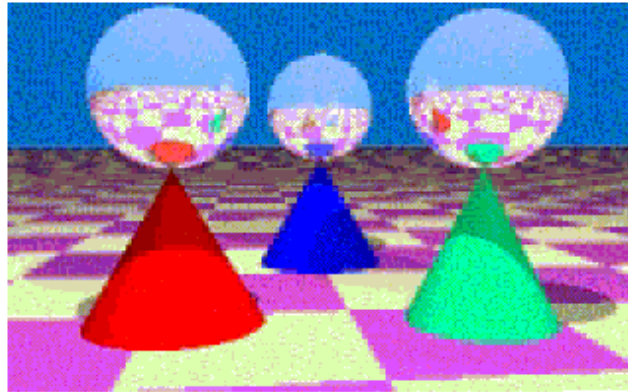


Figura 20. Imagens em tons de cinza descomprimidas por *perceptron* multicamadas com 2 bits/pixel . (a) Imagem correspondente à Figura 19a. (b) Imagem correspondente à Figura 19b.

(a)



(b)

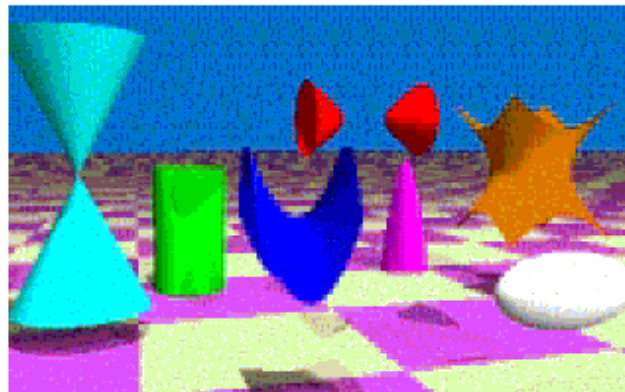


Figura 21. Imagens coloridas tratadas em 24 *bits* descomprimidas por *perceptron* multicamadas com 5,3 *bits/pixel*.(a) Imagem correspondente à Figura 18a.(b) Imagem correspondente à Figura 18b.

5.3 Mapa auto-organizativo

5.3.1 Implementação

Para realizar a compressão de imagens por quantização vetorial (ver Seção 4.2) foi implementado o mapa auto-organizativo, conforme descrito na Seção 2.3.

O mapa auto-organizativo tem apenas uma camada (ver Figura 7). Cada neurônio recebe os mesmos sinais de entrada. Da mesma forma que no *perceptron* multicamadas, estes sinais são os valores dos *pixels* de uma janela da imagem a ser processada. Os sinais de entrada são ponderados pelos pesos das conexões, os quais precisam ser determinados por um processo de aprendizado.

No caso de imagens em tons de cinza, estes sinais de entrada são os níveis de cinza dos *pixels*. No caso de imagens coloridas são os valores dos componentes RGB. Isto significa que para imagens coloridas deve haver três vezes mais conexões por neurônio do que para imagens em tons de cinza. Para o treinamento, isto é, para a formação do mapa de categorias, é necessário um processo de aprendizado que pode ser resumido por

passo 1. dimensionar a rede neuronal;

passo 2. inicializar os pesos das conexões com valores no intervalo -0,01 a +0,01;

passo 3. carregar nova imagem; se não houver mais imagens, ir para passo 10;

passo 4. ler nova janela da imagem; se não houver mais janelas, ir para passo 3;

passo 5. entrar com janela na rede neuronal e calcular desvio para cada neurônio;

passo 6. encontrar o de menor desvio e elegê-lo como vencedor;

passo 7. atualizar frequência de vitórias do vencedor;

passo 8. atualizar pesos das conexões de todos os neurônios;

passo 9. se menor desvio for suficientemente pequeno, ir para passo 11;

passo 10. reinicializar lista de imagens e retornar ao passo 3;

passo 11. gravar arquivo de dados da rede;

passo 12. fim.

No passo 1, devem ser fornecidos: o número de neurônios da rede, o número de conexões de cada um, o que corresponde ao tamanho da janela a ser utilizada, e o valor do fator de atualização dos pesos. Aqui, também, há valores *default* para facilidade do usuário.

No passo 2, os pesos devem ser inicializados com valores pequenos, por exemplo no intervalo $-0,01$ a $+0,01$.

O método de atualização dos pesos implementado foi o de sensibilidade à frequência [KRISHNAMURTHY90] descrito no Capítulo 3. Para isto, é mantido um contador do número de vitórias para cada neurônio. Como já foi visto, este contador pondera o desvio de cada neurônio possibilitando que diversos neurônios sejam vencedores.

