Quantização Vetorial Aplicada à Compressão de Sinais de Voz e Imagem

Francisco Madeiro Bernardino Junior

Dissertação de Mestrado submetida à Coordenação dos Cursos de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Paraíba - Campus II como parte dos requisitos necessários para obtenção do grau de Mestre.

Área de Concentração: Processamento da Informação

Benedito Guimarães Aguiar Neto - Dr.-Ing.
Orientador

Campina Grande, Paraíba, Brasil

QUANTIZAÇÃO VETORIAL APLICADA À COMPRESSÃO DE SINAIS DE VOZ E IMAGEM

FRANCISCO MADEIRO BERNARDINO JÚNIOR

Dissertação Aprovada em 06.03.1998

PROF. BENEDITO GUIMARÃES AGUIAR NETO, Dr.-Ing., UFPB
Orientador

PROF. FRANCISCO MARCOS DE ASSIS, Dr., UFPB
Componente da Banca

PROF. MARCELO SAMPAIO DE ALENCAR, Ph.D., UFPB
Componente da Banca

CAMPINA GRANDE - PB

Dedicatória

Dedico este trabalho aos meus pais, Madeiro e Leni, e aos meus irmãos, Charles e Ricardo.

Agradecimentos

Gostaria de expressar meus sinceros agradecimentos a algumas pessoas e instituições que contribuíram para a realização deste trabalho:

Aos meus pais, a quem devo minhas principais conquistas, cujo carinho foi de fundamental importância nos momentos difíceis e cuja compreensão constituirá sempre minha referência de sublimidade.

À professora Rosângela Maria Vilar, principal responsável pelo grande impulso deste trabalho, a quem expresso meus votos de gratidão por suas valiosas críticas e sugestões, pelo estímulo e, principalmente, por sua amizade.

Ao professor Benedito Guimarães Aguiar Neto, por haver despertado em mim o interesse por Processamento de Sinais de Voz e por ter possibilitado a consecução de importantes realizações.

Ao meu amigo Geovany Araújo, por sua ajuda e presteza.

A todos que contribuíram para a realização deste trabalho: Joseana, Rinaldo, Paulo Márcio, Wallington, Aldenor, Josemar, Sérgio, Robson e demais companheiros do LAPS.

Aos meus grandes amigos Murilo, Fábio, Daniel, Lucas, Luís, Cassandra e Marcos, que sempre me apoiaram e incentivaram.

Aos professores Marcelo Sampaio de Alencar e Ângelo Perkusich, por suas valiosas sugestões e pelo estímulo.

À minha namorada Ana Cristina, presente no meu dia a dia, com seu afeto e companheirismo.

A Universidade Federal da Paraíba - Campus II e a CAPES, órgão financiador deste trabalho.

A Deus, por minha vida ser tão feliz, tranqüila e repleta de realizações.

Resumo

A quantização vetorial tem sido amplamente utilizada como uma excelente técnica de compressão de sinais, especialmente voz e imagem.

Neste trabalho, são apresentadas duas técnicas para projeto de quantizadores vetoriais. A primeira técnica diz respeito à introdução de algumas modificações no algoritmo de treinamento da rede neural de Kohonen. A segunda consiste na derivação de um algoritmo baseado em Análise de Componentes Principais. O desempenho de cada técnica é comparado ao desempenho apresentado pelo tradicional algoritmo LBG, ficando evidenciada a superioridade dos algoritmos propostos.

Para avaliação do enorme potencial e da eficiência da quantização vetorial, são apresentados resultados de simulações realizadas com sinais de voz, imagens e sinais com distribuições conhecidas.

Abstract

Vector quantization has been widely used as an excellent technique for signal compression, specially voice and image.

In this work two techniques for vector quantizers design are presented. The first technique concerns the introduction of some modifications in Kohonen's neural network training algorithm. The second concerns the derivation of an algorithm based on Principal Component Analysis. The performance of each technique is compared to the performance presented by the traditional LBG algorithm and the superiority of the proposed algorithms have been evidenced.

For evaluating the great potential and efficiency of vector quantization, results of simulations of voice signals, images and signals with known distributions are presented.

Lista de Símbolos e Abreviaturas

LAPS – Laboratório de Automação e Processamento de Sinais
CAPES – Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
LBG – Linde-Buzo-Gray
KMTAU – Kohonen Modificado com Taxa de Aprendizagem Uniforme
KMTAN – Kohonen Modificado com Taxa de Aprendizagem Não-Uniforme
PCA – Análise de Componentes Principais (*Principal Component Analysis*)
HDTV – Televisão de Alta Definição (*High Definition Television*)
ISDN – Redes Digitais de Serviços Integrados (*Integrated Services Digital Networks*)
QV – Quantização Vetorial
VLSI – Integração em Escala Muito Alta (*Very Large Scale Integration*)
bpp – Bit por Pixel
PNN – Vizinho Mais Próximo Dois a Dois (*Pair-Wise Nearest Neighbor*)
GLA – Algoritmo de Lloyd Generalizado (*Generalized Lloyd Algorithm*)
MOS – Escore Médio de Opinião (*Mean Opinion Score*)
SNR – Relação Sinal-Ruído (*Signal to Noise Ratio*)
SNR_{seg} – Relação Sinal-Ruído Segmental (*Segmental Signal to Noise Ratio*)
$PSNR$ – Relação Sinal-Ruído de Pico (*Peak Signal to Noise Ratio*)
MSE – Erro Médio Quadrático (*Mean Square Error*)
AR(1) – Fonte de Gauss-Markov de 1^a. Ordem
ROM – Memória Apenas de Leitura (*Read-Only Memory*)
CPU – Unidade Central de Processamento (*Central Processing Unit*)
SOFM – Mapa de Características Auto-Organizativo (*Self-Organizing Features Map*)
KLT – Transformada de Karhunen Loeve (*Karhunen Loeve Transform*)

DFT – Transformada Discreta de Fourier (*Discrete Fourier Transform*)

DWHT – Transformada Discreta de Walsh Hadamard (*Discrete Walsh Hadamard Transform*)

DCT – Transformada Discreta de Cosseno (*Discrete Cosine Transform*)

B – Largura de faixa

f_s – Taxa de amostragem

T – Intervalo de amostragem

I – Taxa de transmissão em bit/s

R – Taxa em $bit/amostra$

Q – Função de mapeamento da quantização vetorial

R^k – Espaço vetorial k-dimensional

k – Dimensão do quantizador vetorial

W – Dicionário do quantizador vetorial ou conjunto dos vetores-código

$\mathbf{w}_i$ – i-ésimo vetor-código pertencente ao dicionário W

N – Número de níveis do quantizador vetorial ou número de vetores-código

$\mathcal{C}$ – Partição promovida pela quantização vetorial

$\mathcal{C}_i$ – i-ésima célula da partição $\mathcal{C}$

$\mathbf{x}_l$ – l-ésimo vetor de entrada $\mathbf{x}$

$\mathbf{X}$ – Sinal de entrada

$Q(\mathbf{X})$ – Versão quantizada de $\mathbf{X}$

$d(\mathbf{x}, \mathbf{w}_i)$ – Distância entre o vetor $\mathbf{x}$ e o vetor $\mathbf{w}_i$

w_{ij} – j-ésima componente do vetor-código $\mathbf{w}_i$

x_j – j-ésima componente do vetor de entrada $\mathbf{x}$

$x(n)$ – n-ésima amostra do sinal de entrada $\mathbf{x}$

$y(n)$ – n-ésima amostra do sinal processado $\mathbf{y}$

$e(n)$ – n-ésima amostra do sinal erro $\mathbf{e} = \mathbf{x} - \mathbf{y}$

E_x – Energia contida no sinal original $\mathbf{x}$

E_e – Energia contida no sinal erro $\mathbf{e} = \mathbf{x} - \mathbf{y}$

$SNR(j)$ – Relação Sinal-Ruído (*Signal to Noise Ratio*) convencional para o j-ésimo segmento ou j-ésima janela de tempo

J – Número de segmentos ou janelas de tempo de comprimento típico de 15 a 25 ms

m_j – Instante final para a j-ésima janela de tempo

v_p – Valor de pico (255 para imagens quantizadas a 8 *bpp*)

a – Coeficiente de correlação da fonte de Gauss-Markov de 1ª. ordem

$\{w(n)\}$ – Seqüência de variáveis aleatórias com distribuição gaussiana, independentes e identicamente distribuídas, com média zero

$\{W(n)\}$ – Ruído gaussiano branco

$h(n)$ – Resposta ao impulso de um filtro

σ_x^2 – Variância do processo $\{X(n)\}$

σ_w^2 – Variância do ruído branco $\{W(n)\}$

σ_e^2 – Variância do erro de quantização $e(n)$, com $e(n) = X(n) - Q(X(n))$

$D_S(R)$ – Menor distorção obtenível para a taxa R

$\sigma_{e\,\min}^2$ – Menor valor possível para σ_e^2

$SNR_{\max}$ – Valor máximo de Relação Sinal-Ruído SNR

p_i – Probabilidade de que um dado vetor de entrada $\mathbf{x}$ pertença à região de Voronoi correspondente ao vetor-código $\mathbf{w}_i$

S – Longa seqüência representativa de n vetores $\mathbf{x}_p$ usados para o treinamento do quantizador vetorial

S_i – i-ésima partição do conjunto S, tal que $S_i = \{\mathbf{x}_p : \mathbf{w}_i = Q(\mathbf{x}_p)\}$

n_i – Tamanho da partição S_i

H_k – Entropia dos vetores-código

H_n – Entropia normalizada dos vetores-código

w_{kj} – Peso correspondente à j-ésima entrada do neurônio k

θ_k – Limiar do neurônio k

φ – Função de ativação do neurônio

u_k – Saída da combinação linear das entradas x_j do neurônio k

y_k – Saída do neurônio k

$\vec{x}$ – Vetor de treino

$d(\vec{x}, \vec{w}_i)$ – Distância do vetor $\vec{x}$ ao neurônio $\vec{w}_i$

$\vec{w}_{i^*}$ – Neurônio vencedor

$\mathcal{N}_{\vec{w}_{i^*}}$ – Vizinhança em torno do neurônio vencedor $\vec{w}_i$

$r(n)$ – Raio de vizinhança na n-ésima iteração

Δw_{ij} – Valor da modificação introduzida na j-ésima componente do vetor $\vec{w}_i$

$\eta(n)$ – Taxa de aprendizagem na n-ésima iteração

$\mathcal{O}_i$ – Função que define a vizinhança ao redor do neurônio $\vec{w}_i$.

x_j – j-ésima componente do vetor de treino $\vec{x}$

w_{ij} – j-ésima componente do neurônio $\vec{w}_i$

$\mathcal{N}_x$ – Vizinhança centrada no vetor de treino $\vec{x}$

$\mu_{x(n)}$ – Média da variável aleatória $X(n)$

$R_{xx}(m,n)$ – Função de autocorrelação do processo $\{X(n)\}$, onde m e n são duas variáveis temporais

$C_{xx}(m,n)$ – Função de autocovariância do processo $\{X(n)\}$, onde m e n são duas variáveis temporais

$R_{xx}(k)$ – Função de autocorrelação do processo $\{X(n)\}$, onde $k = n - m$

$C_{xx}(k)$ – Função de autocovariância do processo $\{X(n)\}$, onde $k = n - m$

ρ_k – Função de autocorrelação normalizada

$\mathbf{x}$ – Vetor de entrada, que representa uma seqüência de amostras

X – Vetor que representa a seqüência de variáveis aleatórias observadas

$\mathbf{R}_{xx}$ – Matriz de autocorrelação

$\mathbf{C}_{xx}$ – Matriz de autocovariância

$\boldsymbol{\mu}_x$ – Vetor-média, com todos elementos iguais a μ_x

λ_i – i-ésimo autovalor

$\mathbf{z}_i$ – i-ésimo autovetor

$\boldsymbol{\theta}$ – Vetor de atributos

$\theta(i)$ – i-ésima componente do vetor de atributos

$a(i,n)$ – Núcleo de transformação direta

$b(n,i)$ – Núcleo de transformação inversa

$\mathbf{A}$ – Matriz de transformação direta

$\mathbf{B}$ – Matriz de transformação inversa

$\mathbf{b}_i$ – Vetores de base, correspondentes às colunas de $\mathbf{B}$

$\mathbf{y}$ – Vetor de saída

$\mathbf{u}$ – Versão quantizada de $\boldsymbol{\theta}$

$\hat{\mathbf{x}}$ – Versão recuperada do vetor de entrada $\mathbf{x}$

$\mathbf{I}$ – Matriz identidade

$\mathbf{L}$ – Matriz cujas linhas são os autovetores de $\mathbf{R}_{xx}$

$\mathbf{l}_i$ – i-ésimo autovetor de $\mathbf{R}_{xx}$

L – Número vetores $\mathbf{z}_i$ ao longo de cujas direções os vetores-código devem ser alocados

$\lambda_{i,p}$ – Valor percentual relativo de cada autovalor λ_i

$\tilde{\mathbf{z}}_i$ – Vetor obtido de $\mathbf{z}_i$ através de um processo de normalização

N_i – Número de vetores-código a serem alocados ao longo das direções definidas pelo vetor $\tilde{\mathbf{z}}_i$

$\mathbf{w}_{i,n_i}$ – n_i-ésimo vetor alocado ao longo da i-ésima direção

$f(n_i, \lambda_i)$ – Escalar determinado assumindo-se que cada direção principal apresenta distribuição gaussiana com variância $\sigma_i{}^2 = \lambda_i$

Lista de Figuras

Lista de Tabelas

Índice

Capítulo 1

Introdução

A compressão de sinais, cujo objetivo principal é a minimização dos requisitos de largura de faixa e de capacidade de memória de armazenamento, desempenha um papel importantíssimo em diversas aplicações, dentre as quais podem ser citadas: redes digitais de serviços integrados, sistemas multimídia, telefonia móvel celular, televisão de alta definição e teleconferência.

Nesse cenário, a quantização vetorial, que tem sido objeto de estudo para aplicações envolvendo sinais de voz e imagem, apresenta-se como uma poderosa técnica, permitindo elevadas taxas de compressão. Convém salientar que de acordo com a Teoria da Distorção Versus Taxa, formulada por Shannon, é sempre possível obter um melhor desempenho codificando um bloco de amostras (isto é, um vetor de amostras), ao invés de codificar cada amostra individualmente. Em outras palavras, essa teoria ressalta a superioridade da quantização vetorial sobre a quantização escalar.

Dentre as técnicas utilizadas para projeto de quantizadores vetoriais, destaca-se o algoritmo LBG (Linde-Buzo-Gray), que muito embora seja o algoritmo mais utilizado para projeto de dicionários para quantização vetorial, apresenta alguns problemas e limitações comumente reportados.

Neste trabalho, que tem como principal objetivo o estudo de técnicas de projeto e avaliação de quantizadores vetoriais, são propostas duas técnicas para projeto de dicionários para quantização vetorial. A primeira técnica consiste na introdução de algumas modificações no algoritmo de Kohonen para treinamento de uma rede neural

de aspecto auto-organizativo. A segunda consiste na utilização da Análise de Componentes Principais. É importante ressaltar que tanto as redes neurais quanto a Análise de Componentes Principais vêm se destacando como poderosas ferramentas, por oferecerem excelentes resultados quando destinadas a uma grande variedade de aplicações.

Para avaliação dos algoritmos propostos, são realizadas simulações em sinais com distribuição gaussiana e de Gauss-Markov (que fornecem limites teóricos para o desempenho de quantizadores vetoriais), em forma de onda de voz e em imagens. O desempenho de cada técnica proposta é comparado ao desempenho apresentado pelo tradicional algoritmo LBG, ficando caracterizados diversos aspectos que evidenciam a superioridade dos algoritmos propostos.

A organização deste trabalho apresenta-se da seguinte forma:

No Capítulo 2, é abordado o princípio da compressão de sinais e são enfocados os principais fatores que fazem com que a compressão de sinais desempenhe um papel importantíssimo em diversas aplicações. Nesse contexto, é dada ênfase à quantização vetorial, com a devida formulação matemática dessa eficiente técnica. São apresentados os principais aspectos do projeto de um quantizador vetorial, a descrição do tradicional algoritmo LBG e importantes critérios para avaliação de desempenho de quantizadores vetoriais.

O Capítulo 3 tem como objetivo promover uma abordagem geral das redes neurais. São apresentados: os limites, as potencialidades e as particularidades referentes ao "duelo" dos homens versus computadores; a modelagem dos neurônios; os principais aspectos que caracterizam as redes neurais, com destaque para três - função de ativação, arquitetura da rede e regra de treinamento ou aprendizagem.

No Capítulo 4, que visa abordar as redes neurais de Kohonen aplicadas à quantização vetorial, são descritos importantes aspectos do algoritmo de Kohonen, tais como: inicialização e treinamento dos pesos, taxa de aprendizagem e raio de vizinhança. Nesse capítulo, ainda, são propostos e avaliados dois algoritmos, denominados KMTAU (Kohonen Modificado com Taxa de Aprendizagem Uniforme) e KMTAN (Kohonen Modificado com Taxa de Aprendizagem Não-Uniforme), resultantes da introdução de modificações no algoritmo de Kohonen. São apresentados os resultados das simulações realizadas em sinais com distribuições conhecidas, em sinais de voz e em

imagens, ficando evidenciadas a potencialidade e a eficiência dos métodos propostos, que apresentam desempenho superior ao apresentado pelo algoritmo LBG.

O Capítulo 5, que trata da Análise de Componente Principais aplicada à quantização vetorial, procura desenvolver o instrumental necessário para a perfeita compreensão dos fundamentos do algoritmo proposto, denominado PCA (*Principal Component Analysis*). Para tanto, são apresentadas: funções de autocorrelação e autocovariância, descrição de autovalores e autovetores e transformações ortogonais, com especial atenção para a Transformada de Karhunen-Loeve. O capítulo é finalizado com a descrição e a avaliação do algoritmo proposto em simulações realizadas com sinais de voz.

No Capítulo 6 são apresentadas as principais conclusões relacionadas aos algoritmos propostos e às simulações realizadas.

Capítulo 2

Quantização Vetorial

2.1 Princípios da Compressão Digital de Sinais

Um sinal analógico pode ser convertido para a forma digital através da combinação de três operações básicas: amostragem, quantização e codificação, como mostra o diagrama de blocos da Figura 2.1 [3].

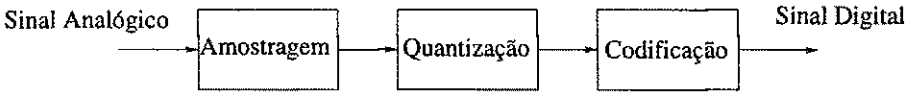


Figura 2.1: Conversão analógico/digital.

Na operação de amostragem, são tomados apenas valores de amostras do sinal analógico situadas em instantes de tempo uniformemente espaçados. O teorema da amostragem estabelece que para sinais limitados em faixa, por exemplo em $B\ Hz$, a taxa de amostragem f_s (em amostras por segundo) deve ser maior ou igual a $2B$. Assim, o intervalo de amostragem T deve ser: $T = \frac{1}{f_s} \leq \frac{1}{2B}$. No processo de quantização, cada valor de amostra é aproximado pelo nível mais próximo dentre um conjunto finito de níveis discretos.

Na etapa de codificação, cada nível de quantização é representado por uma palavra-código, que consiste de uma seqüência de dígitos binários.

A taxa de transmissão I de um sistema de comunicações digitais é o produto da taxa de amostragem f_s pelo número de bits por amostra R que o sistema utiliza para representar as amplitudes das amostras [1]:

$$I = f_s R \; bits \; por \; segundo \; (bit/s, \; ou \; bps) \tag{2.1}$$

Muito freqüentemente, quando B e f_s são conhecidos e fixos, a taxa de transmissão é simplesmente especificada pela taxa

$$R \; bits \; por \; amostra \; (bit/a) \tag{2.2}$$

Nesse contexto, a quantização vetorial surge como uma excelente técnica de compressão de sinais, visando reduzir a taxa de transmissão em um sistema de comunicações digitais.

A Figura 2.2 ilustra o princípio da compressão de sinais.

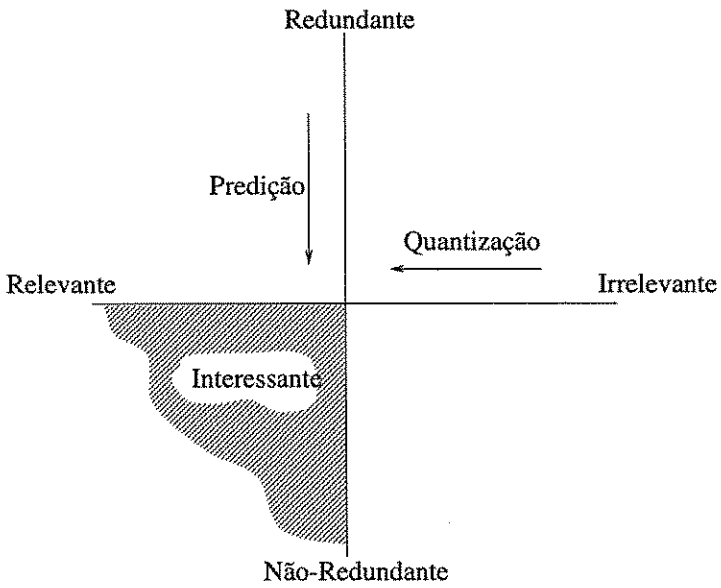


Figura 2.2: Princípio da compressão de sinais [1].

A informação dos sinais de interesse é composta das seguintes partes [1]:

- componente relevante;
- componente irrelevante, que corresponde à parcela supérflua da informação, de que o destino não necessita;
- componente não-redundante;
- componente redundante, que não necessita ser transmitida, uma vez que o receptor tem condições de reconstituí-la automaticamente.

A codificação digital de sinais visa reduzir as componentes redundante e irrelevante da informação, transmitindo apenas o que é imprescindível para o receptor, ou seja, a parte interessante da informação.

A parte irrelevante da informação é reduzida através do processo de quantização, que introduz perda da informação, de caráter irreversível. A parte redundante da informação é reduzida através de técnicas de predição e de transformação do sinal, que apresentam caráter reversível.