Quando o desvio for suficientemente pequeno, o processo de aprendizado é finalizado e a rede está apta a classificar padrões nas categorias que foram formadas. Este processo pode ser resumido por:

passo 1. inicializar rede neuronal a partir do arquivo de dados gerado no aprendizado;

passo 2. carregar imagem a ser comprimida;

passo 3. ler nova janela da imagem; se não houver mais janelas, ir para passo 6;

passo 4. entrar com janela na rede neuronal;

passo 5. encontrar neurônio vencedor, gravar seu índice em arquivo e retornar ao passo 3;

passo 6. fim.

Os índices gravados em arquivo formam a imagem comprimida. O processo de descompressão pode ser resumido por:

passo 1. inicializar rede neuronal a partir do arquivo de dados gerado no aprendizado;

passo 2. ler índice i no arquivo da imagem comprimida, se não houver mais índices, ir para passo 5;

passo 3. formar janela a partir dos pesos do neurônio de índice i ;

passo 4. mostrar janela e retornar ao passo 2;

passo 5. fim.

O processo de compressão é mais lento do que o de descompressão, isto porque no primeiro é necessário fazer-se uma pesquisa seqüencial de índices para cada janela da imagem o que, dependendo do tamanho da rede neuronal, pode ser bastante demorado.

A medida da compressão obtida é novamente o número de *bits* por *pixel* utilizado no arquivo da imagem comprimida. Neste caso, o valor é dado por:

$$NBPP = \frac{\text{número de bits do número de neurônios}}{\text{número de pixels da janela}}.$$

Da mesma forma que na compressão por *perceptron* multicamadas, as imagens coloridas foram pré-processadas utilizando-se a média por componente RGB na janela. De modo, que neste caso, o número de *bits* por *pixel* é dado por:

$$NBPP = \frac{\text{número de bits do número de neurônios} + 24}{\text{número de pixels da janela}}.$$

5.3.2 Resultados

Inicialmente, procedeu-se a definição das categorias pelo mapa auto-organizativo. Isto foi feito a partir das imagens das Figura 16 e 17.

O processamento de imagens em tons de cinza e coloridas, da mesma forma que no *perceptron* multicamadas, envolveu a transformação de imagens originalmente coloridas para 256 níveis de cinza.

Foram gerados alguns arquivos de códigos para serem utilizados na codificação de imagens em tons de cinza e coloridas.

As imagens foram processadas em janelas 2 x 2 [KRISHNAMURTHY90] , com mapas de 64, 128, 256 e 512 neurônios. Os resultados da codificação das imagens das Figuras 18a, 18b, 19a e 19b são mostrados nas Tabelas 5,6 7 e 8.

Nas Figuras 22a, 22b, 23a e 23b são mostradas as imagens resultantes da decodificação das imagens comprimidas.

Novamente, destaque-se que a "qualidade" das imagens descomprimidas é muito boa a nível de inspeção visual, tanto para as imagens em tons de cinza, quanto para as coloridas.

Deve-se observar que a função do mapa auto-organizativo é apenas gerar os códigos para a quantização vetorial. Os valores da taxa de compressão (obtida da mesma forma que no caso do *perceptron* multicamadas) e do número de *bits* por *pixel* são dados que dependem do tamanho do livro código, e não de como ele foi gerado. Obviamente, o processo de definição das categorias deve ser tal que haja uma boa distribuição das categorias no espaço de índices.

Tabela 5

Resultados correspondentes à imagem da Figura 19a processada em tons de cinza

MAPA AUTO-ORGANIZATIVO janelas 2 x 2			
	número de <i>bits/pixel</i>	SNR	taxa de compressão
64 neurônios	1,2	16,0	6,7:1,0
128 neurônios	1,7	16,6	4,7:1,0
256 neurônios	2,0	17,0	4,0:1,0
512 neurônios	2,2	18,1	3,6:1,0

Tabela 6

Resultados correspondentes à imagem da Figura 19b processada em tons de cinza

MAPA AUTO-ORGANIZATIVO janelas 2 x 2			
	número de <i>bits/pixel</i>	SNR	taxa de compressão
64 neurônios	1,2	17,0	6,7:1,0
128 neurônios	1,7	18,3	4,7:1,0
256 neurônios	2,0	19,0	4,0:1,0
512 neurônios	2,2	20,0	3,6:1,0

Tabela 7

Resultados correspondentes à imagem da Figura 18a colorida tratada em 24 *bits*

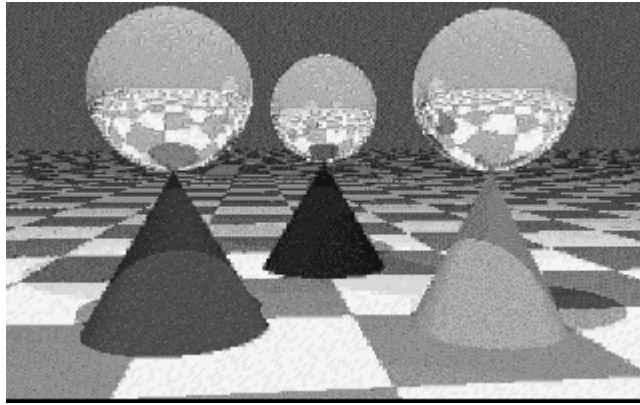
MAPA AUTO-ORGANIZATIVO janelas 2 x 2			
	número de <i>bits/pixel</i>	SNR	taxa de compressão
64 neurônios	7,5	11,1	3,2:1,0
128 neurônios	7,7	11,6	3,1:1,0
256 neurônios	8,0	12,3	3,0:1,0
512 neurônios	8,2	13,1	2,9:1,0

Tabela 8

Resultados correspondentes à imagem da Figura 18b colorida tratada em 24 *bits*

MAPA AUTO-ORGANIZATIVO janelas 2 x 2			
	número de <i>bits/pixel</i>	SNR	taxa de compressão
64 neurônios	7,5	13,4	3,2:1,0
128 neurônios	7,7	13,9	3,1:1,0
256 neurônios	8,0	14,7	3,0:1,0
512 neurônios	8,2	15,5	2,9:1,0

(a)



(b)

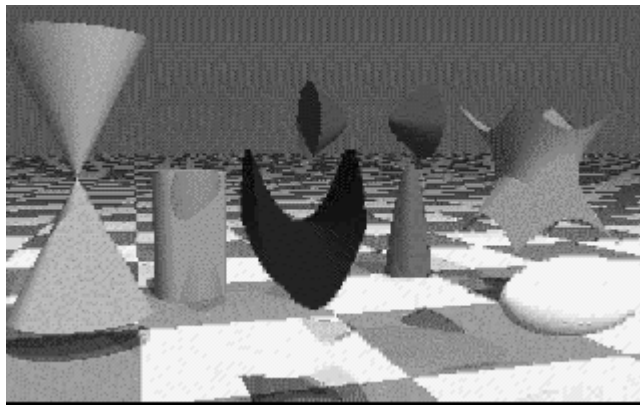
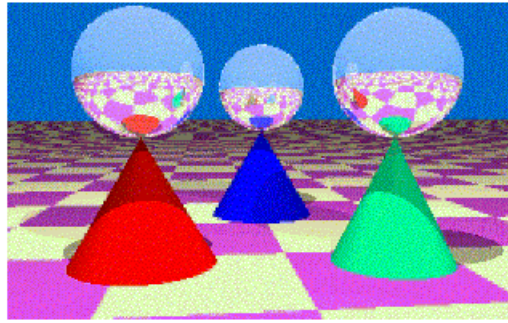


Figura 22. Imagens em tons de cinza decodificadas por quantização vetorial com 2 bits/pixel . (a) Imagem correspondente à Figura 19a. (b) Imagem correspondente à Figura 19b.

(a)



(b)

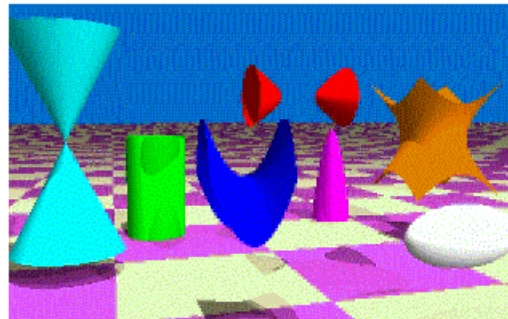


Figura 23. Imagens coloridas tratadas em 24 *bits* decodificadas por quantização vetorial com 8 *bits/pixel*. (a) Imagem correspondente à Figura 18a. (b) Imagem correspondente à Figura 18b.

5.4 Detecção de bordas por *perceptron* multicamadas

5.4.1 Implementação

A detecção de bordas parte do conceito de borda conforme apresentado no Capítulo 4. Cada janela 3x3 da imagem é processada pela rede neuronal para verificar se constitui ou não uma borda.