O principal objetivo da compressão de sinais é minimizar os requisitos de largura de faixa e de capacidade de armazenamento. Apesar de alguns sistemas não apresentarem grandes limitações de largura de faixa, como é o caso das redes de comunicações por fibra ótica, e embora a evolução tecnológica esteja continuamente contribuindo para o surgimento de memórias com grandes capacidades de armazenamento, a compressão de dados desempenha um papel importantíssimo, devido a uma série de fatores [1, 4]:

- um maior número de canais de comunicações pode ser multiplexado em sistemas de faixa larga, através do uso de compressão de dados para reduzir os requisitos de largura de faixa de cada sinal a ser multiplexado;
- algumas fontes de dados, tais como a televisão de alta definição (HDTV) e fontes que geram dados multiespectrais, podem gerar volumes de dados tão grandes a ponto de sobrepujarem até mesmo os sistemas ópticos de transmissão;
- a enorme utilização dos sistemas multimídia tem levado ao aumento da demanda no tocante ao armazenamento de voz, música, imagens, vídeo e dados em forma comprimida;

- em telefonia móvel celular, a largura de faixa é severamente limitada, o que tem motivado muitos estudos em compressão de voz;

- em tecnologias sem fio que envolvem imagem e vídeo e em *broadcasting* terrestre e aéreo de televisão digital de alta definição, a tecnologia de compressão de sinais também se faz necessária;

- nos sistemas de reconhecimento de fala e de locutor e nos sistemas de resposta vocal, vocabulários maiores podem ser armazenados através da redução dos requisitos de memória necessários para cada padrão de voz;

- nas redes digitais de serviços integrados (ISDN, *Integrated Services Digital Networks*), as técnicas de compressão permitem uma integração eficiente de sinais e dados.

Nos últimos anos, a quantização vetorial (QV) [4, 5] de diversas fontes de sinal, especialmente voz e imagem, tem despertado o interesse de vários pesquisadores, como técnica de compressão, devido aos seguintes motivos:

- de acordo com a Teoria da Distorção Versus Taxa, formulada por Shannon, é sempre possível obter um melhor desempenho codificando um bloco de amostras (isto é, um vetor de amostras) ao invés de codificar cada amostra individualmente [4, 6];

- os avanços tecnológicos, em especial os progressos obtidos no desempenho computacional VLSI (*Very Large Scale Integration*) e na capacidade de memória, tornaram possível a implementação de sistemas mais sofisticados de codificação e decodificação;

- as imposições tecnológicas no tocante à necessidade cada vez maior de compressão de dados, aspecto muito importante, por exemplo, nas Redes Digitais de Serviços Integrados (ISDN) e nos sistemas multimídia;

- a diversidade de sinais representados em computadores: sinais artificiais (como símbolos, textos, etc.) e sinais mais próximos dos que ocorrem no mundo físico

(por exemplo: sons, imagens, etc.); estes últimos tendem a ser mais imprevisíveis e difíceis de caracterizar analiticamente. Por esse motivo, esquemas flexíveis e eficientes para representação de sinais tornam-se cada dia mais importantes.

2.2 Quantização Vetorial

A quantização vetorial pode ser vista como um mapeamento Q de um espaço vetorial k-dimensional R^k em um subconjunto finito W de R^k:

$$Q : R^k \rightarrow W \tag{2.3}$$

onde $W = \{\mathbf{w}_i \mid i = 1, 2, ..., N\}$ é o conjunto dos vetores de reprodução e N é o número de vetores em W; cada $\mathbf{w}_i$ em W é chamado vetor-código e W é chamado dicionário do quantizador vetorial.

Na quantização vetorial, portanto, é definida uma partição $\mathcal{C} = \{\mathcal{C}_i : \bigcup \mathcal{C}_i = R^k,\ i = 1, 2, \ldots, N\}$ de R^k; um vetor $\mathbf{w}_i$ é escolhido em cada célula $\mathcal{C}_i$ de modo que seja o melhor vetor representativo para todos os vetores naquela região, segundo algum critério de distância/distorção mínima. A Figura 2.3 ilustra uma partição realizada no espaço R^2.

Para todo vetor de entrada $\mathbf{x}_l$, um vetor-código $\mathbf{w}_j$ em W é selecionado como a representação de $\mathbf{x}_l$. Esse processo é chamado fase de quantização (ou fase de procura no dicionário), denotada por $Q(\mathbf{x}_l) = \mathbf{w}_j$. O vetor-código $\mathbf{w}_j$ é então representado por alguns símbolos (normalmente, o endereço do vetor-código no dicionário) e transmitido ao longo do canal. Esse processo é chamado fase de codificação. No outro lado do canal, os símbolos recebidos são usados para selecionar os vetores-código do dicionário de reprodução do sinal de entrada. Esse processo é chamado fase de decodificação.

O número médio de bits requerido para representar os símbolos na fase de codificação é chamado taxa do quantizador, e o erro médio de quantização ao se representar o sinal de entrada $\mathbf{X}$ por sua versão quantizada $Q(\mathbf{X})$ é chamado distorção do quantizador.

Em se tratando de sinais de voz, quando se utiliza quantização vetorial com um dicionário de N vetores k-dimensionais, a taxa R é expressa por

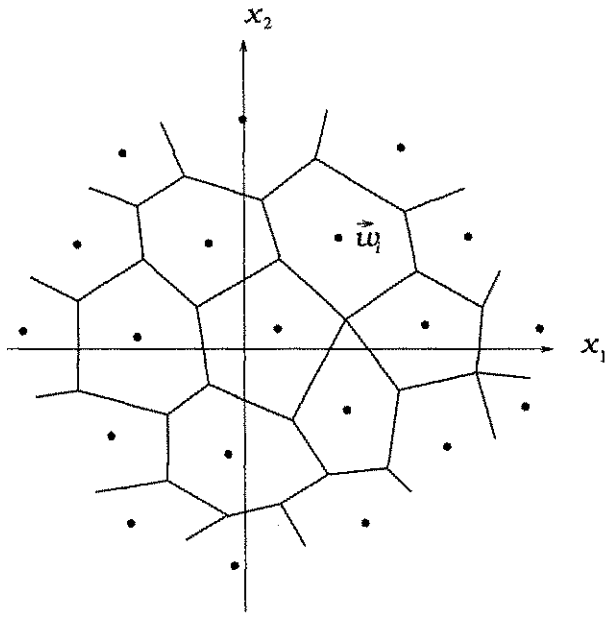


Figura 2.3: Partição promovida pela quantização vetorial.

$$R = \frac{1}{k} \log_2 N \; bit/amostra \tag{2.4}$$

Para o caso de quantização vetorial de sinais de imagem, a taxa R é expressa em $bit/pixel$.

O aumento do número de vetores-código no dicionário (e, conseqüentemente, da taxa) pode levar à redução da distorção. Uma das questões principais no projeto de quantizadores vetoriais é o compromisso entre taxa e distorção. O alvo da quantização vetorial é obter um dicionário ótimo que minimize a distorção média introduzida pela aproximação de cada vetor de entrada por um dos vetores-código. Tal quantizador ótimo deve satisfazer duas condições necessárias: a condição de vizinho mais próximo para partição e a condição de centróide para os vetores-código [4, 7].

2.2.1 A Complexidade na Quantização Vetorial

A etapa de representação de um vetor de entrada por um dos vetores-código, ou seja, a etapa de determinação do vizinho mais próximo, representa uma das principais

limitações da QV, isto é: a complexidade computacional cresce exponencialmente com a dimensão para uma dada taxa, conforme será demonstrado a seguir.

A relação entre o número de representantes (ou níveis, em analogia com a quantização escalar) do dicionário N, a dimensão k e a taxa R pode ser expressa como $N = 2^{kR}$, onde a taxa R é expressa em *bit/amostra* para voz e em *bit por pixel* (*bpp*) para imagem. Para codificar um dado vetor de entrada é necessário encontrar sua distância para cada um dos N vetores-código e depois comparar as distâncias de modo a encontrar o vetor-código mais semelhante, ou seja, o vizinho mais próximo. A codificação de cada vetor de entrada requer, portanto, 2^{kR} cálculos de distância e $2^{kR} - 1$ comparações. No caso da medida de distorção de erro quadrático, isto é,

$$d(\mathbf{x}, \mathbf{w}_i) = \sum_{j=1}^{k} [x_i - w_{ij}]^2 \tag{2.5}$$

onde w_{ij} é a j-ésima componente do vetor-código $\mathbf{w}_i$ e x_j é a j-ésima componente do vetor de entrada $\mathbf{x}$, cada cálculo de distância requer k multiplicações, k subtrações e $k - 1$ adições. Portanto, para codificar cada vetor de entrada, $k2^{kR}$ multiplicações, $k2^{kR}$ subtrações, $(k-1)2^{kR}$ adições e $2^{kR} - 1$ comparações devem ser computadas. A complexidade de um quantizador vetorial pode ser alternativamente expressa em termos de 2^{kR} multiplicações, 2^{kR} subtrações e $(1 - \frac{1}{k})2^{kR}$ adições e $(2^{kR} - 1)/k$ comparações por amostra.

Em resumo, a complexidade computacional de um quantizador vetorial de dimensão k e taxa R requer um número de computações por amostra da ordem de 2^{kR} para cada vetor de entrada se uma busca completa é realizada ao longo do dicionário. Além disso, é necessária uma memória de $k2^{kR}$ palavras.

Diversas abordagens [8, 9, 10] têm sido propostas para otimizar o processo de codificação.

2.2.2 Técnicas para Projeto de Dicionários para Quantização Vetorial

De forma genérica, o projeto de um quantizador vetorial deve levar em consideração o intercâmbio de vários parâmetros:

- taxa;
- qualidade;
- complexidade;
- custo;
- atraso.

Ao se reduzir a taxa R, a qualidade naturalmente sofre uma queda. Para uma mesma taxa, entretanto, o aumento da dimensão k do quantizador implica aumento de qualidade; a complexidade, no entanto, também aumenta, uma vez que o número de operações envolvidas é da ordem de 2^{kR}. A complexidade aumenta o custo e, em muitos sistemas de codificação, aumenta o atraso [11]. Em algumas aplicações que envolvem armazenamento de voz, como por exemplo *voice mail*, o atraso não constitui um problema. Entretanto, em sistemas de transmissão de voz, pode ser incômodo o atraso entre o instante de tempo em que o usuário pronuncia uma sentença e o instante de tempo em que uma versão codificada desta entra no sistema de comunicações.

Existem diversas técnicas para projeto de dicionários para quantização vetorial, dentre as quais podem ser citadas:

- algoritmo LBG, Linde-Buzo-Gray [12];
- rede neural de Kohonen [13, 14, 15];
- algoritmos competitivos de redes neurais [16, 17, 18, 19];
- *stochastic relaxation* [7];
- algoritmos *fuzzy* [20, 21];
- algoritmos genéticos [22];
- algoritmo PNN, *pair-wise nearest neighbor* [23].

Por se tratar da técnica mais popular e amplamente utilizada para projeto de quantizadores vetoriais, o algoritmo LBG será descrito a seguir.

2.2.3 Algoritmo LBG

Também conhecido como GLA (*Generalized Lloyd Algorithm*) ou algoritmo das k-médias, o algoritmo LBG [4, 12] é, atualmente, o mais utilizado para projeto de dicionários para quantização vetorial, utilizando para tanto uma seqüência típica de dados para treinamento do dicionário.

O algoritmo LBG consiste da seguinte seqüência de passos:

- *Passo 1*) condição inicial: inicialize com qualquer configuração inicial desejada. Vá para *passo 2*) se a inicialização ocorreu com um conjunto de vetores-código (dicionário inicial); vá para *passo 3*) se a inicialização ocorreu com a partição dos dados;
- *Passo 2*) particionamento: aloque cada dado (ou vetor de entrada) na respectiva classe segundo o critério do vetor-código mais próximo;
- *Passo 3*) atualização do dicionário: compute os novos vetores-código como os centróides das classes de dados;
- *Passo 4*) teste de convergência: alterne *passo 2*) e *passo 3*) até a convergência do processo.

Em essência, no algoritmo LBG a função distorção decresce monotonicamente, uma vez que o dicionário é iterativamente atualizado visando satisfazer as condições de centróide e de vizinho mais próximo. Infelizmente, como a função distorção é geralmente não convexa e pode conter múltiplos mínimos locais [24], o algoritmo LBG freqüentemente produz dicionários que não são ótimos.

Existem alguns problemas apresentados pelo algoritmo LBG, comumente reportados [22, 25, 26, 27, 28]:

- algumas palavras-código podem ser sub-utilizadas e, em casos extremos, até mesmo nunca serem acessadas, ou seja, na fase de quantização, alguns vetores do dicionário podem ser pouco ou até mesmo nunca acessados;

- o algoritmo converge para um dicionário localmente ótimo sob determinadas condições, entretanto, converge para um dicionário diferente quando uma condição inicial diferente é aplicada;
- a velocidade de convergência e o desempenho do dicionário final dependem do dicionário/partição inicial.

2.3 Avaliação de Desempenho

De forma geral, as medidas de avaliação de desempenho enquadram-se em duas classes:

- medidas de qualidade objetivas;
- medidas de qualidade subjetivas.

As medidas subjetivas baseiam-se em comparações, entre o sinal original e o sinal processado, realizadas por um grupo de pessoas, que subjetivamente classificam a qualidade do sinal processado segundo uma escala pré-determinada. As medidas objetivas, por outro lado, baseiam-se numa comparação matemática entre os sinais original e processado [29].

2.3.1 Medidas Subjetivas

Muito embora os métodos de avaliação objetiva sejam de fácil implementação e possam ser repetidos inúmeras vezes durante o desenvolvimento e avaliação de um algoritmo de processamento de voz, áudio ou imagem, as medidas de avaliação subjetivas desempenham um papel importantíssimo como instrumental necessário para avaliar de forma plena o desempenho dos sistemas de processamento de voz, áudio ou imagem. Cumpre salientar, entretanto, que as medidas subjetivas são bastante dispendiosas de tempo e de difícil implementação, pois utilizam um número relativamente grande de pessoas.

Realizadas através de testes de escuta ou testes de visualização, as medidas de avaliação subjetivas são baseadas nas opiniões individuais de cada pessoa participante do teste. Conforme [30], são utilizadas pessoas de diferentes formações profissionais e possíveis usuários do sistema a ser avaliado. Devem ser utilizadas pelo menos 15 pessoas que não sejam especialistas da área do projeto. Uma sessão de avaliação não deve durar mais do que 30 minutos e começar mostrando para os avaliadores alguns exemplos típicos de imagem, voz, ou áudio com diferentes escalas de degradação, de forma a permitir aclimatar os avaliadores. Deve ser observado, entretanto, que os exemplos utilizados não devem interferir no julgamento.

A seguir, serão apresentadas, de forma breve, algumas das técnicas mais usuais de avaliação subjetiva de qualidade aplicadas a sistemas de voz, áudio ou imagem [1, 30, 31].

Testes de Preferência

Os testes de preferência são realizados por comparação entre pares. Uma forma de avaliação consiste em se conceder um conceito a cada uma das possibilidades de comparação, ou seja:

- A - A qualidade do primeiro é melhor do que a do segundo;
- B - A qualidade do segundo é melhor do que a do primeiro;
- C - A qualidade de ambos não se distingue.

Esse teste é normalmente utilizado para avaliação de sistemas que tenham características próximas. É mais indicado para a avaliação de sistemas de áudio. Para a avaliação de imagens, é comum a utilização de um teste de preferência no qual a comparação é feita com relação a uma imagem de referência. Esse teste utiliza uma escala de graduação com valores que variam de um a cinco, onde cada valor correspondente a um adjetivo do processo de comparação:

- 5 - A imagem sob teste tem qualidade muito superior à de referência;
- 4 - A imagem sob teste tem qualidade um pouco superior à de referência;

Escore	Qualidade	Nível de Distorção
5	Excelente	Imperceptível
4	Boa	Apenas perceptível mas não incômodo
3	Razoável	Perceptível e pouco incômodo
2	Pobre	Incômodo
1	Ruim	Muito incômodo

Tabela 2.1: Escala para o teste MOS.

- 3 - A imagem sob teste tem a mesma qualidade da de referência.
- 2 - A imagem sob teste tem qualidade um pouco inferior à de referência;
- 1 - A imagem sob teste tem qualidade muito inferior à de referência.

Normalmente, são recomendados 20 segundos de cadência de apresentação por par e 10 segundos de intervalo entre pares.

Teste de Qualidade Absoluta: Escore Médio de Opinião (MOS)

Uma medida bastante utilizada para avaliação de desempenho de sistemas de compressão de voz denomina-se MOS (*Mean Opinion Score*). No teste MOS, cada avaliador atribui um escore de qualidade segundo a escala graduada ilustrada na Tabela 2.1. É calculada a média aritmética dos escores obtidos e determinado o valor final da avaliação segundo a Tabela 2.1 [29].

2.3.2 Medidas Objetivas

As medidas objetivas desempenham importante papel para avaliação de desempenho de algoritmos de compressão de sinais, devido a vários fatores:

- permitem avaliação rápida;
- são mais facilmente implementadas e muito menos dispendiosas em tempo, relativamente às medidas subjetivas;

- podem ser repetidas inúmeras vezes durante a fase de desenvolvimento de um algoritmo;

- podem auxiliar mais fácil e efetivamente o ajuste de parâmetros de algoritmos e até mesmo servir como um meio de detecção de diferenças sutis no desempenho dos algoritmos, possivelmente não detectáveis por meio de avaliação subjetiva.

Como as medidas objetivas realizam comparação direta entre os sinais original e processado, é necessário que tais sinais estejam sincronizados. Para sinais de voz, o método mais comum de implementação de medidas objetivas é através da partição do sinal em janelas de tempo de duração típica de 10 a 30 ms seguida do cálculo de uma medida de distância/distorção para cada janela [29].

Para serem úteis, as medidas de distorção objetivas devem apresentar no mínimo duas características. Primeiramente, devem ter significado subjetivo no sentido de que pequenas e grandes distorções objetivas correspondam a qualidades subjetivas boas e ruins, respectivamente. Segundo, elas devem ser tratáveis, no sentido de que sejam amenas sob o ponto de vista da análise matemática e levem a técnicas de projeto facilmente implementáveis [32].

A seguir, serão brevemente descritas as medidas de distorção SNR e SNR_{seg}, por se tratarem de medidas muito utilizadas para avaliação de desempenho de sistemas de compressão de voz.

Relação Sinal-Ruído

A relação sinal-ruído, SNR (*Signal to Noise Ratio*), e suas variantes são apropriadas apenas para avaliação de sistemas de codificação e remoção de ruído que procuram reproduzir a forma de onda do sinal de entrada original.

Sejam $x(n)$ o sinal original, $y(n)$ o sinal processado e $e(n) = x(n) - y(n)$ o sinal erro no instante de tempo n.

A energia contida no sinal original é:

$$E_x = \sum_n x^2(n) \tag{2.6}$$

A energia contida no sinal erro é:

$$E_e = \sum_n e^2(n) = \sum_n [x(n) - y(n)]^2 \tag{2.7}$$

A medida SNR resultante, em dB, é dada por

$$SNR = 10 \log_{10} \frac{E_x}{E_e} = 10 \log_{10} \frac{\sum_n x^2(n)}{\sum_n [x(n) - y(n)]^2} \tag{2.8}$$

A seguir, será apresentada uma importante variação da medida de distorção clássica SNR: a relação sinal-ruído segmental, SNR_{seg}.

Relação Sinal-Ruído Segmental

Apesar da simplicidade matemática, a medida SNR apresenta uma incômoda limitação: a igual ponderação de todos os erros no domínio do tempo. Deste modo, por exemplo, um indesejável e inevitável alto valor de SNR pode ser obtido se uma seqüência de fala apresentar alta concentração de segmentos vocais, sonoros, de alta energia, uma vez que o efeito do ruído é maior nos segmentos de baixa energia, como por exemplo os sons fricativos surdos.

Uma medida de qualidade mais refinada pode ser obtida se for tomada a média da SNR medida em curtos intervalos de tempo. É definida, então, a relação sinal-ruído segmental:

$$SNR_{seg}(dB) = E[SNR(j)\ (dB)] \tag{2.9}$$

onde $SNR(j)$ é SNR convencional para o segmento ou janela de tempo j.

SNR_{seg} é formulada da seguinte forma:

$$SNR_{seg} = \frac{1}{J} \sum_{j=0}^{J-1} 10 \log_{10} [\sum_{n=m_j-N-1}^{m_j} \frac{x^2(n)}{[x(n) - y(n)]^2}], \tag{2.10}$$

onde $m_0, m_1, ..., m_{J-1}$ são os instantes finais para as J janelas de tempo de N amostras, de comprimento típico de 15 a 25 ms [29, 30].

Relação Sinal Ruído de Pico ($PSNR$)

Existem diversas medidas para avaliação da qualidade de imagens processadas [33]. A relação sinal-ruído de pico ($PSNR$), entretanto, é bastante utilizada para a avaliação de imagens comprimidas, muito embora não seja bastante correlacionada com critérios de avaliação subjetivos. Ela é definida como 10 vezes o logaritmo na base 10 da razão entre o quadrado do valor de pico da amplitude de entrada (v_p^2) e o erro médio quadrático (MSE, *Mean Square Error*).

$$PSNR = 10 \log_{10} \left[\frac{(v_p^2)}{MSE} \right] \tag{2.11}$$

Para o caso de uma imagem original codificada a 8 *bpp*,

$$PSNR = 10 \log_{10} \left[\frac{255^2}{MSE} \right] \tag{2.12}$$

Fonte de Gauss-Markov

Uma das fontes mais utilizadas para avaliação e comparação de quantizadores vetoriais é a fonte de Gauss-Markov de 1^a. ordem, também conhecida como fonte de Markov de 1^a. ordem ou fonte AR(1) [34]. Para efeito de simplicidade, essa fonte será referenciada como fonte de Gauss-Markov ao longo de todo o texto.

O processo discreto de Gauss-Markov $\{X(n)\}$ é definido pela equação

$$x(n) = ax(n-1) + w(n), \ \forall n \tag{2.13}$$

onde $\{w(n)\}$ é uma seqüência de variáveis aleatórias com distribuição gaussiana, independentes e identicamente distribuídas, com média zero.