Para imagens em tons de cinza, o próprio valor do nível de cinza é usado para definir o padrão de borda. Para imagens coloridas, isto é feito nos três componentes RGB como mostrado no Capítulo 4.

Foi utilizado um *perceptron* multicamadas de 9 neurônios na camada de entrada, 3 na camada escondida e 1 na camada de saída (ver Figura 3). Os 9 neurônios na entrada correspondem a uma janela 3x3 de uma imagem. Para treinar a rede, foram utilizados os padrões de borda pré-definidos mostrados na Figura 10.

Foi utilizada a função de transferência sigmóide

$$f(u) = \frac{1}{1 + \exp(-u)},$$

que gera valores no intervalo (0,1). Outras funções foram testadas mas esta é a que apresentou melhores resultados em termos de tempo de treinamento.

No treinamento, foram utilizados padrões binários de borda, conforme descrito no Capítulo 4. Os pesos das conexões foram, então, determinados para futura utilização no processo de detecção de borda.

Na detecção de borda de imagens reais, é necessário um pré-processamento, uma vez que as imagens não são binárias e são até mesmo coloridas. Este pré-processamento está descrito no Capítulo 4.

Para detecção das bordas a imagem é percorrida *pixel a pixel* por janelas superpostas. Este procedimento pode ser resumido por:

passo 1. inicializar rede neuronal com parâmetros gerados pelo treinamento;

passo 2. carregar imagem;

passo 3. ler nova janela; se não houver mais janelas, ir para passo 7;

passo 4. pré-processar janela, conforme descrito no Capítulo 3;

passo 5. passar janela pré-processada pela rede neuronal;

passo 6. se a saída do neurônio da camada de saída for maior do que 0,5, o *pixel* central da janela pertence a uma borda, caso contrário não; fazer o *pixel* central igual a 255 em caso afirmativo e igual 0 em caso negativo; retornar ao passo 3;

passo 7. fim.

No pré-processamento, é utilizado um limiar para construir a janela binária a partir da janela real, sendo que este limiar deve ser fornecido pelo usuário ou pode-se usar o valor *default* existente.

5.4.2 Resultados

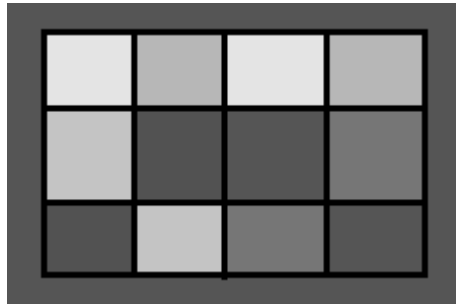
Como demonstração de utilização deste conceito de detecção de borda, a Figura 24b mostra o resultado da aplicação do algoritmo na imagem da Figura 24a. A imagem sintética binária tem suas bordas perfeitamente definidas pela rede neuronal.

Na Figura 25a, aparece uma imagem real em tons de cinza e na Figura 25b é mostrado o resultado da aplicação do algoritmo com limiar de 5. Na Figura 25c está o resultado da aplicação de um algoritmo convencional (operador de Sobel), utilizando-se o mesmo pré-processamento com limiar de 5. Pode-se observar que o algoritmo apresenta resultados bastante satisfatórios em comparação ao operador de Sobel, eliminando grande parte do ruído sem, no entanto, perder o essencial. O valor do limiar foi escolhido de modo a manter bastante ruído e permitir a diferenciação entre os algoritmos (para valores maiores, a binarização prévia da imagem torna praticamente indistinguível os resultados obtidos).

A Figura 26b apresenta o resultado da aplicação do algoritmo na imagem colorida (tratada em 24 *bits*) da Figura 26a, utilizando-se um limiar de 30.

Isto indica que esta abordagem é válida e pode ser melhor explorada através de outros dimensionamentos como janelas 5 x 5 e conceitos mais refinados de borda.

(a)



(b)

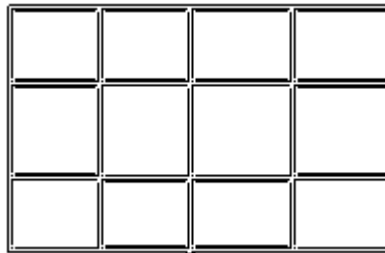


Figura 24. (a) Imagem sintética em tons de cinza. (b) Mapa de bordas da imagem sintética gerado pela rede neuronal.