É importante observar que o processo de Gauss-Markov pode ser obtido através da passagem do ruído gaussiano branco $\{W(n)\}$ através de um filtro com resposta ao impulso [1]

$$h(n) = \begin{cases} a^n, n \geq 0 \\ 0, n < 0 \end{cases} \tag{2.14}$$

Sejam σ_x^2 e σ_w^2 as variâncias do processo de Gauss-Markov $\{X(n)\}$ e do ruído branco $\{W(n)\}$, respectivamente.

O limite teórico para o desempenho do quantizador vetorial no processo de Gauss-Markov é dado pela Teoria da Distorção versus Taxa de Shannon [6]:

$$D_S(R) = \sigma_w^2 2^{-2R} \tag{2.15}$$

onde $D_S(R)$ é a menor distorção obtenível para a taxa R, expressa em $bit/amostra$.

Seja $e(n) = X(n) - Q(X(n))$ o erro de quantização obtido ao se representar $X(n)$ por sua versão quantizada $Q(X(n))$. Sejam σ_e^2 a variância do erro de quantização e $\sigma_{e\,\min}^2$ o menor valor possível para σ_e^2.

Da equação 2.15, obtém-se:

$$\sigma_{e\,\min}^2 = \sigma_w^2 2^{-2R} \tag{2.16}$$

Da equação 2.13,

$$\sigma_w^2 = (1 - a^2)\sigma_x^2 \tag{2.17}$$

Logo,

$$\sigma_{e\,\min}^2 = (1 - a^2)\sigma_x^2 2^{-2R} \tag{2.18}$$

A relação sinal-ruído de quantização, SNR, é dada por

$$SNR = 10 \log_{10} \frac{\sigma_x^2}{\sigma_e^2} \tag{2.19}$$

O valor máximo de SNR, $SNR_{\max}$, é obtido da equação 2.19, fazendo $\sigma_e^2 = \sigma_{e\,\min}^2$. Desse modo,

$$SNR_{\max} = 10 \log_{10} \frac{\sigma_x^2}{\sigma_{e\,\min}^2} = 10 \log_{10} \frac{\sigma_x^2}{(1 - a^2)\sigma_x^2 2^{-2R}} \tag{2.20}$$

Portanto,

$$SNR_{\max} = 10 \log_{10} \frac{2^{2R}}{1 - a^2} \tag{2.21}$$

Entropia Normalizada

Nesta seção será descrita a entropia normalizada dos vetores-código, proposta como medida de avaliação da uniformidade de distribuição dos vetores da fonte nas regiões de Voronoi.

Seja $W = \{\mathbf{w}_i \mid i = 1, 2, ..., N\}$ o dicionário de tamanho N, onde $\mathbf{w}_i = (w_{ij},\ j = 1, ..., k)$ é um vetor k-dimensional. Seja p_i a probabilidade de que um dado vetor de entrada $\mathbf{x}$ pertença à região de Voronoi correspondente ao vetor-código $\mathbf{w}_i$. Em outras palavras, p_i corresponde à probabilidade de que um vetor-código $\mathbf{w}_i$ seja o vetor-código mais próximo para um dado vetor $\mathbf{x}$ qualquer.

Seja $S = \{\mathbf{x}_p,\ p = 1, ..., n\}$ uma longa seqüência representativa de vetores usados para o treinamento do quantizador. Ao final do treinamento, o conjunto S é particionado em N subconjuntos S_i, $i = 1, ..., N$, onde $S_i = \{\mathbf{x}_p : \mathbf{w}_i = Q(\mathbf{x}_p)\}$, ou seja, $\mathbf{w}_i$ é o vetor-código mais próximo de todos $\mathbf{x}_p \in S_i$.

Se n_i é o tamanho da região S_i, a probabilidade de que $\mathbf{w}_i$ seja o vetor-código mais próximo para qualquer $\mathbf{x}_p$ pode ser obtida como:

$$p_i = \frac{n_i}{n} \text{ para } n \to \infty \tag{2.22}$$

Os p_i's obtidos com o uso de uma longa seqüência representativa de vetores de treino serão utilizados para o cálculo da entropia H_k dos vetores-código, definida como:

$$H_k = \sum_{i=1}^{N} p_i \log_2(\frac{1}{p_i}) \tag{2.23}$$

Para avaliação da uniformidade de distribuição dos vetores da fonte nas regiões de Voronoi, será utilizada a entropia normalizada dos vetores-código, definida como:

$$H_n = \frac{H_k}{\log_2 N} \tag{2.24}$$

ou seja,

$$H_n = \frac{\sum_{i=1}^{N} p_i \log_2(\frac{1}{p_i})}{\log_2 N} \tag{2.25}$$

Para vetores-código equiprováveis, ocorre máxima distribuição de entropia no dicionário, de modo que $H_n = 1$, uma vez que para essa condição $H_k = \log_2 N$. Convém salientar que $H_n \to 1$ à medida que aumenta a uniformidade de distribuição dos vetores da fonte nas regiões de Voronoi, pois nesse caso $H_k \to \log_2 N$.

Capítulo 3

Redes Neurais

3.1 Introdução

Antes de uma abordagem de caráter eminentemente matemático, cumpre tratar de uma polêmica discussão: as potencialidades, os limites e as particularidades referentes ao "duelo" dos homens versus computadores.

Como os computadores conseguem realizar, em frações de segundos, algumas tarefas que os seres humanos realizam com certa demora, como por exemplo somar números com muitos algarismos, espera-se que os computadores sejam parecidos com os homens em muitas outras atividades. Dessa forma, um desapontamento natural surge ao se perceber que os computadores não desempenham de forma fácil, plena e eficiente algumas tarefas que os seres humanos realizam com naturalidade, como por exemplo o reconhecimento de imagens. Esse é um dos problemas que os pesquisadores da área de Inteligência Artificial procuram atacar, entretando, mesmo depois de vários anos de pesquisa, os resultados são insuficientes para a concepção de sistemas computacionais artificialmente inteligentes no sentido que se deseja.

Os computadores são projetados para desempenhar uma instrução após outra, com extrema velocidade. Dessa forma, em atividades essencialmente seriais, como por exemplo contagem e soma, o computador supera o cérebro. Por outro lado, em problemas de natureza paralela, que solicitam o processamento e a interação de muitas informações, o cérebro supera o computador: é o que ocorre, por exemplo, nos problemas de visão e

reconhecimento de voz. O conhecimento requerido para solucionar esses problemas vem de informações diferentes, cada uma contribuindo para o resultado final. O cérebro, com sua natureza paralela, é capaz de representar e armazenar esse conhecimento de forma acessível.

O objetivo das redes neurais é capturar os princípios que regem o funcionamento do cérebro ao solucionar tais problemas. Será discutido mais adiante como o estudo dos sistemas neuronais reais tem permitido a modelagem do paralelismo existente no cérebro na forma de redes neurais artificiais [15, 16].

Além do paralelismo, os sistemas neuronais reais têm outras características interessantes. Talvez uma das características mais importantes seja a capacidade que o cérebro tem de aprender sozinho. Quando crianças, os seres humanos aprendem através de exemplos: aprendem a falar, comer, beber e até mesmo desenvolver os conceitos próprios de moral. O mesmo não é verdade para os sistemas computacionais convencionais. Seria maravilhoso se, ao invés de se desenvolver um programa para executar uma tarefa, fosse possível apenas deixar o computador observar a execução de tal tarefa de modo que ele aprendesse através do exemplo [2].

3.2 Estrutura do Cérebro

Atualmente, muito se conhece sobre a estrutura do cérebro, que contém cerca de 10^{10} unidades básicas, chamadas de neurônios. Cada neurônio conecta-se a cerca de 10^4 outros.

Uma representação de um neurônio é mostrada na Figura 3.1. O soma é o corpo do neurônio. Presos ao soma estão filamentos longos e irregulares chamados dendritos, que atuam como conexões através das quais todas as entradas chegam ao neurônio. Essas células são capazes de desempenhar funções mais complexas que uma simples adição das entradas que recebem, entretanto, a adição é uma aproximação razoável. Um outro tipo de terminação nervosa presa ao soma é o axônio que, ao contrário do dendrito, é eletricamente ativo e funciona como canal de saída do neurônio [2].

O axônio é um dispositivo não-linear, que opera com um limiar. O axônio produz um pulso de tensão, chamado potencial de ação, quando o potencial do soma atinge

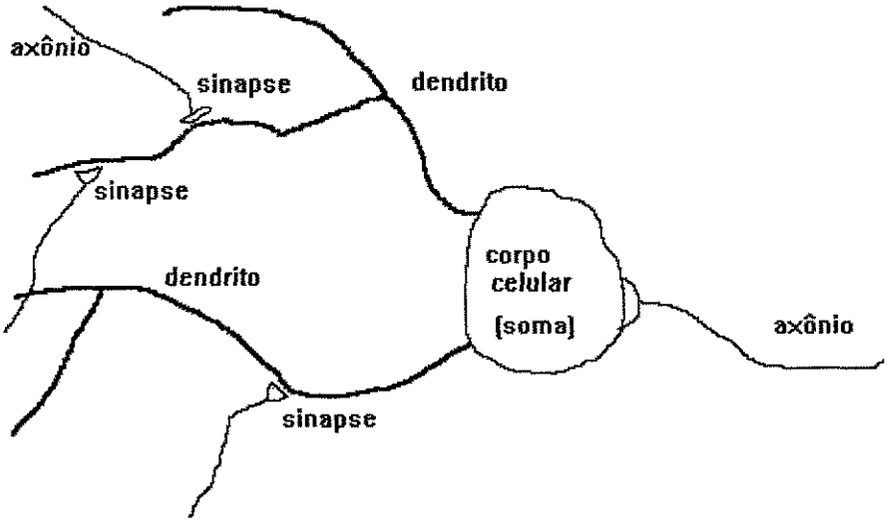


Figura 3.1: Neurônio real.

um determinado valor limiar. O axônio termina num contato especializado chamado sinapse, que acopla o axônio ao dendrito de outra célula. Não há ligação direta ao longo da junção; ao contrário, trata-se de uma ligação química temporária. A sinapse libera os neurotransmissores quando seu potencial é suficientemente aumentado pelo potencial de ação. Cada dendrito apresenta muitas sinapses agindo sobre si, o que permite interconectividade massiva. Convém salientar que algumas sinapses excitam o dendrito que atingem, ao passo que outras o inibem. Um único neurônio apresenta muitas sinapses de entrada em seus dendritos e muitas sinapses de saída conectando-o a outras células.

3.3 Cérebro versus Computador: Diferenças e Coincidências

Conforme abordado na introdução deste capítulo, o cérebro desempenha de forma bastante satisfatória algumas atividades difíceis de ser desempenhadas de forma plena e eficiente através de computador, como por exemplo visão, reconhecimento de voz e

aprendizagem através de exemplos. Por outro lado, existem tarefas, como é o caso da soma de números com muitos algarismos, que são executadas mais eficientemente pelo computador.

O cérebro apresenta uma estrutura extremamente desenvolvida, caracterizada pela existência de um sistema massivamente paralelo, onde um considerável número de pequenas unidades de processamento dividem o trabalho, ao contrário de deixar o trabalho a cargo de uma única unidade extremamente rápida, como ocorre no computador. Essa divisão do trabalho apresenta várias vantagens. Como muitos neurônios estão envolvidos ao mesmo tempo, a contribuição de um único neurônio não se torna tão importante, de modo que se um neurônio falhar os demais não se tornam significativamente comprometidos. Essa divisão do trabalho, conhecida como processamento distribuído, portanto, apresenta a importante característica de ser, de certa forma, tolerante a falhas. De fato, como o cérebro é capaz de aprender, também é capaz de se adaptar inclusive à perda de alguns neurônios; apesar de perder neurônios diariamente, o cérebro continua desempenhando suas funções como se nada tivesse ocorrido. O computador, por sua vez, apresenta uma única unidade trabalhando arduamente para realizar inúmeros cálculos com extrema rapidez, de modo que se essa unidade quebrar, não há como dar continuidade à realização da tarefa, não há como transferir tal tarefa para outro local. O computador apresenta uma estrutura bastante diferente daquela apresentada pelo cérebro. Ao invés de se estruturar com inúmeras unidades de processamento extremamente interconectadas, ele consiste de um (ou eventualmente um pouco mais de um) excepcionalmente rápido processador, capaz de uma enorme quantidade de cálculos por segundo, o que o torna excelente no desempenho de ações repetitivas, mas fracos no desempenho de grandes quantidades de diferentes tipos de dados, requeridos, por exemplo, num sistema visual [2].

Eventos em um chip de silicone ocorrem na faixa de nanosegundos enquanto que eventos em redes de neurônios humanos ocorrem na faixa de milisegundos. Entretanto, a interconexão massiva dos neurônios confere ao cérebro a execução de tarefas que exigem processamento de informações, como por exemplo o reconhecimento de objetos, mais rápida e eficientemente que um computador.

Do ponto de vista do funcionamento, cérebro e computador exibem algumas coin-

cidências. O computador sai de fábrica com algumas instruções para operações básicas do sistema já armazenadas na memória ROM. Os seres humanos também já nascem com algumas regiões do cérebro em sua configuração definitiva, como aquelas destinadas à respiração, às batidas do coração e ao metabolismo em geral.

Nos seres humanos, é a experiência de vida que vai moldando a rede neuronal, com a formação de novas sinapses ou eliminação das que são inúteis. Nesse sentido, o cérebro não faz distinção entre *hardware* e *software*, ou seja, o cérebro tem a capacidade de se autoprogramar por meio da formação de novas sinapses.

Uma das diferenças mais marcantes entre cérebro e computador talvez seja a maior capacidade de miniaturização do primeiro. O cérebro humano tem aproximadamente 10^{10} neurônios. Algumas dessas células, como as de Purkinje, podem alcançar até 90000 conexões sinápticas com outros neurônios. Em termos de comparação, um chip de 80486 tem apenas 1,2 milhão de transistores e cada um deles se liga diretamente no máximo a meia dúzia de outros componentes vizinhos. Uma CPU pode acessar apenas uma posição de memória a cada vez, embora possa realizar essa tarefa rapidamente. Uma célula de Purkinje está acessando, a cada momento, outras 90000 simultaneamente. Isso confere ao cérebro a capacidade de processar milhares de informações recebidas em tempo real.

3.4 Modelagem dos Neurônios

Nesta seção serão descritos os elementos processadores básicos de uma rede neural, comumente denominados neurônios ou nós.

Um neurônio é uma unidade de processamento da informação, fundamental para a operação da rede neural. A Figura 3.2 mostra o modelo de um neurônio. Três elementos básicos podem ser identificados no modelo [15]:

1. Um conjunto de *sinapses*, cada uma caracterizada por um *peso*. Especificamente, uma entrada x_j via sinapse j conectada ao neurônio k é multiplicada pelo peso sináptico w_{kj}. Note que na notação w_{kj}, o primeiro subscrito diz respeito ao neurônio em questão enquanto que o segundo diz respeito à entrada à qual o peso

sináptico se refere. Em outras palavras w_{kj} representa o peso correspondente à j-ésima entrada do neurônio k. O peso w_{kj} é positivo se a sinapse a ele associada for excitatória; é negativo se a sinapse for inibitória.

2. Um *somador* para realizar uma soma ponderada das entradas (os pesos sinápticos são os termos de ponderação); neste sentido, portanto, a operação descrita constitui um *operador linear.*

3. Uma *função de ativação* para limitar a amplitude da saída de um neurônio.

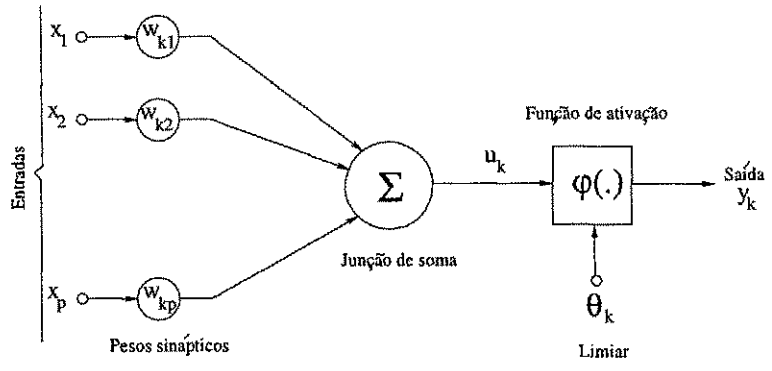


Figura 3.2: Modelo do neurônio.

O modelo apresentado na Figura 3.2 também inclui um *limiar* θ_k para a função de ativação φ. Tipicamente, a saída de um neurônio está compreendida no intervalo $[0, 1]$ ou alternativamente $[-1, 1]$.

Matematicamente, o neurônio k pode ser descrito pelo par de equações abaixo:

$$u_k = \sum_{j=1}^{p} w_{kj} x_j \tag{3.1}$$

e

$$y_k = \varphi(u_k - \theta_k) \tag{3.2}$$

onde $x_1, x_2, ..., x_p$ são os sinais de entrada; $w_{k1}, w_{k2}, ..., w_{kp}$ são os pesos sinápticos do neurônio k; u_k é a saída da combinação linear; θ_k é o limiar; e y_k é a saída do neurônio.

Observando o par de equações 3.1 e 3.2, é possível reformular o modelo do neurônio, através de uma simples manipulação matemática:

Seja

$$v_k = u_k - \theta_k \tag{3.3}$$

Assim,

$$v_k = \sum_{j=1}^{p} w_{kj} x_j - \theta_k \tag{3.4}$$

ou ainda,

$$v_k = \sum_{j=0}^{p} w_{kj} x_j \tag{3.5}$$

Dessa forma,

$$y_k = \varphi(v_k) \tag{3.6}$$

Na equação 3.5 uma sinapse foi implicitamente definida, cuja entrada é

$$x_0 = -1 \tag{3.7}$$

e cujo peso é

$$w_{k0} = \theta_k \tag{3.8}$$

Deste modo, o modelo do neurônio pode ser reformulado, conforme apresenta a Figura 3.3 [15].

3.4.1 Tipos de Função de Ativação

A função de ativação [15], denotada por $\varphi(.)$, define a saída de um neurônio em termos do nível de atividade de sua entrada. Existem três tipos básicos de função de ativação:

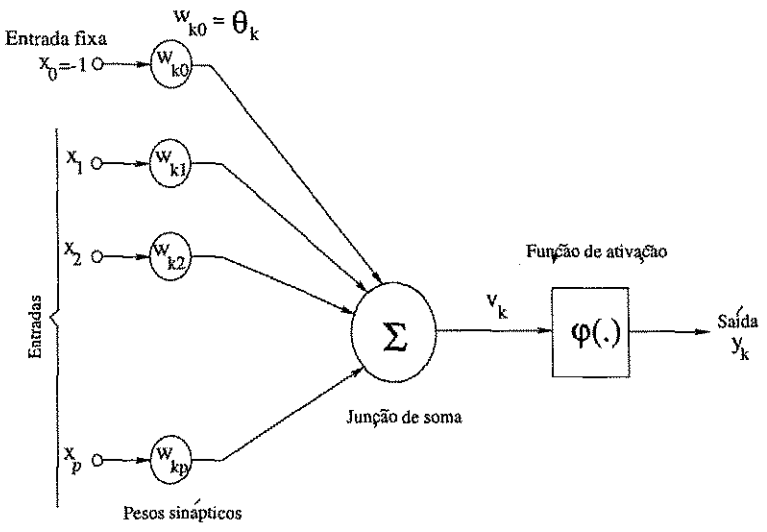


Figura 3.3: Modelo do neurônio incluindo o limiar como peso sináptico.

1. Função limiar, também denominada limitador brusco;
2. Função linear por partes, também denominada limitador lógico;
3. Função sigmóide.

A função limiar, mostrada na Figura 3.4, é definida como:

$$\varphi(v) = \begin{cases} 1, v \geq 0 \\ 0, v < 0 \end{cases} \tag{3.9}$$

Desta forma, a saída do neurônio k que emprega o limitador lógico é:

$$y_k = \begin{cases} 1, v_k \geq 0 \\ 0, v_k < 0 \end{cases} \tag{3.10}$$

onde v_k é o nível de atividade interna do neurônio, ou seja,

$$v_k = \sum_{j=1}^{p} w_{kj} x_j - \theta_k \tag{3.11}$$

Tal neurônio é referenciado na literatura como modelo de McCulloch-Pitts, reconhecimento ao trabalho pioneiro realizado por McCulloch e Pitts (1943) [15]. Nesse

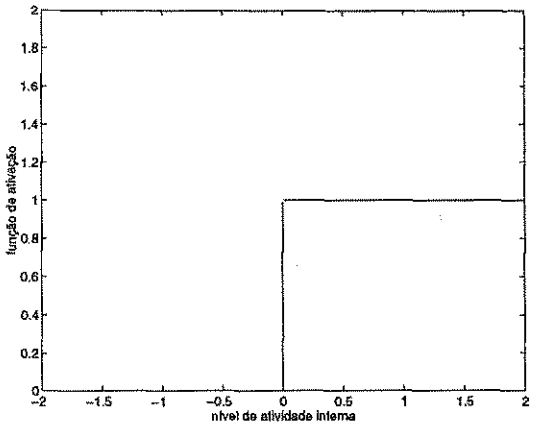


Figura 3.4: Função limiar.

modelo, a saída do neurônio assume valor 1 se o nível de atividade interna do neurônio é não-negativo e 0 caso contrário.

Para a função linear por partes, mostrada na Figura 3.5, tem-se:

$$\varphi(v) = \begin{cases} 1, v \geq \frac{1}{2} \\ v + \frac{1}{2}, \frac{1}{2} > v > -\frac{1}{2} \\ 0, v \leq -\frac{1}{2} \end{cases} \tag{3.12}$$

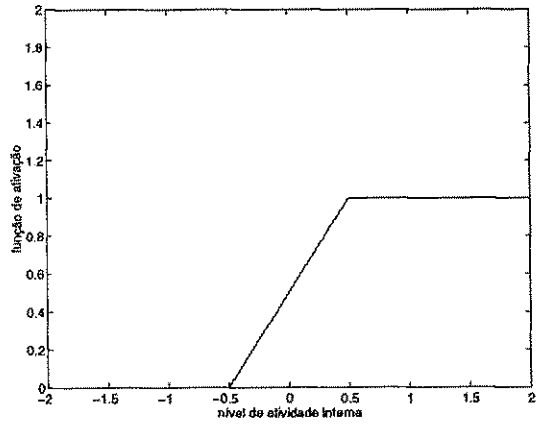


Figura 3.5: Função linear.

A função sigmóide é a mais utilizada como função de ativação em redes neurais. Um exemplo de função sigmóide é a função logística, definida como:

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \tag{3.13}$$

onde a é o parâmetro inclinação da função sigmóide. Variando esse parâmetro, é possível obter sigmóides com diferentes inclinações, como mostra a Figura 3.6. É importante perceber que à medida que a inclinação tende a infinito, a função sigmóide se aproxima da função limiar. Enquanto a função limiar assume valor 0 ou 1, a função sigmóide assume uma faixa contínua de valores. Além disso, a função sigmóide, ao contrário da função limiar, é diferenciável. Convém mencionar que diferenciabilidade é uma característica importante na teoria de redes neurais.

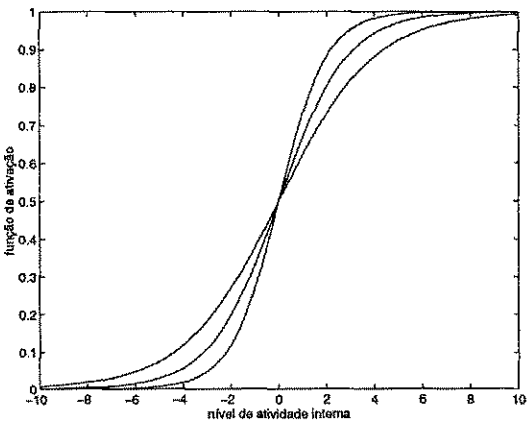


Figura 3.6: Função sigmóide.

3.5 Caracterização de Redes Neurais

É bastante difícil conceber uma definição completa e satisfatória de redes neurais, tendo em vista a diversidade de redes neurais existentes, o enorme número de aplicações a que elas se destinam e a variedade de interesses dos pesquisadores envolvidos com esta área. Quando redes de elementos relativamente simples, com alta conectividade,

que utilizam algoritmo de treinamento não supervisionado são estudadas, o interesse pode ser focalizado na auto-organização. Em se tratando de aplicações técnicas, o desempenho é que importa; é secundário quão próximo da rede biológica foi modelada a rede.

De forma genérica, uma rede neural é uma máquina projetada para modelar a maneira como o cérebro desempenha uma tarefa particular ou uma função de interesse; a rede é geralmente implementada utilizando componentes eletrônicos ou simulada em *software* em um computador. Para que um bom desempenho seja obtido, as redes neurais empregam interconexão massiva de unidades denominadas neurônios ou elementos processadores. Vista como uma máquina adaptativa, uma rede neural pode ser definida da seguinte forma [15]:

Uma rede neural é um processador massivamente paralelo e distribuído que apresenta uma aptidão natural para armazenar conhecimento obtido através de experiência e torná-lo disponível para uso. Ela se assemelha ao cérebro em dois aspectos:

1. *Conhecimento é adquirido pela rede através de um processo de aprendizagem;*

2. *Conexões interneuronais conhecidas como pesos sinápticos são utilizadas para armazenar o conhecimento.*

Uma outra definição de redes neurais é fornecida a seguir:

As redes neurais são compostas de muitos elementos simples (neurônios), interconectados através de ligações (pesos), de acordo com uma arquitetura. Alguns desses elementos recebem informações externas e as distribuem para a rede, onde as informações são processadas. O comportamento da rede depende de sua arquitetura e dos valores de seus pesos. A idéia de aprendizagem em uma rede neural é implementada através da modificação dos pesos de tal forma que, à medida que a informação é transmitida à rede, o processamento desejado é obtido [35].

Tomadas isoladamente, as definições de redes neurais artificiais não são inteiramente satisfatórias: ou são bastante genéricas ou enfocam apenas alguns aspectos.

Convém salientar que as redes neurais também costumam ser denominadas neurocomputadores, redes conexionistas ou ainda processadores distribuídos paralelos.

As redes neurais são dotadas de algumas características interessantes, dentre as quais destacam-se sua estrutura massivamente paralela e distribuída e sua habilidade de aprender e generalizar. A generalização diz respeito ao fato das redes neurais apresentarem saídas razoáveis para entradas não encontradas durante sua fase de treinamento ou aprendizagem. Essas características referentes à capacidade de processamento de informação tornam as redes neurais capazes de solucionar problemas complexos, comumente não tratáveis. Conforme [15], as redes neurais apresentam as seguintes propriedades:

1. *Não-linearidade.* Um neurônio é um dispositivo não-linear. Conseqüentemente, por se constituir de uma interconexão de neurônios, uma rede neural é não-linear. Além disso, a não-linearidade ocorre de um modo bastante especial, uma vez que é distribuída ao longo da rede. A não-linearidade é uma propriedade extremamente importante, particularmente se o mecanismo responsável pela geração de um sinal de entrada, como por exemplo o sinal de voz, é inerentemente não-linear.

2. *Mapeamento entrada-saída.* Um paradigma de aprendizagem denominado *aprendizagem supervisionada* envolve a modificação dos pesos sinápticos de uma rede neural através da aplicação de um conjunto de treinamento, constituído de amostras de treinamento ou exemplos. Cada exemplo consiste de uma única entrada e a correspondente resposta desejada. Os pesos sinápticos da rede são modificados de modo a minimizar a diferença entre a resposta desejada e a resposta apresentada pela rede. Deste modo, a rede aprende, através de exemplos, a construir um mapeamento entrada-saída para o problema em questão.

3. *Adaptatividade.* As redes neurais apresentam uma capacidade intrínseca de adaptar seus pesos sinápticos de acordo com a situação em questão. Em particular, uma rede neural treinada para operar em um ambiente específico pode ser facilmente retreinada para lidar com pequenas modificações nas condições ambientais de operação. Além disso, ao operar em um ambiente não-estacionário, uma rede neural pode ser projetada para modificar seus pesos em tempo real. A arquitetura de uma rede neural, combinada com sua capacidade de adaptação, faz com

que a rede neural constitua uma ferramenta ideal para uso em processamento adaptativo de sinais, controle adaptativo e classificação adaptativa de padrões.

4. *Tolerância a falhas.* Implementada em *hardware*, uma rede neural é potencialmente tolerante a falhas no sentido de que seu desempenho é gradativamente degradado sob condições de operação adversas, tendo em vista a natureza distribuída da informações ao longo da rede.

5. *Possibilidade de implementação em VLSI.* A natureza massivamente paralela de uma rede neural a torna potencialmente rápida para realização de certas tarefas. Esta mesma característica a torna idealmente adequada para implementação utilizando tecnologia VLSI. A principal virtude de implementação VLSI é que ela provê um meio de capturar o comportamento complexo de uma rede neural em uma feição extremamente hierárquica, possibilitando o uso de uma rede neural como uma ferramenta para operação em tempo real em aplicações envolvendo reconhecimento de padrões, processamento de sinais e controle.

6. *Uniformidade.* Uma mesma notação é utilizada em todos os domínios que envolvem aplicações de redes neurais. Além disso, os neurônios, de uma forma ou de outra, representam o ingrediente comum a todas redes neurais, e nesse contexto é possível aplicar teorias e algoritmos de aprendizagem em aplicações diversas de redes neurais.

Existe uma enorme diversidade de redes neurais, destinadas a diversas aplicações. Dentre as redes neurais mais conhecidas, destacam-se: Perceptron, Hopfield, Madaline, Mapa Auto-Organizativo de Kohonen, Teoria da Ressonância Adaptativa, Memória Associativa Bidirecional, Máquinas de Cauchy e Boltzmann, Neocognitron, Avalanche, Cerebellatron e Contrapropagação. Como principais aplicações de redes neurais, podem ser citadas [36, 37]: controle de processos, controle de motores elétricos, controle de robôs, controle de veículos, diagnóstico médico, processamento de sinais médicos, processamento de imagens médicas, reconhecimento de caracteres, aplicações financeiras, linguagem natural, sensoriamento remoto, processamento de sinais de voz e

imagem, processamento de sinais sísmicos, inspeção automática, realidade virtual e comunicações digitais.

Dentre as regras gerais para representação do conhecimento em redes neurais, destacam-se três [15]:

- *Regra 1*: entradas similares devem produzir na rede neural representações internas semelhantes e devem, portanto, ser classificadas como pertencentes à mesma classe;
- *Regra 2*: entradas pertencentes a classes distintas de padrões devem resultar em representações internas bastante diferentes em uma rede neural;
- *Regra 3*: se uma determinada característica é importante, então um grande número de neurônios deve ser utilizado para a representação de tal característica na rede neural.

Como principais aspectos que caracterizam as redes neurais, destacam-se:

- tipo de não linearidade ou função de ativação
- forma de entrada dos dados;
- topologia ou arquitetura da rede;
- regra de treinamento ou aprendizagem.

Em muitas redes neurais, os neurônios podem ser caracterizados pelo tipo de não linearidade utilizada. Conforme mencionado na seção 3.4, há três tipos comuns de não linearidade: limitador brusco, limitador lógico e sigmóide. Os neurônios mais simples somam várias entradas ponderadas e transmitem o resultado através de uma não linearidade dentre as citadas.

Quanto à forma de entrada de dados, as redes podem ser classificadas como redes que aceitam dados contínuos ou redes que aceitam dados discretos.

A topologia da rede e a regra de treinamento constituem os principais aspectos que caracterizam as redes neurais e serão brevemente descritos a seguir.

3.5.1 Topologia da Rede

Em geral, podem ser identificadas quatro classes diferentes de arquiteturas ou topologias de redes neurais:

1. Redes de camada única com propagação direta;
2. Redes multicamadas com propagação direta;
3. Redes recorrentes;
4. Estruturas reticuladas.

As seções seguintes fornecem uma breve descrição dessas arquiteturas.

Redes de camada única com propagação direta

A Figura 3.7 ilustra uma rede de camada única com quatro neurônios na camada de entrada e quatro neurônios na camada de saída. É importante observar que a designação *camada única* se deve ao fato de que a rede apresenta apenas uma camada de neurônios de processamento, enquanto que o termo *propagação direta* diz respeito ao fato da informação transitar no sentido entrada-saída, ou seja, sem realimentação. Um exemplo típico dessa topologia é a memória associativa.

Redes multicamadas com propagação direta

Neste tipo de rede neural, observa-se a presença de uma ou mais *camadas escondidas*, cujos neurônios ou unidades de processamento são denominados *neurônios escondidos* ou *unidades escondidas*. Em se tratando de reconhecimento de padrões, a habilidade que a rede apresenta de extrair as características estatísticas dos padrões de entrada aumenta com o número de camadas escondidas. Em redes multicamadas, os neurônios de cada camada têm como entradas as saídas das camadas precedentes. A Figura 3.8 ilustra o *layout* de uma rede neural multicamadas com propagação direta para o caso de uma única camada escondida. A Figura 3.8 representa uma rede 9-4-2 visto que há 9 neurônios de entrada, 4 neurônios escondidos e 2 neurônios de

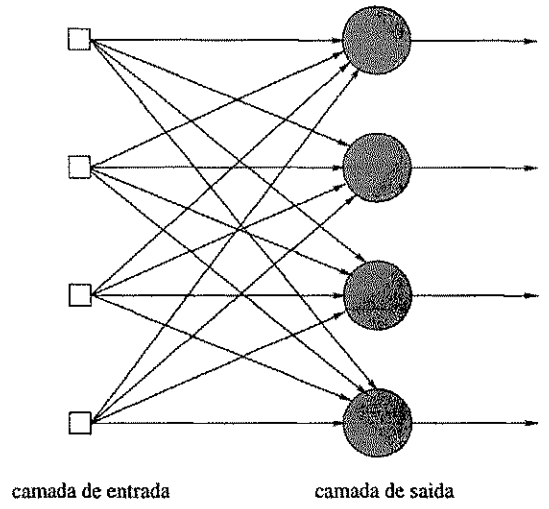


Figura 3.7: Rede com propagação direta com uma única camada de neurônios.

saída. A rede neural da Figura 3.8 é dita *completamente conectada* no sentido de que todo neurônio em cada camada da rede conecta-se a todos os neurônios da camada adjacente posterior. Por outro lado, uma rede é denominada parcialmente conectada quando algumas ligações ou pesos sinápticos estão ausentes. A Figura 3.9 ilustra esse tipo de situação.

Redes Recorrentes

Redes neurais recorrentes são caracterizadas pela presença de pelo menos um *loop* de realimentação. Esse tipo de arquitetura é ilustrado na Figura 3.10. É importante observar que os *loops* de realimentação são caracterizados pela presença de elementos de atraso, denotados por z^{-1}.

Estruturas Reticuladas

Um reticulado consiste de um arranjo de neurônios unidimensional, bidimensional ou n-dimensional, cujas entradas são fornecidas por um conjunto de neurônios de entrada. A Figura 3.11 ilustra um arranjo unidimensional de três neurônios alimentados por uma camada de entrada com três neurônios. Um arranjo bidimensional com 3x3

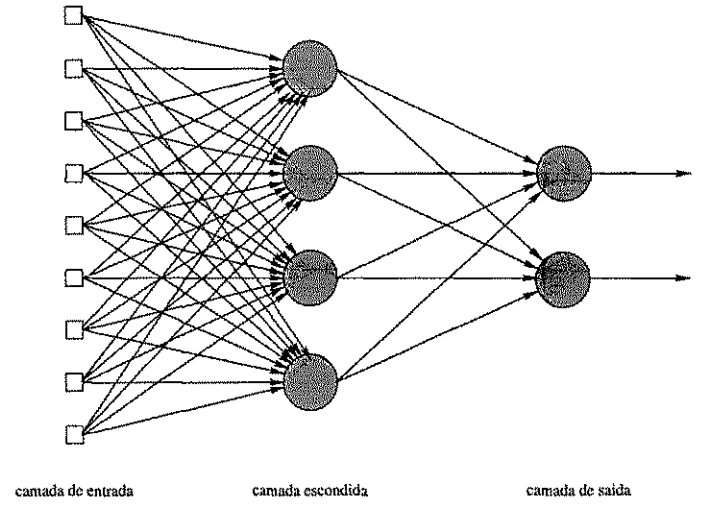


Figura 3.8: Rede neural com propagação direta, completamente conectada, com uma camada escondida e uma camada de saída.

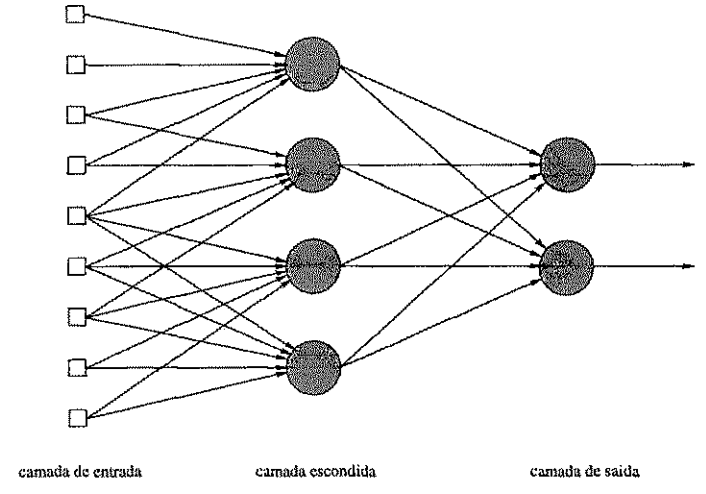


Figura 3.9: Rede neural com propagação direta parcialmente conectada.

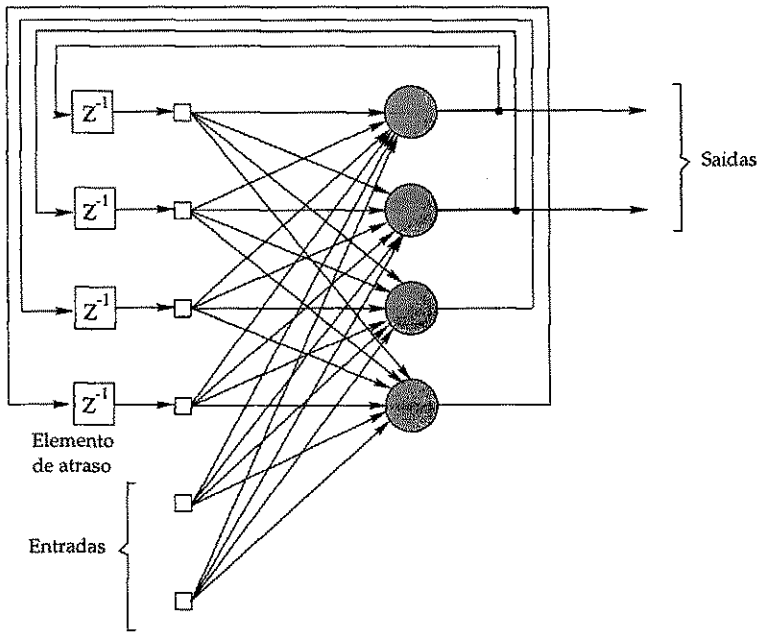


Figura 3.10: Rede recorrente.

neurônios alimentados a partir de uma camada de entrada com três neurônios é ilustrado na Figura 3.12. É importante observar que cada neurônio de entrada conecta-se a todos os neurônios do reticulado.

3.5.2 Regra de Treinamento ou Aprendizagem

Quanto à regra de aprendizagem, os algoritmos utilizados durante a fase de treinamento de redes neurais podem ser classificados em:

- algoritmos supervisionados;
- algoritmos não-supervisionados.

As redes que utilizam algoritmo de treinamento supervisionado recebem informação externa sobre a que classe pertence cada padrão de entrada utilizado na fase de treinamento. De posse dessa informação, os pesos das unidades de processamento são modificados de forma a se atingir o desempenho desejado, ou seja, uma classificação

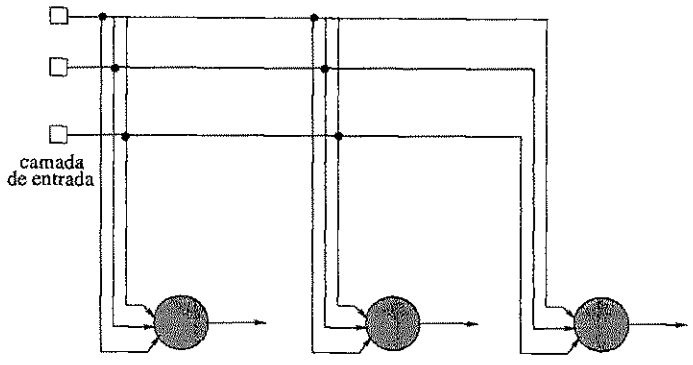


Figura 3.11: Reticulado unidimensional com três neurônios.

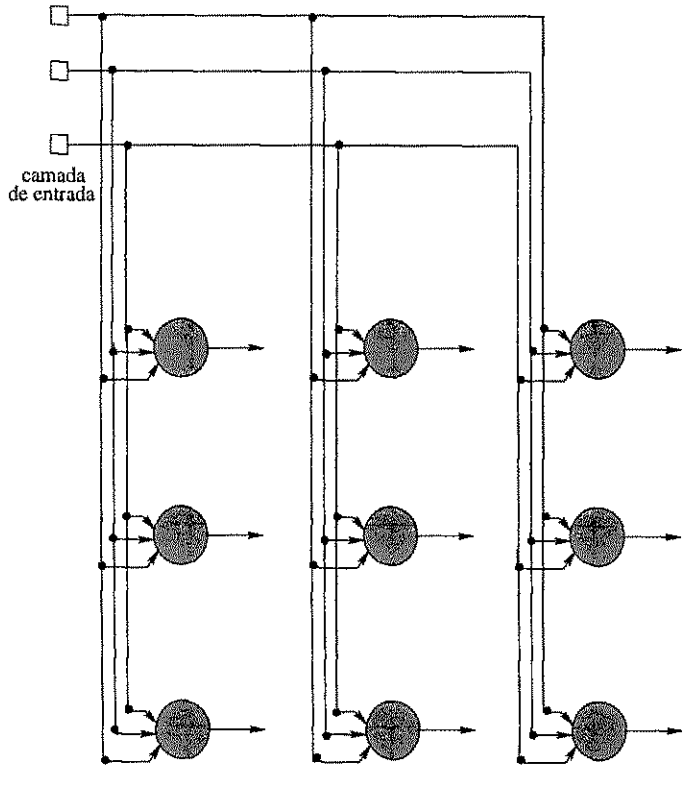


Figura 3.12: Reticulado bidimensional 3x3.

satisfatória. Um dos algoritmos de treinamento supervisionado mais utilizados é o algoritmo *backpropagation* ou algoritmo de propagação retroativa de erros, que descreve a metodologia de modificação dos pesos na rede neural denominada perceptron multicamadas [2, 15, 16].

Uma característica marcante da aprendizagem supervisionada é a presença de um supervisor externo, conforme ilustra a Figura 3.13. O supervisor é simplesmente o elemento que detém o conhecimento do problema a ser solucionado pela rede, ou seja, ele conhece o mapeamento entrada-saída. O supervisor fornece à rede um conjunto de exemplos entrada-seída. Em outras palavras, a *resposta desejada* (resposta correta), é fornecida à rede pelo supervisor. Essa resposta desejada representa a ação ótima a ser executada pela rede. Nesse contexto, os parâmetros da rede são ajustados de forma a minimizar o *sinal erro*, definido como a diferença entre a *resposta real*, apresentada pela rede, e a *resposta desejada*, fornecida pelo supervisor.

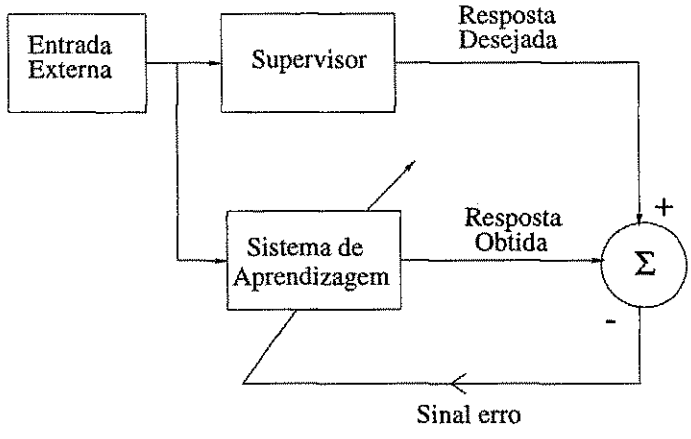


Figura 3.13: Aprendizagem supervisionada.