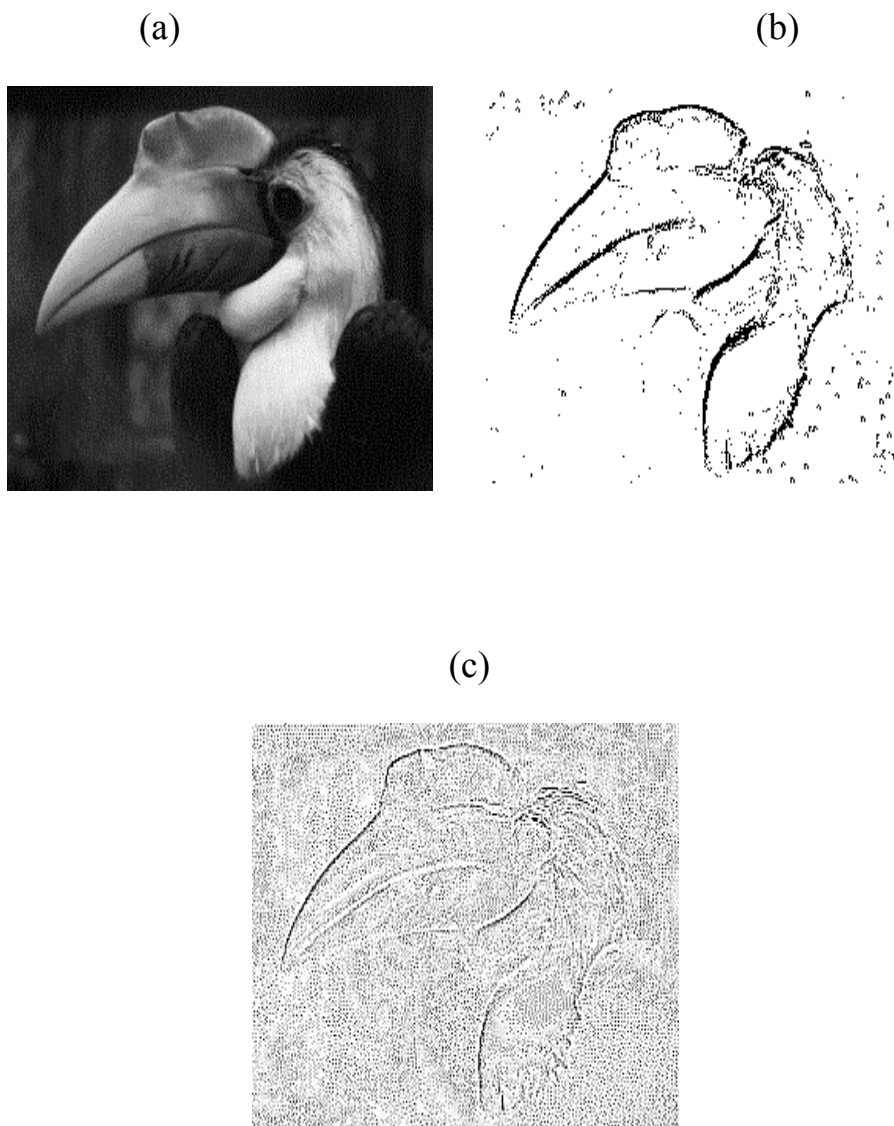


Figura 25. (a) Imagem real em tons de cinza. (b) Mapa de bordas gerado pela rede neuronal. (c) Resultado obtido com o operador de Sobel.

(a)



(b)



Figura 26. (a) Imagem real colorida tratada em 24 *bits*. (b) Mapa de bordas gerado pela rede neuronal para esta imagem.(observação: o mapa de bordas foi impresso em negativo)

5.5. Conclusão

Neste capítulo, foram apresentadas as implementações de alguns paradigmas de redes neuronais para aplicação em processamento digital de imagens.

Os resultados de alguns testes de demonstração e comparação foram apresentados. Os aspectos gerais da implementação dos algoritmos foram discutidos, tendo em vista esclarecer possíveis usuários das ferramentas implementadas.

Em primeiro lugar, foi apresentada a implementação do *perceptron* multicamadas para compressão de imagens. No caso de imagens em tons de cinza o seu desempenho é razoável, tanto a nível de inspeção visual quanto a nível da relação sinal/ruído. Para imagens coloridas, o autor teve de recorrer a um artifício de utilização do valor médio dos *pixels* da janela. Naturalmente, para se conhecer o real potencial deste tipo de abordagem, devem ser feitos testes exaustivos com outras imagens e comparações com outros tipos de algoritmos.