Na aprendizagem não-supervisionada ou auto-organizativa, não existe um supervisor externo para orientar o processo de aprendizagem. Nos algoritmos de treinamento não-supervisionados, portanto, nenhuma informação externa referente aos dados de treinamento é disponível à rede. As modificações efetuadas nos pesos são realizadas com base apenas na observação de erros internos. Nessa abordagem, o principal objetivo da

rede é capturar as regularidades estatísticas dos dados de entrada, ou seja, desenvolver a habilidade de promover representações internas eficientes das características dos padrões de entrada. Os algoritmos de aprendizagem não-supervisionada podem ser:

- Hebbianos, quando mais de uma saída é ativada para cada entrada;
- Competitivos, quando apenas uma saída é acionada para cada entrada.

As redes com treinamento supervisionado são geralmente usadas como classificadores em sistemas de reconhecimento de padrões. Por outro lado, redes com treinamento não-supervisionado são geralmente utilizadas em análise de agrupamento ou como quantizadores vetoriais.

Capítulo 4

Redes de Kohonen Aplicadas à Quantização Vetorial

4.1 Introdução

Ao contrário dos algoritmos de aprendizagem supervisionada, que durante a fase de treinamento necessitam de uma resposta externa para cada padrão de entrada ou dado de treinamento, o algoritmo de Kohonen utiliza aprendizagem não-supervisionada para modificar o estado interno da rede neural e desta forma modelar as características encontradas nos dados de treinamento. Em outras palavras, nos mapas auto-organizativos de Kohonen, a rede neural apresenta, por si mesma, sem supervisão, uma organização coerente dos dados de treinamento

Kohonen, um professor da *Faculty of Information Sciences, University of Helsinki*, dedicou-se bastante, ao longo de muitos anos, à área de redes neurais, muito antes da corrente onda de interesse desencadeada em meados da década de 80. Ele trabalhou extensivamente com conceitos de memória associativa e modelos para atividade neurobiológica. Seu trabalho pode ser encarado como um guia para modelar a auto-organização e as características de aprendizagem adaptativa do cérebro.

Há muito tempo, os neurobiologistas descobriram que áreas localizadas do cérebro, particularmente ao longo do córtex cerebral, desempenham funções específicas. Por exemplo, as funções de visão, fala e controle de movimento podem, cada uma delas, ser

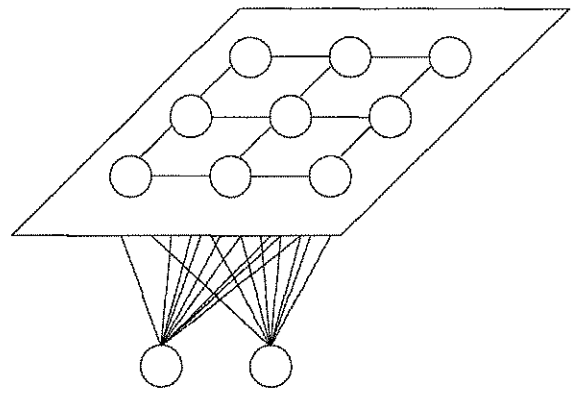


Figura 4.1: Visão geral de uma rede de Kohonen típica

4.3 O Algoritmo de Kohonen

No algoritmo de Kohonen, um mapa topográfico é organizado, de maneira autônoma, por um processo cíclico de comparação dos padrões de entrada com vetores "armazenados" em cada neurônio. Nenhuma resposta de treinamento é especificada para qualquer entrada de treinamento. Onde as entradas se equiparam aos vetores dos neurônios, aquela área do mapa é seletivamente otimizada para representar uma espécie de média dos dados de treinamento para aquela classe. De um conjunto de neurônios organizados aleatoriamente, a grade atinge uma mapa de características que apresenta representação local e é auto-organizado. O algoritmo é muito simples:

Algoritmo de Kohonen:

1. Apresente o vetor de treino $\vec{x}$;
2. Encontre o neurônio vencedor $\vec{w}_{i^*}$ (de acordo com o critério de distância mínima $d(\vec{x}, \vec{w}_{i^*}) < d(\vec{x}, \vec{w}_i), \forall i \neq i^*$);
3. Atualize $\vec{w}_{i^*}$ e a vizinhança $\mathcal{N}_{\vec{w}_{i^*}} = \{\vec{w}_i | d(\vec{w}_i, \vec{w}_{i^*}) \leq r(n)\}$, na direção de $\vec{x}$, ou seja, $\Delta w_{ij} = \eta(n)\mathcal{O}_i(x_j - w_{ij})$.

No algoritmo descrito acima, d é a medida de distorção, n é a iteração corrente ($1 \leq n \leq n_{max}$), $\eta(n)$ é a taxa de aprendizagem na n-ésima iteração, $r(n)$ é o raio

de vizinhança na n-ésima iteração, $\mathcal{O}_i$ é a função que define a vizinhança ao redor do neurônio $\vec{w}_{i^*}$ ($\mathcal{O}_i = 1$ para $\vec{w}_i \in \mathcal{N}_{\vec{w}_{i^*}}$, $\mathcal{O}_i = 0$ caso contrário), x_j é a j-ésima componente de $\vec{x}$, w_{ij} é a j-ésima componente de $\vec{w}_i$ ($1 \leq i \leq N, 1 \leq j \leq K$). Os passos 1 a 3 são repetidos até que todos os vetores da seqüência de treino sejam apresentados. Tanto a taxa de aprendizagem como a função que define a vizinhança decrescem com a iteração e os passos 1 a 3 são repetidos. O procedimento completo é repetido iterativamente n_{max} vezes (n_{max} passagens da seqüência de treino). A taxa de aprendizagem $\eta(n)$ e o raio de vizinhança $r(n)$ decrescem a cada iteração n.

Resumindo:

- Encontre a unidade mais semelhante à unidade de treinamento;
- Aumente a similaridade dessa unidade, e das unidades pertencentes à vizinhança, com a entrada.

Conforme se percebe no algoritmo, a regra de aprendizagem é simples. Não existe a necessidade de cálculo de derivadas, como ocorre nos métodos *gradient descent*, utilizado por exemplo pelo algoritmo *backpropagation*.

A rede de Kohonen é inicializada com pequenos valores aleatórios de pesos. Cada neurônio apresenta um único vetor peso, cuja dimensão é definida pelo número de componentes do vetor de entrada. Durante o ciclo de aprendizagem, um conjunto de padrões de treinamento é apresentado à rede. É feita uma comparação entre cada padrão de entrada apresentado e os vetores peso: o neurônio cujo vetor peso mais se aproxima do padrão de entrada é selecionado como vencedor. O neurônio vencedor "reivindica" o vetor de entrada (padrão de entrada) e modifica seu próprio peso para alinhá-lo com a entrada. O neurônio tornou-se, nesse momento, sensível a essa particular entrada de treinamento.

No algoritmo, é fácil perceber que os neurônios na vizinhança do neurônio vencedor também são modificados. A razão para esse fato é que a rede está tentando criar regiões que responderão a uma faixa de valores em torno da entrada de treinamento. Aos neurônios situados na vizinhança do neurônio vencedor é dado um alinhamento semelhante, e ao longo do ciclo de treinamento isso estabelece uma representação média

daquele padrão de entrada. Como conseqüência, vetores que estejam espacialmente próximos dos vetores de treinamento também serão representados corretamente, mesmo que a rede não os tenha visto antes. Isso demonstra as propriedades de generalização da rede.

É importante observar que o mapa de características auto-organizativo (SOFM, *Self-Organizing Features Map*) de Kohonen é baseado em aprendizagem competitiva: para cada vetor de entrada um único neurônio de saída é obtido, ou seja, os neurônios de saída competem entre si pelo direito de responder a uma determinada entrada.

Cumpre salientar que o desempenho do algoritmo de Kohonen depende da seleção dos parâmetros envolvidos no processo de aprendizagem. Não existe, contudo, uma base teórica para a escolha desses parâmetros. A escolha é geralmente feita através de um processo de tentativa e erro.

A seguir, serão abordados três importantes aspectos do algoritmo de Kohonen: treinamento dos pesos, inicialização dos pesos e vizinhança.

4.3.1 Treinamento dos Pesos

Observa-se, no algoritmo de Kohonen, que a modificação Δw_{ij} introduzida na j-ésima componente do vetor peso $\vec{w}_i$ é proporcional à diferença entre a j-ésima componente do vetor de entrada $\vec{x}$ e a j-ésima componente do vetor peso $\vec{w}_i$, ou seja:

$$\Delta w_{ij} = \eta(n)\mathcal{O}_i(x_j - w_{ij}) \tag{4.1}$$

A unidade de proporcionalidade $\eta(n)$ é denominada taxa de aprendizagem, que decrementa a taxa de adaptação dos pesos.

É possível distinguir dois estágios no algoritmo de Kohonen. O primeiro estágio consiste na criação de alguma forma de ordenamento topológico no mapa de neurônios aleatoriamente orientados. O processo de treinamento visa reunir os neurônios no mapa topológico para refletir uma faixa para os padrões encontrados nos dados de treinamento. Trata-se, portanto, de um mapeamento grosseiro. Como neste contexto são realizadas modificações de larga escala, a taxa de aprendizagem é mantida alta, para permitir grandes modificações nos pesos, na intenção de se atingir um mapea-

mento razoável o mais rapidamente possível. Uma vez encontrada uma representação grosseira, os neurônios no interior das regiões localizadas do mapa são "finamente sintonizados" aos vetores de entrada de treinamento. Para alcançar essa sintonia fina, modificações muito menores devem ser efetuadas nos vetores peso de cada neurônio; deste modo, a taxa de aprendizagem é reduzida à medida que o treinamento progride. Cada vez que uma nova entrada de treinamento é aplicada à rede, o neurônio vencedor deve primeiramente ser localizado: isso identifica a região do mapa que terá seus valores de pesos atualizados.

4.3.2 Inicialização dos Pesos

É comum inicializar os pesos em pequenos valores aleatórios, muito embora essa inicialização seja bastante simplificada, tendo em vista que se os vetores peso são de fato espalhados aleatoriamente a rede pode sofrer ciclos não convergentes ou muito lentos. A razão para isso pode ser explicada mais ou menos intuitivamente. Os vetores de entrada ou vetores de treinamento agrupar-se-ão, tipicamente, numa região limitada do espaço de padrões, correspondente à classe deles. Se os vetores peso armazenados nos neurônios da rede estão espalhados aleatoriamente, então é possível que surja uma situação na qual muitos vetores peso estejam orientados de forma bastante diferente da maioria das entradas de treinamento. Esses neurônios jamais vencerão em quaisquer comparações (segundo o critério de distância mínima) e permanecerão inutilizados na formação do mapa topológico. A conseqüência desse fato é que as vizinhanças no mapa de Kohonen serão muito esparsamente "povoadas" com neurônios treináveis, de tal modo que pode não haver um número suficiente de neurônios utilizáveis para representar adequadamente as classes de padrões. Como resultado, um desempenho insatisfatório pode ser obtido, refletindo a incapacidade apresentada pelo rede no tocante à incorporação de características estatísticas dos dados de entrada

Uma boa distribuição para os pesos iniciais é aquela que fornece ao algoritmo uma pista inicial com relação à provável orientação dos pesos. Mas, considerando que essa é a informação exata que se espera que a rede descubra, tal proposta não é prática, muito embora existam técnicas para se determinar uma distribuição aproximada dos

dados de entrada.

Convém mencionar que existem técnicas que consistem em adicionar ruído aleatório às entradas nos estágios iniciais de treinamento, visando distribuir os vetores num espaço de padrões mais amplo, e desse modo utilizar um maior número de neurônios no algoritmo de aprendizagem. Procura-se, dessa forma, diminuir a possibilidade de ocorrência de classes vazias, ou seja, de neurônios nunca selecionados no processo de representação dos padrões de entrada.

Também é possível fixar um valor limiar para cada neurônio, de modo que esse limiar monitore o grau de sucesso ou falha de um neurônio ao ser escolhido como vencedor. Se um neurônio está sendo selecionado regularmente, ele terá seu limiar aumentado temporariamente. Essa prática reduz a chance de tal neurônio ser selecionado como vencedor novamente e permite que neurônios redundantes sejam usados na formação das características do mapa.

4.3.3 Vizinhança

A vizinhança pode ser vista como uma superfície dinamicamente alterável que define quantos neurônios ao redor do neurônio vencedor terão seus pesos modificados durante o processo de treinamento. Inicialmente, a cada neurônio na rede será atribuída uma vizinhança grande (onde grande pode significar todos os neurônios da rede). Quando um neurônio é selecionado como o mais próximo de uma entrada, ele terá seus pesos adaptados para sintonizá-lo ao sinal de entrada. Contudo, todos os neurônios na vizinhança também serão adaptados de forma semelhante. À medida que o treinamento progride, o tamanho da vizinhança é gradativamente diminuído até um limite pré-definido.

Para verificar como isso pode forçar grupos de neurônios topologicamente relacionados, consideremos a Figura 4.2, que ilustra uma seqüência relacionada à formação topológica de grupos de características durante uma fase de treinamento. Por questões de clareza, é ilustrada a formação de apenas um grupo de padrões, o qual está centrado no neurônio realçado. A Figura 4.2 mostra o treinamento de uma vizinhança localizada. O efeito de reduzir a vizinhança é localizar áreas de atividade similar. To-

das as unidades com a área sombreada em A são inicialmente afetadas, realinhando-se ligeiramente na direção do vetor peso do neurônio vencedor. À medida que o número de iterações aumenta, a vizinhança é reduzida e apenas os neurônios situados na vizinhança do neurônio vencedor são alterados. Eles alinham-se mais e mais, até que a área ao redor do neurônio vencedor consista de vetores peso semelhantes. Na rede resultante, uma entrada semelhante àquela que acionou o neurônio original trará à tona uma resposta de um neurônio topologicamente próximo. Em A, a rede é mostrada em seu estado inicial, com vetores peso aleatórios e vizinhanças grandes ao redor de cada neurônio. As flechas dentro de cada neurônio podem ser vistas como uma representação espacial da orientação de cada vetor peso do neurônio. Em B, observa-se a rede depois de muitas passagens através do conjunto de treinamento. A região realçada do mapa está começando a formar uma orientação de classe específica estabelecida ao redor do neurônio realçado. O tamanho da vizinhança também diminui, de modo que desta maneira as modificações de peso tenham um menor campo de influência. A rede completamente treinada é mostrada em C. As vizinhanças diminuíram para um limite pré-definido, e os neurônios no interior da região foram todos adaptados para representar uma faixa média de valores em relação aos dados de treinamento para aquela classe [2].

O algoritmo de treinamento produz grupos para todos os tipos de classe encontrados nos dados de treinamento. A ordenação dos grupos no mapa e os tempos de convergência para o treinamento são dependentes do modo como os dados de treinamento são apresentados para a rede. Note que a rede forma as características internas sem supervisão.

Já foi dada ênfase para o fato de que o tamanho da vizinhança é reduzido ao longo do tempo durante a fase de treinamento. Mas quão rapidamente deve ser realizada tal redução e até que tamanho final? Infelizmente, não existem regras rígidas para algoritmos de treinamento adaptativo dessa natureza e alguma experiência é requerida em aplicações individuais. Kohonen, contudo, enfatiza que seu método não é um método frágil, isto é, pequenas mudanças nos parâmetros do sistema não refletem divergências grosseiras dos resultados de treinamento.

A taxa de aprendizagem ou adaptação, conforme visto, deve ser reduzida durante

o ciclo de treinamento, de modo que as modificações nos pesos são feitas mais gradativas à medida que o mapa se desenvolve. Isso assegura que os grupos formem uma representação interna precisa dos dados de treinamento. Em aplicações típicas, Kohonen sugere que a taxa de adaptação seja uma função que decresça linearmente com o número de passagens ao longo do conjunto de treinamento.

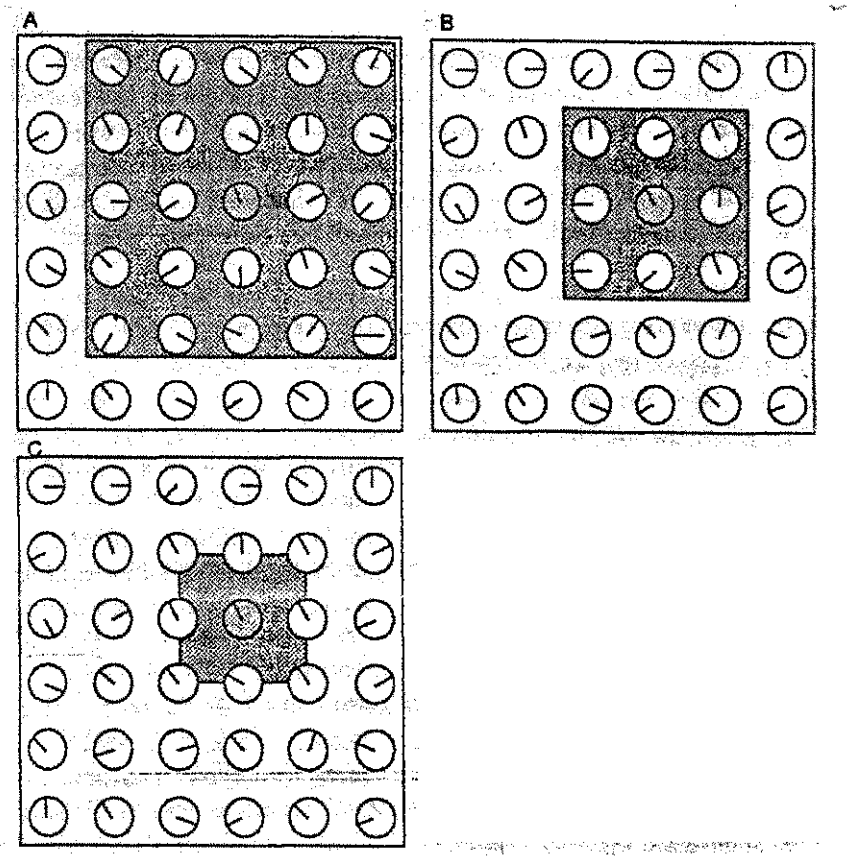


Figura 4.2: Treinamento de uma vizinhança [2].

O treinamento não é afetado apenas pela taxa de adaptação e pela taxa com a qual a vizinhança é reduzida, mas também pela forma da fronteira de vizinhança. O exemplo apresentado na Figura 4.2 discutiu apenas a possibilidade de utilização de uma vizinhança quadrada. Contudo, a vizinhança poderia ter sido definida como sendo uma região circular ou hexagonal. Assim como a taxa de aprendizagem, entretanto,

é preferível iniciar com vizinhanças muito longas inicialmente e permití-las diminuir lentamente com o número de iterações.

4.4 Algoritmos Modificados de Kohonen

Esta seção tem como objetivo principal descrever a implementação de dois algoritmos resultantes de modificações introduzidas no algoritmo de Kohonen tradicional. Serão apresentados resultados de simulações realizadas em sinais com função densidade de probabilidades conhecidas e em aplicações de quantização vetorial para compressão de voz e imagem a baixas taxas.

4.4.1 Algoritmo KMTAU

Esta seção descreve o algoritmo modificado de Kohonen, denominado KMTAU (Kohonen Modificado com Taxa de Aprendizagem Uniforme). A modificação consiste em centrar a vizinhança ao redor do vetor de treino $\vec{x}$ na tentativa de se produzir uma influência mais efetiva do sinal de treinamento nos vetores peso. No algoritmo de Kohonen tradicional, quando um neurônio vencedor $\vec{w}_{i^*}$ é determinado, a vizinhança $\mathcal{N}_{\vec{w}_{i^*}}$ é centrada em torno deste e algumas modificações insatisfatórias nos vetores peso são realizadas. Na Figura 4.3, por exemplo, apesar de $\vec{w}_1$ ser mais parecido (mais próximo, segundo um critério de distância) com $\vec{x}$ que $\vec{w}_2$, não seria atualizado pela ocorrência de $\vec{x}$ enquanto que $\vec{w}_2$ sim, uma vez que $\vec{w}_2$ pertence à vizinhança $\mathcal{N}_{\vec{w}_{i^*}}$, definida pela abordagem tradicional de Kohonen. A modificação sugerida evita essa situação, conforme mostra a Figura 4.3, visto que $\vec{w}_1$ pertence à vizinhança $\mathcal{N}_x$, definida ao redor do vetor de entrada $\vec{x}$.

Avaliação do algoritmo KMTAU

Simulações realizadas em sinais com distribuições conhecidas

A quantização vetorial costuma ser avaliada através de sinais com distribuições conhecidas [5, 7, 28, 34, 38, 39, 40], uma vez que os resultados experimentais obtidos

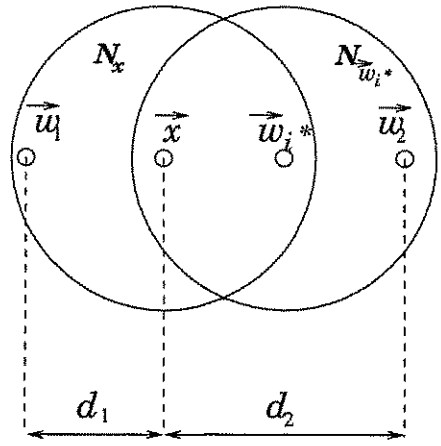


Figura 4.3: Vizinhança centrada no vetor de treino.

podem ser comparados com os limites de desempenho teóricos disponíveis para alguns desses sinais.

Em todas as simulações que envolveram sinais com funções densidade de probabilidades conhecidas, foram utilizadas seqüências de treino com 30000 amostras apresentando variância $0,05$ e média zero. O processo de Gauss-Markov utilizado para avaliação de desempenho de quantizadores vetoriais apresenta coeficiente de correlação $0,9$.