A compressão de imagens por quantização vetorial, expressa na implementação do mapa auto-organizativo com algoritmo de aprendizado sensível à frequência, também apresentou resultados razoáveis para imagens em tons de cinza. Para imagens coloridas o autor teve de recorrer, novamente, ao artifício do valor médio por componente de cor.

A detecção de bordas por *perceptron* multicamadas indicou que o tipo de abordagem adotada merece atenção, inclusive através da utilização de outros paradigmas, como o de Hopfield, discutido no Capítulo 4.

O treinamento para determinação dos pesos das conexões das redes neuronais pode ser bastante demorado (várias horas de CPU), dependendo de quantas imagens forem utilizadas no treinamento.

Após essa etapa, o tempo do processo de compressão/descompressão por *perceptron* multicamadas é comparável ao tempo de processamento dos algoritmos convencionais.

No caso do mapa auto-organizativo, o tempo de compressão de uma imagem depende do tamanho do código, isto é, do número de neurônios. Isto é uma característica do procedimento de quantização vetorial, que exige uma pesquisa por todo o código no processo de compressão, e não da rede neuronal. Por outro lado, o processo de descompressão é muito rápido (da ordem de segundos).

No caso da detecção de bordas por *perceptron* multicamadas, o tempo de processamento para geração do mapa de bordas é maior do que para o operador de Sobel. Isto decorre do fato de a rede neuronal fazer o processamento em ponto flutuante. No entanto, como pode ser visto pelas Figuras 25a, 25b e 25c, os resultados compensam.

Outros modelos além do RGB também podem ser utilizados, como: o modelo HSV (H = matiz, S = saturação e V = brilho), o sistema YIQ (Y = luminância, I e Q são os componentes de cromaticidade). Isto pode resultar em que um componente possa ser tratado com menos detalhe do que os outros e melhorar, desta forma, a qualidade dos resultados obtidos [GOLDBERG86].

6

Conclusão

Neste trabalho, foi demonstrado o potencial de aplicação de redes neuronais em processamento digital de imagens, o que justifica futuros trabalhos nesta área. Outras aplicações de redes neuronais em PDI, como segmentação [AMARTUR92], análise de texturas [KOSKO92] e filtros neuronais, podem vir a ser objeto de futuros desenvolvimentos no DCC-UFMG.

Ficou claro com este trabalho que a importância das redes neuronais está na sua capacidade de generalização a partir de informações não bem definidas. Muitos problemas, no entanto, podem ser tratados por métodos convencionais. Outro ponto importante é o pré-processamento (através da transformada de Fourier por exemplo), que pode ser de muita utilidade para reduzir o volume de dados a ser tratado pela rede neuronal.

Os resultados obtidos justificam plenamente o esforço dispendido em desenvolvimento e programação. Outros refinamentos são possíveis a partir dos algoritmos já implementados:

- eliminação de janelas repetidas no arquivo comprimido; isto certamente aumentaria a taxa de compressão;
- utilização de diversos tamanhos de janelas e variações na relação entre o número de neurônios das camadas;
- uso de janelas maiores e conceitos mais sofisticados de borda, inclusive com gradação no nível de cinza;
- tratamento de imagens coloridas por componentes de cores RGB separados;
- truncamento não linear na saída da camada escondida do *perceptron* multicamadas, levando em conta a distribuição não uniforme dos valores;
- utilização de *perceptron* de três camadas, isto é, com duas camadas escondidas.

Para realização destes trabalhos, os algoritmos aqui implementados são de grande valia, estando disponíveis aos usuários.

A aplicação do modelo de Hopfield na detecção de bordas e na restauração de imagens indica que a implementação destes algoritmos deve ser considerada.

Todos os algoritmos apresentados podem se beneficiar do uso de paralelismo. No DCC- UFMG há um simulador de redes neuronais em ambiente heterôgeneo que pode ser usado para esta finalidade. Este utilitário permite a configuração de alguns tipos de redes neuronais na rede de estações de trabalho do DCC-UFMG. Outra possibilidade, mais ambiciosa, é a utilização de placas de redes neuronais já disponíveis comercialmente.

Finalmente, na forma como estão implementados, estes algoritmos podem ser facilmente incorporados ao ambiente KHOROS [RASURE91]. Esta interface gráfica para processamento digital de imagens possui uma série de facilidades para incorporação de novos procedimentos desenvolvidos por usuários.

Referências bibliográficas

[AMARTUR92] Amatur, S. C., Piraino, D. and Takefuji, Y. Optimization Neural Networks for the Segmentation of Magnetic Resonance Images. **IEEE Transactions on Medical Imaging**, v. 11, n. 2, Jun. 1992.