Conforme mostra a Tabela 4.1, os vetores do dicionário de quantização vetorial obtidos com o algoritmo KMTAU tendem a descrever a distribuição estatística dos vetores da fonte a ser quantizada.

As Figuras 4.4 e 4.5 apresentam o desempenho do algoritmo KMTAU operando à taxa de $1,0$ *bit/amostra* para diversas dimensões em termos da relação sinal-ruído de quantização (SNR), para o caso das distribuições de Gauss-Markov e gaussiana, respectivamente. De baixo para cima, as curvas correspondem, respectivamente, ao algoritmo LBG, ao algoritmo KMTAU e ao limite da Distorção versus Taxa. Observe que KMTAU supera o algoritmo LBG em todas as dimensões consideradas. Além disso, à medida que a dimensão k aumenta, SNR tende para o limite fornecido pela Teoria da Distorção Versus Taxa de Shannon.

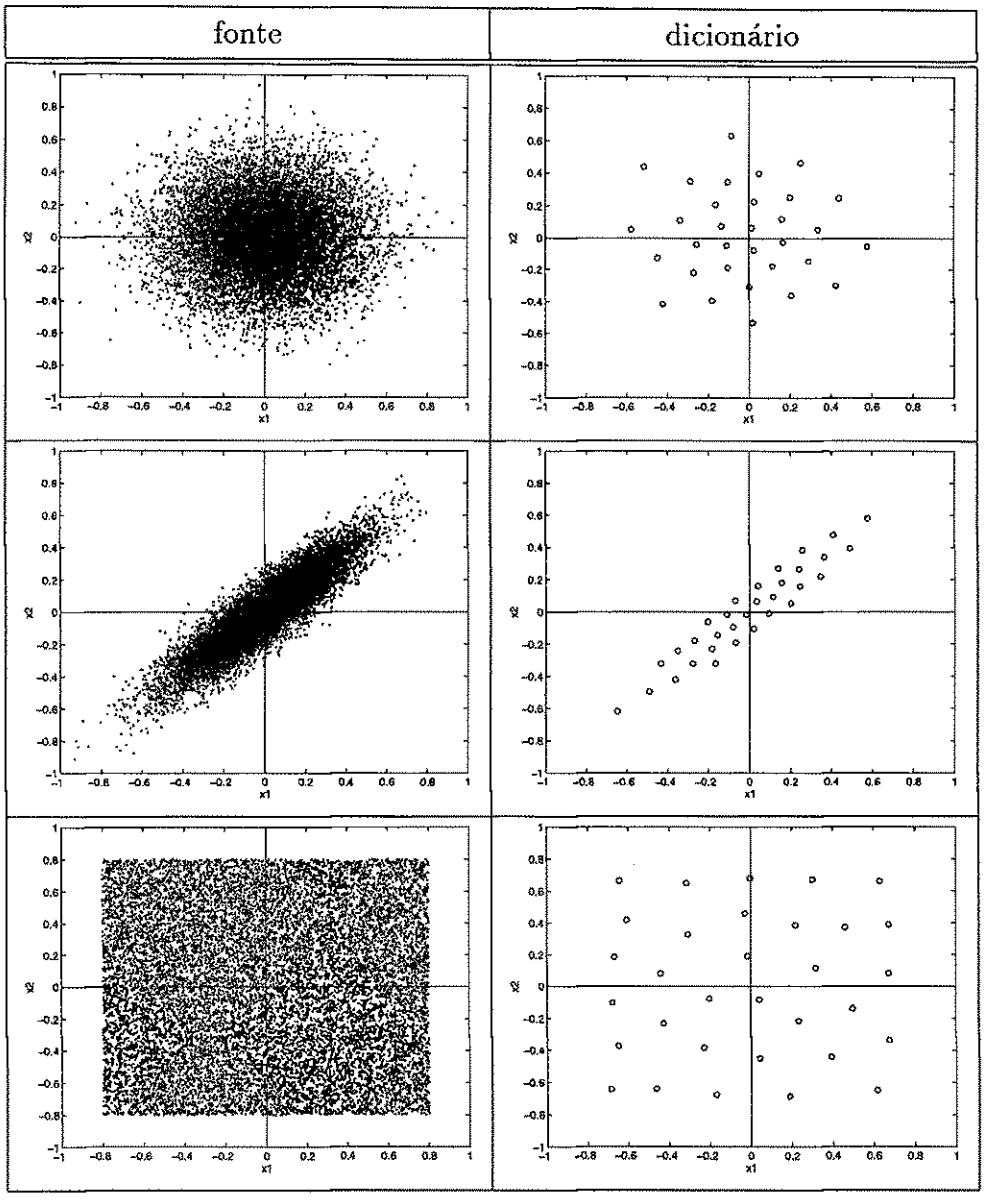


Tabela 4.1: Fonte de entrada representada por 15000 vetores bidimensionais e os respectivos dicionários obtidos, compostos por 32 vetores bidimensionais. x_1 e x_2 representam, respectivamente, a primeira e a segunda componente do vetor $\mathbf{x} \in R^2$.

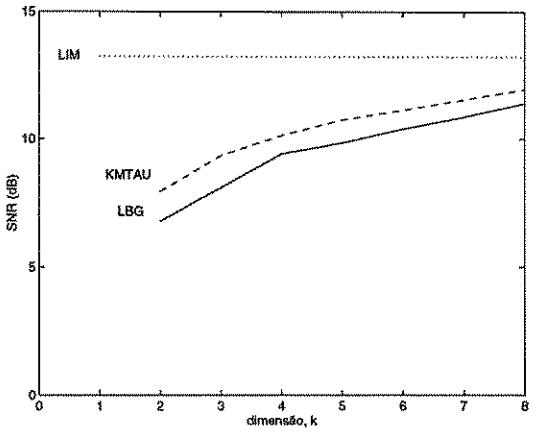


Figura 4.4: Comparação do desempenho do quantizador vetorial com o limite teórico (LIM) para a taxa de $1,0$ *bit/amostra* para a distribuição de Gauss-Markov.

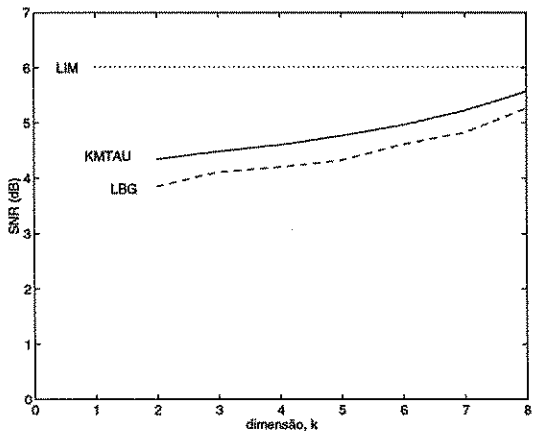


Figura 4.5: Comparação do desempenho do quantizador vetorial com o limite teórico (LIM) para a taxa de $1,0$ *bit/amostra* para a distribuição gaussiana.

Um outro conjunto de experimentos consistiu em fixar uma dimensão e avaliar o desempenho do quantizador vetorial para diferentes taxas. As Figuras 4.6 e 4.7 apresentam o desempenho do algoritmo KMTAU para dimensões $k = 4$ e $k = 8$, respectivamente: de baixo para cima, as curvas correspondem, respectivamente, ao algoritmo LBG para distribuição gaussiana, ao algoritmo KMTAU para distribuição gaussiana, ao algoritmo LBG para distribuição de Gauss-Markov e ao algoritmo KMTAU para a distribuição de Gauss-Markov. Na Figura 4.8, é apresentado o resultado das simulações envolvendo um sinal com distribuição uniforme para dimensão $k = 2$: de baixo para cima, as curvas dizem respeito, respectivamente, ao algoritmo LBG e ao algoritmo KMTAU. Conforme se pode observar, para esse conjunto de experimentos, o algoritmo KMTAU apresenta desempenho superior ao apresentado pelo algoritmo LBG.

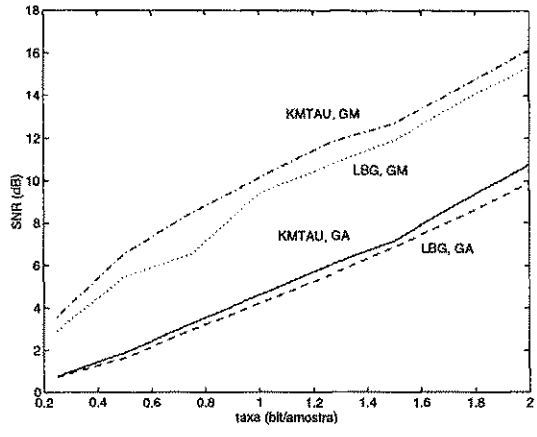


Figura 4.6: Comparação do desempenho dos algoritmos KMTAU e LBG a diferentes taxas de um quantizador vetorial de dimensão $k = 4$. GA e GM representam, respectivamente, as distribuições gaussiana e de Gauss-Markov.

A forma de onda correspondente às 800 primeiras amostras do sinal original com distribuição de Gauss-Markov é apresentada na Figura 4.9. O sinal quantizado a $1,0$ $bit/amostra$ e o correspondente erro de quantização são apresentados, respectivamente, nas Figuras 4.10 e 4.11. Observe a semelhança dos sinais original e reconstruído. A Figura 4.12 ilustra o erro de quantização a $3,0$ $bit/amostra$. Conforme esperado, este erro é inferior ao obtido com a taxa de $1,0$ $bit/amostra$.

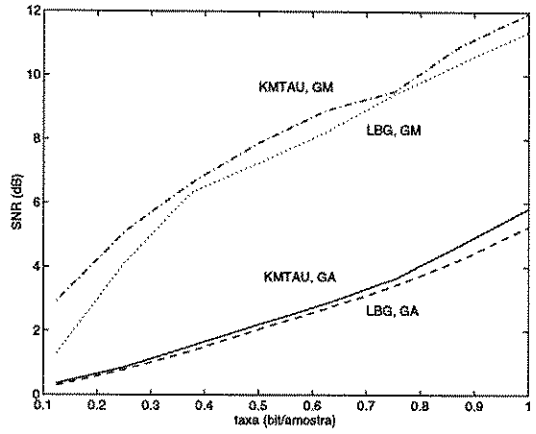


Figura 4.7: Comparação do desempenho dos algoritmos KMTAU e LBG a diferentes taxas de um quantizador vetorial de dimensão $k = 8$. GA e GM representam, respectivamente, as distribuições gaussiana e de Gauss-Markov.

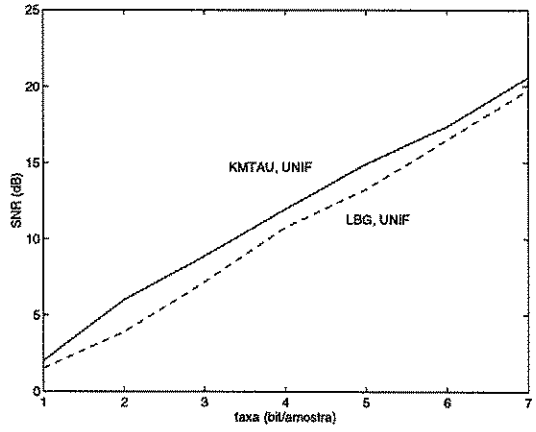


Figura 4.8: Comparação do desempenho dos algoritmos KMTAU e LBG a diferentes taxas de um quantizador vetorial de dimensão $k = 2$. UNIF representa a distribuição uniforme.

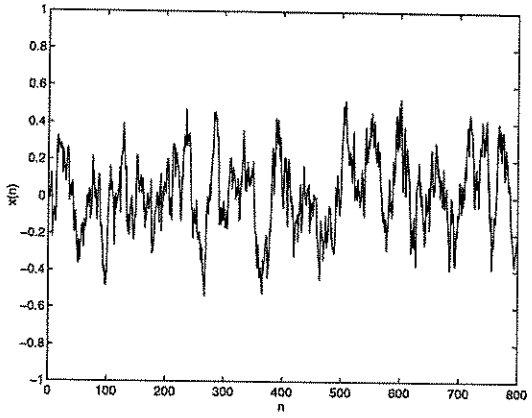


Figura 4.9: Forma de onda (800 amostras) do sinal original com distribuição de Gauss-Markov.

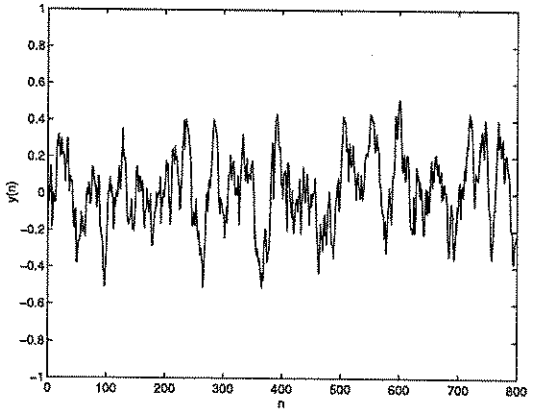


Figura 4.10: Forma de onda (800 amostras) do sinal com distribuição de Gauss-Markov reconstruído a 1,0 *bit/amostra*.

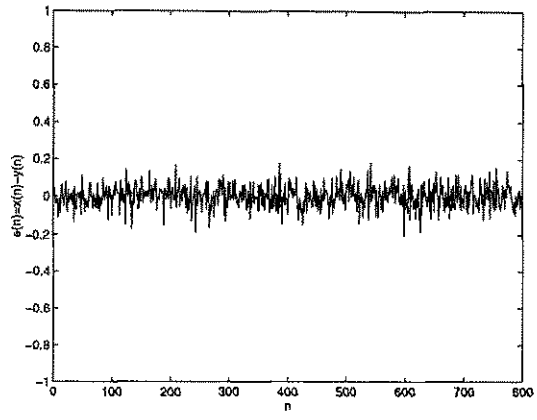


Figura 4.11: Forma de onda (800 amostras) do erro de quantização a 1,0 *bit/amostra* (distribuição de Gauss-Markov).

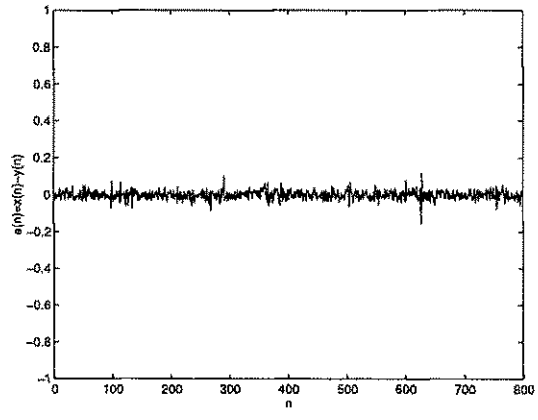


Figura 4.12: Forma de onda (800 amostras) do erro de quantização a 3,0 *bit/amostra* (distribuição de Gauss-Markov).

k	N	H_n
4	8	0.998
4	32	0.991
4	64	0.971
4	128	0.979

Tabela 4.2: Entropia normalizada para distribuição gaussiana.

k	N	H_n
8	4	0.985
8	8	0.990
8	32	0.979
8	256	0.972

Tabela 4.3: Entropia normalizada para distribuição de Gauss-Markov.

As Tabelas 4.2, 4.3 e 4.4 apresentam resultados de entropia normalizada para as distribuições gaussiana, de Gauss-Markov e uniforme, respectivamente. Observe que o algoritmo KMTAU apresenta um excelente desempenho com relação à entropia normalizada dos vetores-código, uma vez que em todas as simulações, os resultados obtidos estiveram sempre próximos do valor máximo ($H_n = 1$).

k	N	H_n
2	16	0.999
2	32	0.998
2	64	0.981
2	128	0.984

Tabela 4.4: Entropia normalizada para distribuição uniforme.

Simulações realizadas com forma de onda de voz

A quantização vetorial tem sido objeto de estudo para aplicação em codificação de forma de onda de voz [5, 7, 35, 39, 41, 42, 43, 44, 45, 46, 47, 48].

Nesta seção, serão reportados os resultados de experimentos realizados com quantização vetorial aplicada à codificação de forma de onda de voz. Convém salientar, entretanto, que a quantização vetorial pode ser utilizada em codificação paramétrica.

O primeiro conjunto de experimentos consistiu em projetar um quantizador vetorial de forma de onda de voz a 1,0 *bit/amostra*. Para tanto, utilizou-se uma seqüência de treino bastante curta, constituída de 7120 amostras (0,89s), correspondente à palavra "*aplausos*". Optou-se pela escolha de uma seqüência de treino curta depois de se observar que os resultados foram compatíveis com os resultados obtidos ao se utilizar longas seqüências de treino. A utilização de seqüências de treino curtas implica projeto de dicionários com maior eficiência com relacão ao tempo de processamento. A escolha da palavra "*aplausos*" foi motivada pelo fato de que ela apresenta uma boa variedade de sons: explosivos (fonema $|p|$), sonoros (fonema $|a|$) e fricativos surdos (fonema $|s|$). Além disso, esta palavra é bastante representativa no que diz respeito às características da fala, como por exemplo predominância de baixas amplitudes (Figuras 4.13 e 4.14), função de autocorrelação típica (Figuras 4.15 e 4.16) e distribuição típica de vetores no plano bidimensional (Figuras 4.17 e 4.18). Para avaliação do quantizador, utilizou-se uma seqüência de teste relativamente longa, constituído de 29120 amostras (3,64s), correspondente ao conjunto de setenças "*O sol ilumina a fachada de tarde. Trabalhou mais do que podia*". Conforme mostra a Figura 4.19, KMTAU apresenta desempenho superior ao apresentado pelo algoritmo LBG à taxa de $1,0$ *bit/amostra*, para todas as dimensões consideradas.

O segundo conjunto de experimentos consistiu em avaliar a faixa dinâmica do quantizador vetorial. Conforme se observa nas Figuras 4.20 e 4.21, a quantização é relativamente pobre nas regiões de baixa amplitude do sinal, correspondentes às regiões negativas da Figura 4.20. Esta limitação pode ser trabalhada através do uso de esquemas adaptativos de QV [49]. Conforme se observa na Figura 4.22, a faixa dinâmica do quantizador vetorial melhora com o aumento da taxa em *bit/amostra*.

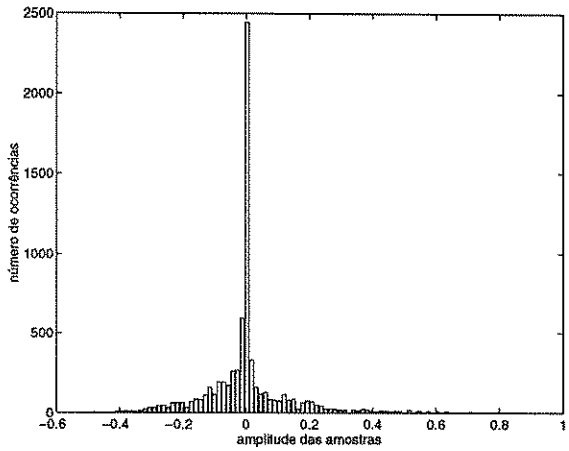


Figura 4.13: Histograma do sinal "*aplausos*".

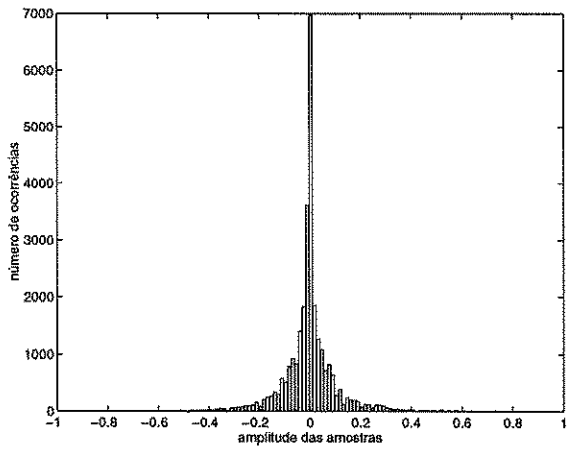


Figura 4.14: Histograma do sinal "*O sol ilumina a fachada de tarde. Trabalhou mais do que podia.*"

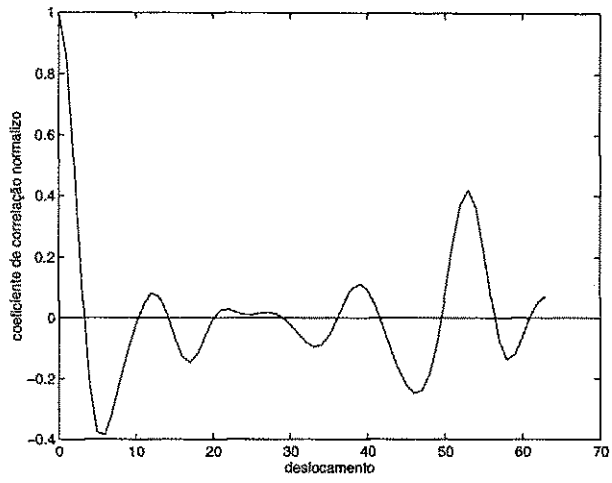


Figura 4.15: Função de autocorrelação normalizada do sinal "*aplausos*".

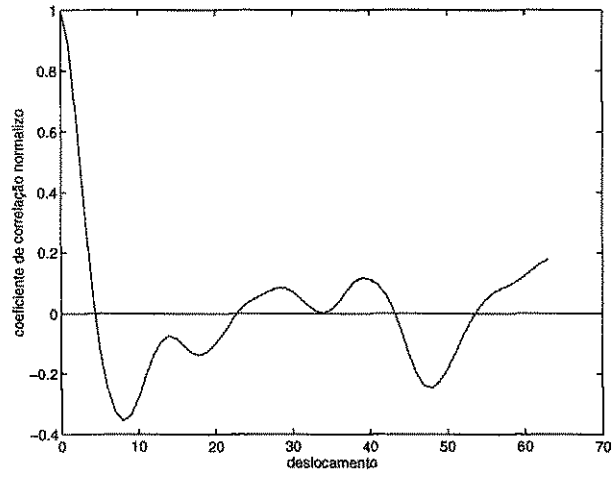


Figura 4.16: Função de autocorrelação normalizada do sinal "*O sol ilumina a fachada de tarde. Trabalhou mais do que podia.*"

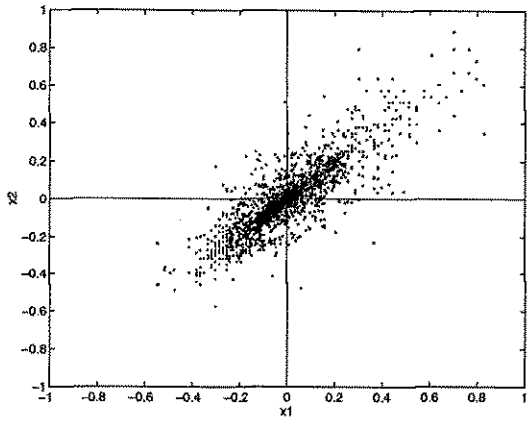


Figura 4.17: Palavra "*aplausos*" (0,89s, 3560 vetores). x_1 e x_2 representam, respectivamente, a primeira e a segunda componente do vetor $\mathbf{x} \in R^2$.

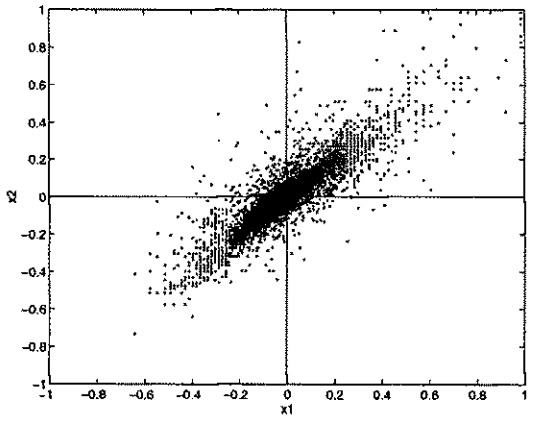


Figura 4.18: Seqüência de voz "*O sol ilumina a fachada de tarde. Trabalhou mais do que podia*" (3,64s, 14560 vetores). x_1 e x_2 representam, respectivamente, a primeira e a segunda componente do vetor $\mathbf{x} \in R^2$.

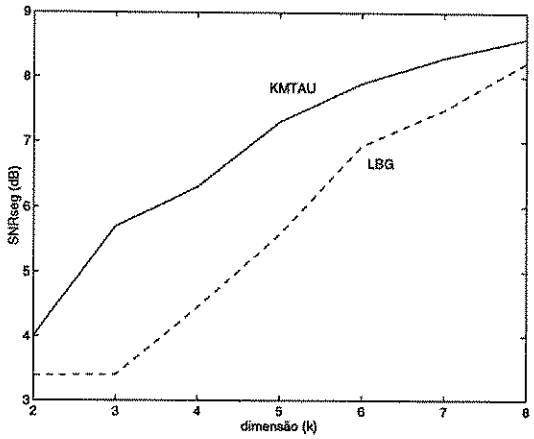


Figura 4.19: Comparação do desempenho dos algoritmos LBG e KMTAU em relação aos valores de SNR_{seg} para diversas dimensões k à taxa de 1,0 *bit/amostra.*

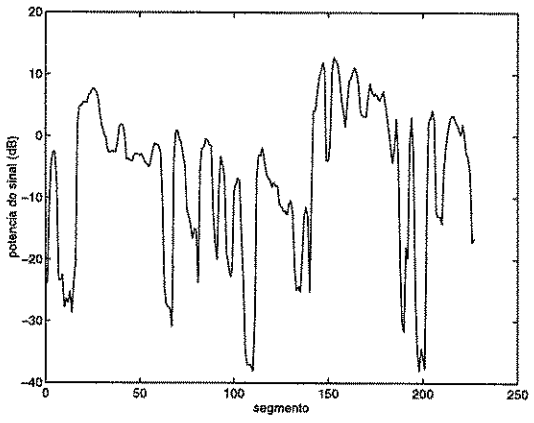


Figura 4.20: Faixa dinâmica do sinal de voz "*O sol ilumina a fachada de tarde. Trabalhou mais do que podia.*"

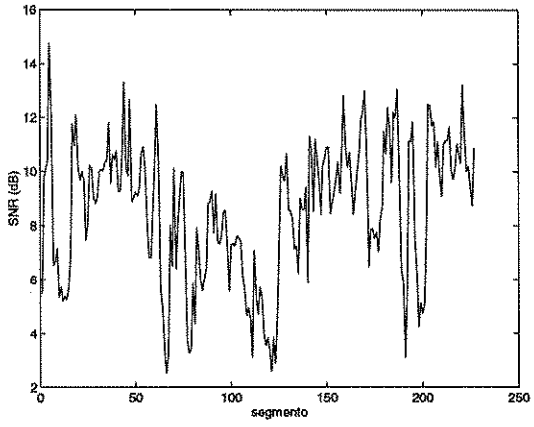


Figura 4.21: SNR_{seg} observada a cada segmento de 128 amostras após a quantização a 1,0 *bit/amostra* (dimensão $k = 8$) do sinal de voz "*O sol ilumina a fachada de tarde. Trabalhou mais do que podia.*"

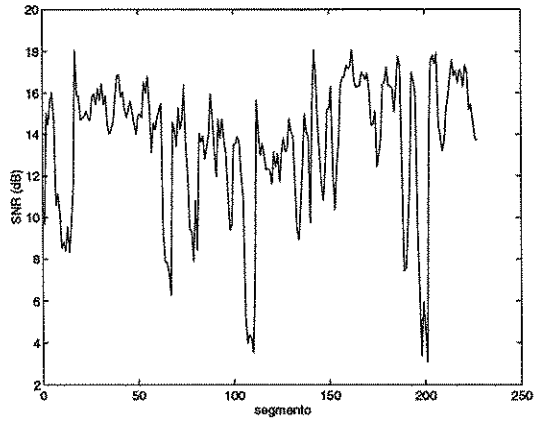


Figura 4.22: SNR_{seg} observada a cada segmento de 128 amostras após a quantização a 3,0 *bit/amostra* (dimensão $k = 2$) do sinal de voz "*O sol ilumina a fachada de tarde. Trabalhou mais do que podia.*"

K	N	DI	DII	DIII
2	32	6.56	4.47	7.06
2	64	7.50	4.60	9.58
2	128	7.99	4.68	13.78
4	32	5.04	3.96	6.26
4	64	5.58	4.17	8.76
4	128	5.84	10.02	10.02
8	32	3.83	3.25	4.63
8	64	4.12	5.36	5.70
8	128	4.58	6.61	6.74

Tabela 4.5: Valores de SNR_{seg} obtidos com o algoritmo LBG para dicionários iniciais diferentes (DI, DII e DIII).

Como parte dos experimentos, ainda, analisou-se a influência do dicionário inicial no desempenho do algoritmo. A Tabela 4.5 mostra que o algoritmo LBG é bastante sensível ao dicionário inicial. Por outro lado, a Tabela 4.6 mostra claramente que o algoritmo KMTAU é consideravelmente menos sensível ao dicionário inicial. Desta forma, portanto, KMTAU dispensa a incômoda preocupação em densenvolver algoritmos para uma boa escolha de dicionário inicial.

A Tabela 4.7 mostra que o algoritmo KMTAU apresenta um bom desempenho em relação à entropia normalizada, tendo em vista que o valor máximo alcançado por esta medida é 1.

As Figuras 4.23, 4.24 e 4.25 mostram claramente a superioridade da quantização vetorial sobre a escalar. Obseve que à taxa de 1,0 *bit/amostra*, o sinal reconstruído através de quantização escalar não apresenta nenhuma semelhança com o sinal original. Por outro, lado, à mesma taxa, o sinal reconstruído através de quantização vetorial apresenta forma de onda semelhante à forma de onda original.

K	N	DI	DII	DIII
2	32	11.1	10.73	10.94
2	64	13.7	13.83	13.59
2	128	15.9	16.00	16.05
4	32	7.8	8.05	5.37
4	64	9.8	9.78	9.72
4	128	11.1	11.20	11.30
8	32	5.5	5.55	4.09
8	64	6.8	6.58	6.19
8	128	7.8	7.32	7.67

Tabela 4.6: Valores de SNR_{seg} obtidos com o algoritmo KMTAU para dicionários iniciais diferentes (DI, DII e DIII).

k	N	H_n
2	4	0.919
3	8	0.871
4	16	0.885
5	32	0.883
6	64	0.872
7	128	0.882
8	256	0.892

Tabela 4.7: Entropia normalizada obtida para sinal de voz "*O sol ilumina a fachada de tarde. Trabalhou mais do que podia.*"

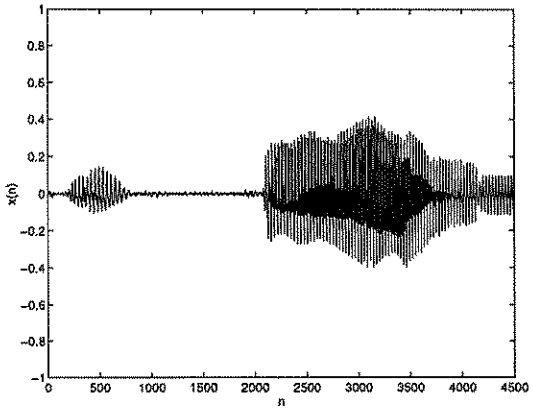


Figura 4.23: Trecho do sinal original.

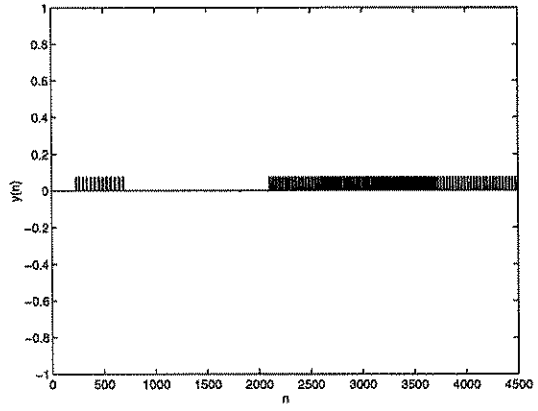


Figura 4.24: Sinal resultante da quantização escalar a 1,0 *bit/amostra*.

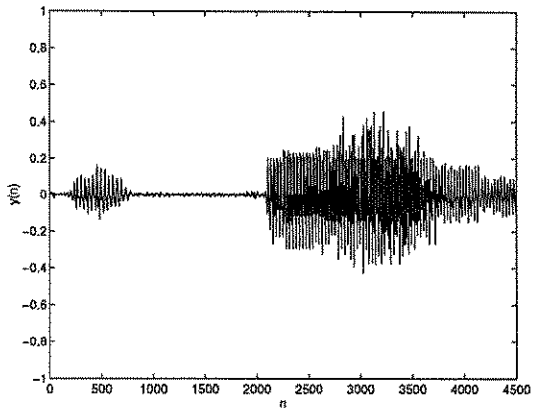


Figura 4.25: Sinal resultante da quantização vetorial a 1,0 *bit/amostra.*

Simulações realizadas com imagem

A compressão de imagens desempenha um papel importante em diversas aplicações, tais como: videoconferências, sensoriamento remoto, multimídia, dentre outras.

Tal qual para voz, o objetivo de um sistema de compressão de imagens é reduzir a quantidade de dados necessária para se representar uma imagem digital

Dentre as técnicas comumente utilizadas para compressão de imagens, podem ser citadas a codificação por transformada e a quantização vetorial. Na primeira, uma transformação é utilizada para mapear a imagem num conjunto de coeficientes transformados, os quais são quantizados e codificados. A segunda consiste num mapeamento de um espaço vetorial k-dimensional em um conjunto finito de protótipos, cujo objetivo é reduzir a taxa de bits de forma a permitir transmissão e armazenamento eficientes.

Em se tratando de codificação por transformada, uma grande variedade de abordagens tem sido utilizada, como por exemplo: transformada discreta de cosseno [1], análise de componentes principais [50, 51, 52] e transformada wavelet [53, 54, 55, 56].

A quantização vetorial pode ser aplicada à imagem nos domínios espacial, preditivo ou transformado, ou ainda em uma combinação destes, os chamados domínios híbridos. A quantização vetorial, portanto, tem sido amplamente utilizada em diversos sistemas de codificação de imagens [5, 7, 20, 21, 23, 27, 40, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70]. Vale salientar que a estrutura básica de todos os sistemas de quantização

vetorial é bastante adequada para implementação VLSI.

Em quantização vetorial de imagens, os vetores são geralmente formados pela divisão da imagem em blocos quadrados de pixels: a dimensão k é o número de pixels do bloco e taxa em bit por pixel (*bpp*) é $R = (\log_2 N)/k$, onde N é o número de vetores do dicionário.

Em todas as simulações realizadas, foram utilizados blocos de tamanho 4x4. Utilizou-se, portanto, quantização vetorial com dimensão $k = 16$. Foram utilizadas imagens 256x256, originalmente quantizadas à taxa de 8 *bpp*.

A Tabela 4.8 apresenta imagens originais (8 *bpp*), quantizadas (reconstruídas ou comprimidas) e erro (diferença entre a imagem original e a quantizada). Na primeira linha da Tabela, a imagem Lena foi quantizada a 0,5 *bpp*, na segunda linha a imagem Frog foi reconstruída a 0,437 *bpp* e na terceira linha a imagem Mandrill foi quantizada a 0,312 *bpp*. Apesar da quantização a baixas taxas, as imagens reconstruídas apresentam boa qualidade.

A Figura 4.26 apresenta o desempenho dos algoritmos LBG e KMTAU para a imagem Lena. Para a imagem Gull, ilustrada na Figura 4.27, o desempenho dos algoritmos é apresentado na Figura 4.28. Para todas as taxas consideradas, o algoritmo KMTAU apresenta desempenho superior ao apresentado pelo algoritmo LBG.

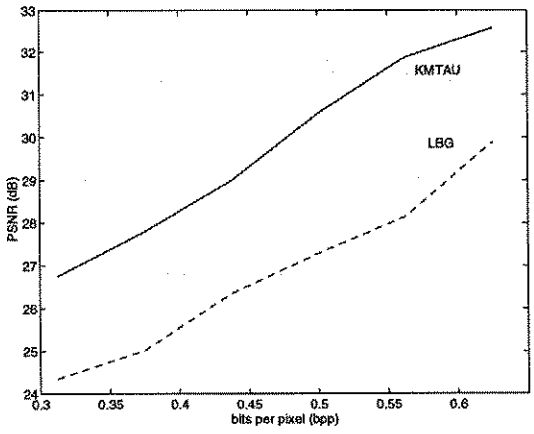


Figura 4.26: Comparação de desempenho dos algoritmos LBG e KMTAU em relação aos valores de $PSNR$ para a imagem Lena a diferentes taxas.

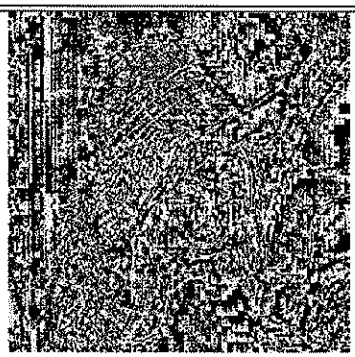
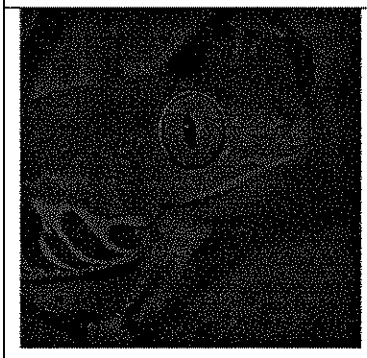
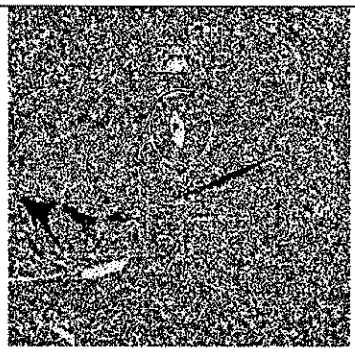
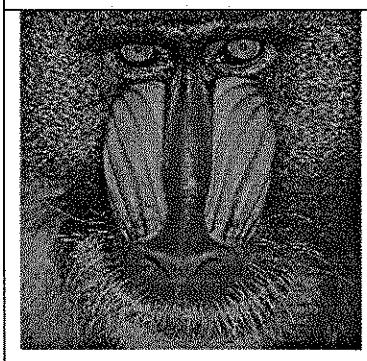
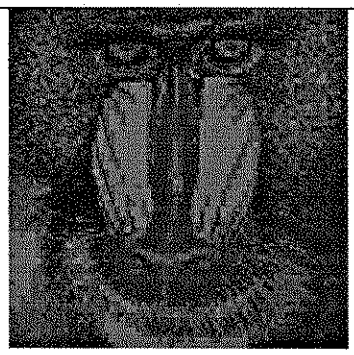
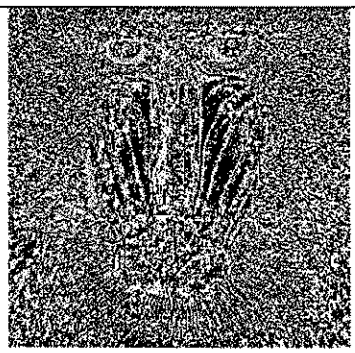
original	quantizada	erro
		
		
		

Tabela 4.8: Avaliação das imagens reconstruídas.

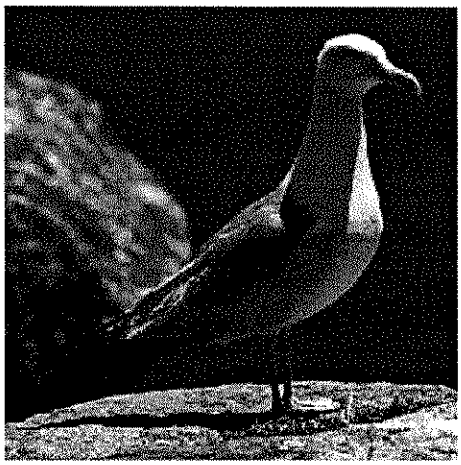


Figura 4.27: Imagem Gull original.

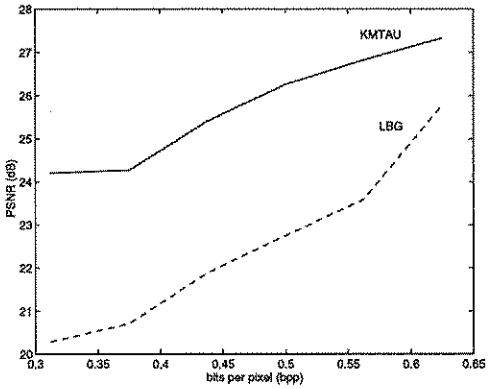


Figura 4.28: Comparação de desempenho dos algoritmos LBG e KMTAU em relação aos valores de $PSNR$ para imagem Gull a diferentes taxas.

Como parte dos experimentos, ainda, analisou-se a influência do dicionário inicial no desempenho do algoritmo. A Tabela 4.9 mostra que o algoritmo LBG é bastante sensível ao dicionário inicial. A Tabela 4.10 mostra claramente que o algoritmo KMTAU é praticamente insensível ao dicionário inicial.

Taxa	DI	DII	DIII
0.375	22.78	25.02	19.64
0.437	22.17	26.35	19.81
0.500	25.11	27.29	19.85

Tabela 4.9: $PSNR$ para três dicionários iniciais diferentes (DI, DII e DIII) para o algoritmo LBG para a imagem Lena.

Taxa	DI	DII	DIII
0.375	27.80	27.80	27.80
0.437	29.07	29.02	28.99
0.500	30.53	30.53	30.56

Tabela 4.10: $PSNR$ para três dicionários iniciais diferentes (DI, DII e DIII) para o algoritmo KMTAU para a imagem Lena.

4.4.2 Algoritmo KMTAN

Esta seção descreve o algoritmo KMTAN (Kohonen Modificado com Taxa de Aprendizagem Não-Uniforme), que corresponde à modificação do algoritmo de Kohonen em dois aspectos. A primeira modificação consiste em centrar a vizinhança ao redor do vetor de treino $\vec{x}$, ao invés de centrá-la em torno do neurônio vencedor $\vec{w}_{i^*}$, obedecendo a abordagem utilizada no algoritmo KMTAU. A segunda modificação diz respeito à introdução de um fator de modificação da taxa de aprendizagem $\eta(n)$. No algoritmo tradicional de Kohonen, $\eta(n)$ diminui a cada iteração mas permanece constante (ou seja, uniforme) dentro de cada iteração. A modificação realizada é a seguinte: ao invés de permanecer constante dentro de cada iteração, a taxa de aprendizagem varia inversamente com $d(\vec{x}, \vec{w}_i)$ dentro de cada iteração:

$$\Delta w_{ij} = \eta(n) f(d(\vec{x}, \vec{w}_i)) \mathcal{O}_x(x_j - w_{ij}) \tag{4.2}$$

Faixa de distâncias	$f(d)$
$d \geq 10^{-1}$	1
$10^{-2} \leq d < 10^{-1}$	2
$10^{-3} \leq d < 10^{-2}$	3
$10^{-4} \leq d < 10^{-3}$	4
$d < 10^{-4}$	5

Tabela 4.11: Valores associados com distâncias.

A função $f(d(\vec{x}, \vec{w}_i))$ reflete a ordem de grandeza da distância $d(\vec{x}, \vec{w}_i)$ de acordo com a Tabela 4.11; $\mathcal{O}_x$ é a função de vizinhança ao redor do vetor de treino ou vetor de entrada $\vec{x}$: ($\mathcal{O}_x = 1$ para $\vec{w}_i \in \mathcal{N}_x$, $\mathcal{O}_x = 0$, caso contrário). A taxa de aprendizagem $\eta(n)$ deve ser escolhida de modo que a estabilidade do algoritmo seja assegurada ($\eta(n) f(d(\vec{x}, \vec{w}_i)) \leq 1$ sempre). Espera-se, com essa abordagem, melhorar o mapeamento do sinal de treinamento.