[AMIT89] Daniel, D.J. **Modeling Brian Functions, The World of Attractor Neural Networks**. Cambridge University Press 1989.

[ANDREWS77] Andrews, H. C. and Hunt, B. R., **Digital Image Restoration**. Prentice-Hall, 1977.

[BRESSLOF91] Bresslof, P. C. and Weir, D. J. Neural Networks. **GEC Journal of Research**, v. 8, n. 3, 1991.

[CARVALHO91] Carvalho, L. A. V., Barbosa V. C., Mendes, S. T., Eizirik, L. M. e França, F. G. Redes Neurais Artificiais: A Volta do Cérebro Eletrônico? **Ciência Hoje**, v. 12, n. 70, 1991.

[CAUDILL90] Caudill, M. Neural Network Primer. **AI EXPERT Magazine**, 1990.

[CAUDILL93] Caudill, M. Kohonen Maps. **AI EXPERT**, Jun. 1993.

[CHARNIAK86] Charniak, E. and McDermott, D. **Introduction to Artificial Intelligence**. Addison Wesley, 1986.

[COHEN83] Cohen, M. A. and Grossberg, S. Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks. **IEEE Transactions on Systems, Man and Cybernetics**, v. SMC-13, Sep./Oct. 1983.

[COTTRELL87] Cottrell, G., Munro, P. and Zipser, D. **Image Compression by Backpropagation, An Example of Extensional Programming**. Institute For Cognitive Science, University of California. San Diego, La Jolla, California, 1987.

[DAVIS92] Davis Jr., C. A. **Pixelware, um Sistema de Processamento Digital de Imagens**. Dissertação de Mestrado em Ciência da Computação. UFMG, Belo Horizonte, 1992.

[EGLLEN92] Eglen, S. J., Hill, A. G., Lazare, F. J. and Walker, N. P. Using Neural Networks. **GEC Review**, v. 7, n. 3, 1992.

[ESGOLTS70] Esgolts, L. **Differential Equations and the Calculus of Variations**. Mir Publishers, Moscow, 1970.

[FAHLMANN87] Fahlmann, S. E. and Hinton, G. E. Connectionist Architectures for Artificial Intelligence. **IEEE Computer**, Jan. 1987.

[FREEMANN92] Freemann, J. A. Backpropagation in a Neural Network. **AI EXPERT, Neural Network Special Report**, 1992.

[GALBIATI90] Galbiati Jr., L. J. **Machine Vision and Digital Image Processing Fundamentals**. Prentice-Hall, 1990.

[GIBSON90] Gibson, G. J. and Cowan, C. F. N. On the Decision Regions of Multilayer Perceptrons. **Proceedings of the IEEE**, v. 78, n. 10, Oct. 1990.

[GOLDBERG86] Goldberg, M., Boucher, P.R. and Shlien, S. Image Compression Using Adaptive Vector Quantization. **IEEE Transactions on Communications**, v. COM-34, n. 2, Feb. 1986.

[GONZALEZ87] Gonzalez, R. C. and Wintz, P. **Digital Image Processing**. 2ed. Reading MA, Addison-Wesley, 1987.

[GORNI93] Gorni, A. A. Redes Neurais Artificiais. **Micro Sistemas** ano XII, n. 133, nov. 1993.

[GRAY84] Gray, R. M. Vector Quantization. **IEEE ASSP Magazine**, Apr. 1984.

[HERTZ91] Hertz, J., Krogh, A. and Palmer, R. G. **Introduction to the Theory of Neural Computation**. Addison-Wesley 1991. [HOPFIELD85] Hopfield, J. J. and Tank, D. W., 'Neural' Composition of Decisions Optimization Problems, **Biological Cybernetics**, v. 52, 1985.

[HOPFIELD86] Hopfield, J. J. and Tank, D. W. Computing with Neural Circuits: a Model. **Science**, v. 233, Aug. 1986.

[HUBEL79] Hubel, D.H. The Brain. **Scientific American**, Sept. 1979.

[HUSSAIN91] Hussain, Z. **Digital Image Processing, Practical Applications of Parallel Processing Techniques**. Ellis Horwood Limited, 1991.

[JAIN89] Jain, A. K. **Fundamentals Of Digital Image Processing**. Prentice-Hall, 1989.

[JAMISON88] Jamison, T. A. and Schalkoff, R. J. Image Labelling: A Neural Network Approach. **Image and Vision Computing**, v. 6, n. 4, Nov. 1988.

[JANSSON91] Jansson, P. A. Neural Networks: An Overview. **Analytical Chemistry**, v. 53, n. 6, Mar. 1991.

[JONES87] Jones, W.P. and Hoskins, J. Backpropagation, a generalized delta rule. **BYTE Magazine**, Oct 1987.

[JOSIN87] Josin, Gal. Network Heuristics, three heuristic algorithms that learn from experience. **Byte Magazine**, Oct. 1987.

[KLIMASAUSKAS90] Klimasauskas, C. C. Neural Networks and Image Processing. **Dr. Dobb's Journal**, Apr. 1990.

[KOHONEN89] Kohonen, T. **Self-Organization and Associative Memory**. Springer-Verlag, 3th ed., 1989.

[KOHONEN90] Kohonen, T. The Self-Organizing Map. **Proceedings of the IEEE**, v. 78, n. 9, Sep. 1990.

[KOSKO88] Kosko, B., Bidirectional Associative Memories. **IEEE Transactions on Systems, Man and Cybernetics**, v.18, n.1, Jan./Feb. 1988.

[KOSKO92-1] Kosko, B. **Neural Networks for signal processing**. Prentice-Hall, Inc, 1992.

- [KOSKO92-2] Kosko, B. **Neural Networks and Fuzzy Systems**. Prentice Hall, 1992.
- [KREYSZIG70] Kreyszig, E. **Introductory Mathematical Statistics**. John Wiley & Sons, 1970.
- [KRISHNAMURTHY90] Krishnamurthy, A. K., et alii. Neural Networks for Vector Quantization of Speech and Images. **IEEE Journal On Selected Areas in Communications**, v. 8, n. 8, Oct. 1990.
- [KÜRKOVA92] Kůrková, V. Kolmogorov's Theorem and Multilayer Neural Networks. **Neural Networks**, v. 5, 1992.
- [LAUTRUP88] Lautrup, B. **Lectures Notes on the Theory of the Hopfield Model**. The Niels Bohr Institute, Jan. 24th, 1988.
- [LIM89] Lim, J. S. **Two-dimensional Signal and Image Processing**. Prentice-Hall, 1990.
- [LIPPMANN87] Lippmann, R. P., An Introduction to Computing with Neural Nets. **IEEE ASSP Magazine**, Apr. 1987.
- [MÜLLER91] Müller, B. and Reinhardt, J. **Neural Networks, An Introduction**. Springer-Verlag, 1991.
- [NIBLACK86] Niblack, W. **An Introduction to Digital Image Processing**. Prentice-Hall, 1986.
- [REECE87] Reece, P. Perceptions & Neural Nets. **AI EXPERT**, Jan. 1987.
- [RASURE91] Rasure, J. and Williams, C. KHOROS, "An Integrated Visual Language and Software Development Environment". **Visual Computing**, v.2, 1991.
- [ROSENBLATT62] Rosenblatt, F. **Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms**. Spartan books, 1962.
- [SALEM91] Salem, J. G. and Young, T. Y. A Neural Network Approach to the Labelling of Line Drawings. **IEEE Transactions on Computers**, v. 40, n. 12, Dec. 1991.

- [SCHALKOFF89] Schalkoff, R. J. **Digital Image Processing and Computer Vision**. John Wiley & Sons, 1989.
- [SILVA93] Silva, N. I. **Um Sistema de Compressão de Imagens Aplicado a Documentos Históricos**. Dissertação de Mestrado, UFMG, 1993.
- [SIMPSON90] Simpson, P. K. **Artificial Neural Systems, Foundations, Paradigms, Applications and Implementations**. Pergamon Press, 1990.
- [SOUCEK88] Soucek, B. and Soucek, M. **Neural and Massively Paralell Computer, The Sixth Generation**. John Wiley & Sons, 1988.
- [STEVENS79] Stevens, C.F. The Neuron. **Scientific American**, Sept. 1979.
- [TAGLIARINI91] Tagliarini, G. A., Christ, J. F. and Page, E.W. Optimization Using Neural Networks. **IEEE Transactions On Computers**, v.40, n.12, Dec. 1991.
- [WIDROW90] Widrow, B. and Lehr, M. A. 30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation. **Proceedings of the IEEE**, v. 78, n. 9, Sep. 1990.
- [WOODS81] Woods, J. W. et alli. "Kalman Filtering in two Dimensions: Further Results". **IEEE Transactions on Acoustics Speech and Signal Processing**, v. ASSP-29, Apr. 1981.
- [ZURADA92] Zurada, J. M. **Introduction to Artificial Neural Systems**. West Publishing Copany, 1992.