Neste algoritmo modificado, denominado KMTAN, tanto a taxa de aprendizagem $\eta(n)$ como o raio de vizinhança $r(n)$ decrescem linearmente com a iteração n. O algoritmo KMTAN oferece 7 parâmetros ajustáveis: dimensão k, número de níveis N, taxa de aprendizagem inicial $\eta(1)$, taxa de aprendizagem final $\eta(n_{max})$, raio inicial $r(1)$, número de iterações n_{max} e fração de n_{max} a partir da qual $r(n) = 0$ (esse último parâmetro, portanto, define a iteração a partir da qual apenas o neurônio vencedor é atualizado).

Avaliação do algoritmo KMTAN

Simulações realizadas com forma de onda de voz

O primeiro conjunto de experimentos consistiu em projetar um quantizador vetorial de forma de onda de voz a 1,0 *bit/amostra*. Para tanto, utilizou-se uma seqüência de treino bastante curta, constituída de 7120 amostras (0,89s), correspondente à palavra "*aplausos*". Para avaliação do quantizador, utilizou-se uma seqüência de teste rela-

tivamente longa, constituído de 29120 amostras (3,64s), correspondente ao conjunto de setenças "*O sol ilumina a fachada de tarde. Trabalhou mais do que podia.*" Conforme mostra Figura 4.29, KMTAN apresenta desempenho superior ao apresentado pelo algoritmo LBG à taxa de $1,0$ $bit/amostra$ para todas as dimensões consideradas.

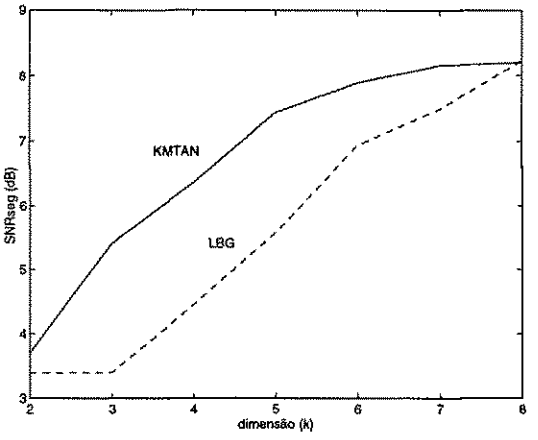


Figura 4.29: Comparação do desempenho dos algoritmos LBG e KMTAN em relação aos valores de SNR_{seg} para diversas dimensões k à taxa de 1,0 $bit/amostra$.

O segundo conjunto de experimentos consistiu em fixar uma dimensão e avaliar o desempenho do algoritmo para diferentes taxas. As Figuras 4.30 e 4.31 apresentam os resultados das simulações para dimensões $k = 4$ e $k = 8$, respectivamente. Observe que em todas as taxas avaliadas, o algoritmo KMTAN supera o algoritmo LBG.

Conforme se observa nas Figuras 4.32, 4.33 e 4.34, o algoritmo KMTAN produz dicionários que incorporam as características estatísticas do sinal de voz, como por exemplo predominância de vetores de amostras de baixas amplitudes (um maior número de vetores é alocado para a região de maior incidência de vetores de amostras do sinal de voz) e correlação típica entre amostras consecutivas (evidenciada pelo posicionamento dos vetores-código ao longo das direções das componentes principais) [46, 50].

As Tabelas 4.12, 4.13 e 4.14, mostram a evolução dos vetores-código durante o treinamento de um quantizador para $k = 3$ e $N = 8$. Foram utilizados raio inicial $r(1) = 0,25$, taxa de aprendizagem inicial $\eta(1) = 0,05$ e três iterações. O dicionário

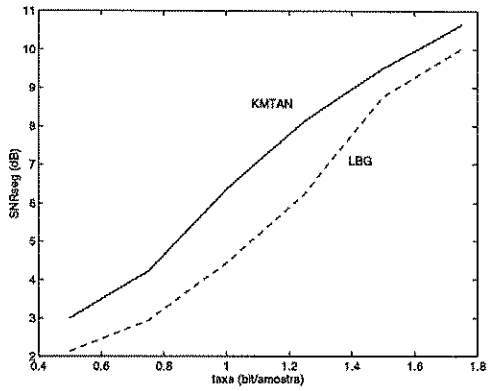


Figura 4.30: Comparação do desempenho dos algoritmos LBG e KMTAN em relação aos valores de SNR_{seg} para diversas taxas para a dimensão $k = 4$.

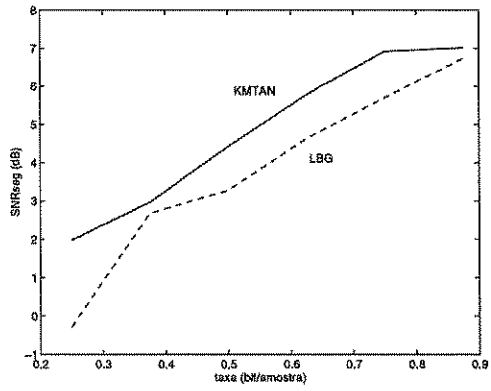


Figura 4.31: Comparação do desempenho dos algoritmos LBG e KMTAN em relação aos valores de SNR_{seg} para diversas taxas para a dimensão $k = 8$.

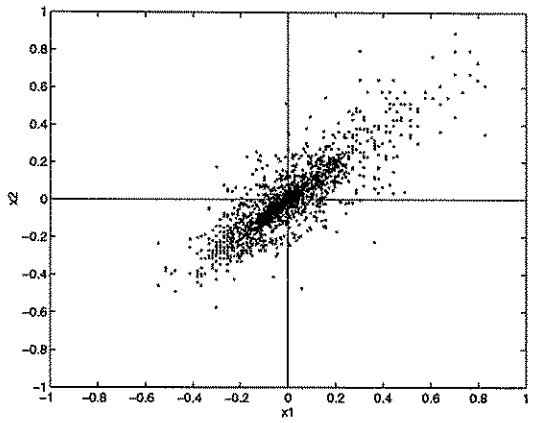


Figura 4.32: Seqüência de treino: palavra "*aplausos*" (0,89s, 3560 vetores). x_1 e x_2 representam, respectivamente, a primeira e a segunda componente do vetor $\mathbf{x} \in R^2$.

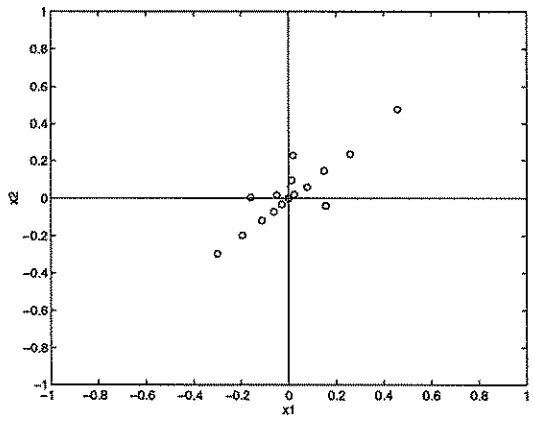


Figura 4.33: Dicionário treinado (k=2, N=16). x_1 e x_2 representam, respectivamente, a primeira e a segunda componente do vetor $\mathbf{x} \in R^2$.

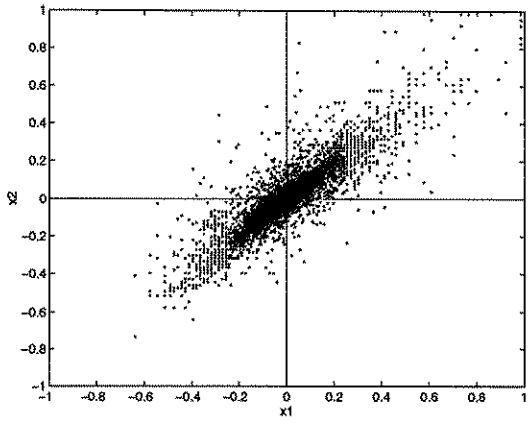


Figura 4.34: Seqüência de teste: "*O sol ilumina a fachada de tarde. Trabalhou mais do que podia*" (3,64s, 14560 vetores). x_1 e x_2 representam, respectivamente, a primeira e a segunda componente do vetor $\mathbf{x} \in R^2$.

inicial consiste de vetores aleatórios com componentes descorrelacionadas e de baixas amplitudes. Esses vetores, ao longo do processo de treinamento, são gradualmente transformados em vetores com componentes bastante correlacionadas, incorporando, assim, uma importante característica dos sinais de voz.

Para um determinado número de iterações n_{max}, foi investigado o comportamento de SNR_{seg} com a iteração n. A Figura 4.35 mostra claramente que uma escolha adequada dos parâmetros do algoritmo KMTAN pode fazer com que o dicionário atinja um melhor máximo local. A monitoração de SNR_{seg} em cada iteração n permite que o projetista escolha um dicionário que não seja necessariamente aquele obtido na última iteração. Com essa abordagem, os melhores vetores-código podem ser obtidos em uma iteração intermediária.

Durante os testes, observou-se que os melhores valores de taxa de aprendizagem inicial $\eta(1)$ e número de iterações n_{max} parece aumentar com a dimensão k utilizada. Para dimensões 2, 4, 6 e 7, os melhores resultados de SNR_{seg} para taxa de 1,0 $bit/amostra$ foram obtidos com 2, 4, 6 e 7 iterações, respectivamente. Para dimensões 2, 6, 8 e 10, os melhores resultados para taxa de 0,5 $bit/amostra$ foram obtidos com 2, 3, 3, e 5 iterações, respectivamente.

i	w_{i1}	w_{i2}	w_{i3}
1	0.01703	-0.00685	0.01518
2	0.01918	0.01369	-0.00488
3	0.00067	-0.01786	0.00684
4	0.01648	-0.01327	0.00253
5	-0.00430	0.00702	0.00683
6	-0.02005	-0.01700	-0.01792
7	0.01160	0.01677	0.01260
8	-0.01106	0.01282	0.00324

Tabela 4.12: Dicionário inicial.

i	w_{i1}	w_{i2}	w_{i3}
1	0.19523	0.19957	0.17855
2	0.05213	0.05132	0.04625
3	-0.00039	-0.00106	-0.00068
4	0.00367	0.00772	0.00852
5	-0.01867	-0.01663	-0.01466
6	-0.08341	0.01807	0.10741
7	-0.05850	-0.06267	-0.06331
8	-0.16418	-0.17680	-0.16393

Tabela 4.13: Dicionário após duas iterações.

i	w_{i1}	w_{i2}	w_{i3}
1	0.21772	0.22911	0.20475
2	0.08465	0.07525	0.05396
3	0.00009	-0.00046	-0.00051
4	0.01136	0.01475	0.01650
5	-0.02619	-0.02786	-0.02878
6	-0.09422	0.02358	0.13044
7	-0.07796	-0.08377	-0.08162
8	-0.18272	-0.20874	-0.19121

Tabela 4.14: Dicionário treinado.

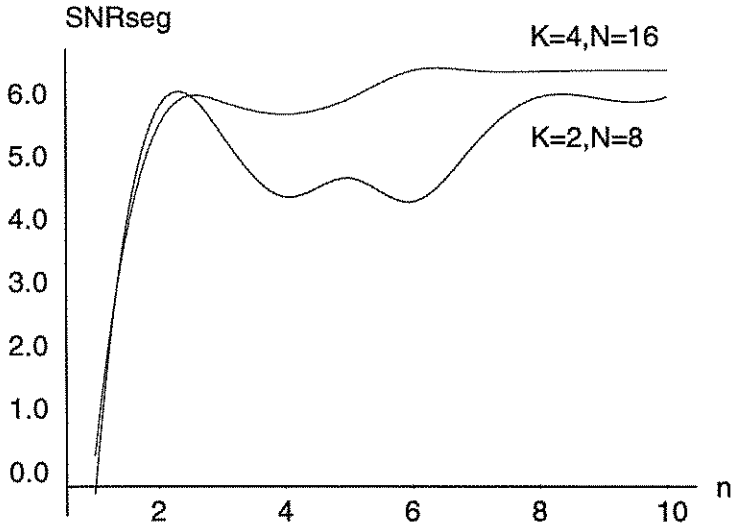


Figura 4.35: SNR_{seg} em função da iteração.

Capítulo 5

Análise de Componentes Principais Aplicada à Quantização Vetorial

5.1 Funções de Autocorrelação e Autocovariância

Seja um processo aleatório discreto $\{X(n)\}$, $n = 0, \pm 1, \pm 2, ...$ Sejam $X(m)$ e $X(n)$ duas variáveis aleatórias desse processo, com médias $\mu_{x(m)}$ e $\mu_{x(n)}$, respectivamente.

A função de autocorrelação do processo $\{X(n)\}$ é expressa por 5.1:

$$R_{xx}(m,n) = E[X(m)X(n)] \tag{5.1}$$

e a função de autocovariância é dada por

$$\begin{aligned} C_{xx}(m,n) &= E[(X(m) - \mu_{x(m)})(X(n) - \mu_{x(n)})] && (5.2) \\ &= R_{xx}(m,n) + \mu_{x(m)}\mu_{x(n)} && (5.3) \end{aligned}$$

Para seqüências com média zero, as funções de autocovariância e de autocorrelação são iguais, isto é:

$$R_{xx}(m,n) = C_{xx}(m,n), \text{ se } \mu_{x(m)} = 0 \text{ ou } \mu_{x(n)} = 0 \tag{5.4}$$

As funções de autocorrelação e de autocovariância, portanto, dependem de duas variáveis temporais m e n. Em outras palavras, essas funções dependem do instante de tempo específico m e do intervalo de tempo $k = n - m$.

Para processos estacionários em amplo senso, as funções de autocorrelação e autocovariância não dependem do instante específico de tempo, podendo ser escritas como

$$R_{xx}(k) = E[X(n)X(n+k)] \tag{5.5}$$

$$C_{xx}(k) = E[(X(n) - \mu_x)(X(n+k) - \mu_x)] \tag{5.6}$$

Dentre as propriedades da função de autocorrelação, destacam-se:

$$R_{xx}(k) = R_{xx}(-k) = R_{xx}(|k|); \tag{5.7}$$

$$R_{xx}(k) \leq R_{xx}(0), \text{ para todo } k; \tag{5.8}$$

$$R_{xx}(0) = E[X^2(n)] = \sigma_x^2 + \mu_x^2 \tag{5.9}$$

onde μ_x e σ_x^2 representam, respectivamente, a média e a variância do processo $\{X(n)\}$ e podem ser estimadas por

$$\mu_x = \frac{1}{N} \sum_{n=0}^{N-1} x(n) \tag{5.10}$$

$$\sigma_x^2 = \frac{1}{N} \sum_{n=0}^{N-1} [x(n) - \mu_x]^2 \tag{5.11}$$

É muito comum a utilização da função de autocorrelação normalizada, definida como

$$\rho_k = \rho_{xx}(k) = \frac{R_{xx}(k)}{R_{xx}(0)} \tag{5.12}$$

Por conveniência, será utilizada, a partir deste momento, a notação matricial.

Seja uma seqüência de N amostras denotada pelo vetor

$$\mathbf{x} = \{x(k)\};\ k = 0, 1, ..., N-1 \tag{5.13}$$

Seja a seqüência de variáveis aleatórias observadas, denotada pelo vetor

$$\mathbf{X} = \{X(k)\};\ k = 0, 1, ..., N-1 \tag{5.14}$$

A matriz de autocorrelação será denotada por

$$\mathbf{R}_{xx} = \{R_{xx}(\mid k-l \mid)\} = R_{xx}(0)\{\rho_{xx}(\mid k-l \mid)\};\ k, l = 0, 1, ..., N-1; \tag{5.15}$$

ou seja,

$$\mathbf{R}_{xx} = \begin{bmatrix} R_0 & R_1 & R_2 & ... & R_{N-1} \\ R_1 & R_0 & R_1 & ... & R_{N-2} \\ R_2 & R_1 & R_0 & ... & R_{N-3} \\ . & . & . & ... & . \\ R_{N-1} & . & . & ... & R_0 \end{bmatrix} \tag{5.16}$$

$$= R_0 \begin{bmatrix} 1 & \rho_1 & \rho_2 & ... & \rho_{N-1} \\ \rho_1 & 1 & \rho_1 & ... & \rho_{N-2} \\ \rho_2 & \rho_1 & 1 & ... & \rho_{N-3} \\ . & . & . & ... & . \\ \rho_{N-1} & . & . & ... & 1 \end{bmatrix} \tag{5.17}$$

onde foi utilizado a notação

$$R_{|k-l|} = R_0\rho(\mid k-l \mid) = R_{xx}(\mid k-l \mid) = E[X(n+l)X(n+k)] \tag{5.18}$$

para o elemento na k-ésima linha e l-ésima coluna.

Observa-se que a matriz de autocorrelação é simétrica. Além disso, elementos ao longo de qualquer diagonal são iguais. Uma matriz com essas características é chamada matriz Toeplitz simétrica.

A matriz de autocorrelação também pode ser escrita como

$$\mathbf{R}_{xx} = E[\mathbf{X}\mathbf{X}^T] \tag{5.19}$$

onde T denota a transposta de um vetor.

De modo semelhante, a matriz de autocovariância pode ser definida como

$$\mathbf{C}_{xx} = \mathbf{R}_{xx} - \boldsymbol{\mu}_x\boldsymbol{\mu}_x^t = \{C_{xx}(\mid k-l \mid)\};\ k,l = 0,1,...,N-1 \tag{5.20}$$

onde $\boldsymbol{\mu}_x$ é o vetor-média, com todos elementos iguais a μ_x:

$$\boldsymbol{\mu}_x = E[\mathbf{X}] = \{E[X(k)]\};\ k = 0,1,2,...,N-1 \tag{5.21}$$

Convém salientar que $\mathbf{R}_{xx} = \mathbf{C}_{xx}$ se $\boldsymbol{\mu}_x = 0$

5.2 Autovalores e Autovetores

Associado à matriz de correlação $\mathbf{R}_{xx}$ existe um conjunto de N autovalores distintos λ_i e N autovetores distintos $\mathbf{z}_i$ definidos por

$$\mathbf{R}_{xx}\mathbf{z}_i = \lambda_i\mathbf{z}_i;\ i = 0,1,...,N-1 \tag{5.22}$$

Dentre as propriedades de autovetores e autovalores, destacam-se:

- Autovetores são ortogonais, ou seja,

$$\mathbf{z}_i^T\mathbf{z}_j = 0 \text{ para } i \neq j; \tag{5.23}$$

- Autovetores definem $\mathbf{R}_{xx}$ na forma

$$\mathbf{R}_{xx} = \sum_{i=0}^{N-1} \lambda_i\mathbf{z}_i\mathbf{z}_i^T \tag{5.24}$$

- Autovalores apresentam as propriedades:

$$\mid \mathbf{R}_{xx} \mid = \prod_{j=0}^{N-1} \lambda_j; \tag{5.25}$$

$$traço(\mathbf{R}_{xx}) = \sum_{j=0}^{N-1} \lambda_j \tag{5.26}$$

Para um processo com média zero,

$$traço(\mathbf{R}_{xx}) = N\sigma_x^2 \tag{5.27}$$

Para qualquer N, portanto, o valor médio dos autovalores é igual à variância do processo:

$$\sigma_x^2 = \frac{1}{N} \sum_{j=0}^{N-1} \lambda_j \tag{5.28}$$

5.3 Transformação

Uma transformação é aplicada no vetor de entrada N-dimensional

$$\mathbf{x} = \{x(n)\};\ n = 0, 1, 2, ..., N-1 \tag{5.29}$$

resultando num vetor de atributos N-dimensional

$$\boldsymbol{\theta} = \{\theta(i)\};\ i = 0, 1, ..., N-1 \tag{5.30}$$

A transformada direta é definida por:

$$\theta(i) = \sum_{n=0}^{N-1} a(i,n)x(n) \text{ para } i = 0, 1, ..., N-1 \tag{5.31}$$

onde $a(i,n)$ é o núcleo de transformação direta e $\theta(i)$ é a i-ésima componente do vetor de atributos.

A transformada inversa, que permite recuperar o vetor de entrada, é dada por:

$$x(n) = \sum_{n=0}^{N-1} b(n,i)\theta(i) \text{ para } n = 0, 1, ..., N-1 \tag{5.32}$$

onde $b(n,i)$ é o núcleo de transformação inversa.

A seguinte notação matricial será adotada:

$$\boldsymbol{\theta} = \mathbf{A}\mathbf{x}; \tag{5.33}$$

$$\mathbf{x} = \mathbf{B}\boldsymbol{\theta}; \tag{5.34}$$

$$\mathbf{B} = \mathbf{A}^{-1} \tag{5.35}$$

O vetor de entrada $\mathbf{x}$ e o vetor atributos $\boldsymbol{\theta}$ serão representados como matrizes Nx1, de modo que

$$\mathbf{x}^T = \{x(0)\ x(1)\ ...\ x(N-1)] \tag{5.36}$$

$$\boldsymbol{\theta}^T = \{\theta(0)\ \theta(1)\ ...\ \theta(N-1)] \tag{5.37}$$

As matrizes $\mathbf{A}$ e $\mathbf{B}$ serão definidas do seguinte modo:

$$\mathbf{A} = \{a(m,n)\}_{m,n=0,1,...,N-1}; \tag{5.38}$$

$$\mathbf{B} = \{b(m,n)\}_{m,n=0,1,...,N-1} \tag{5.39}$$

Para simplificar, será adotada a sequinte notação compacta:

$$\mathbf{B} = \{\mathbf{b}_i\}_{i=0,1,...,N-1} \tag{5.40}$$

$$\mathbf{b}_i = \{b(m,i)\}_{m=0,1,...,N-1} \tag{5.41}$$

onde $\mathbf{b}_k$ são vetores de base.

A matriz $\mathbf{B}$, portanto, é representada da seguinte forma:

$$\mathbf{B} = \begin{bmatrix} b(0,0) & b(0,1) & ... & b(0,N-1) \\ b(1,0) & b(1,1) & ... & b(1,N-1) \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ b(N-1,0) & b(N-1,1) & ... & b(N-1,N-1) \end{bmatrix} \tag{5.42}$$

ou ainda,

$$\mathbf{B} = [\mathbf{b}_0 \ \mathbf{b}_1 \ ... \ \mathbf{b}_{N-1}] \tag{5.43}$$

onde os vetores de base $\mathbf{b}_i$, $i = 0, 1, ..., N-1$ são as colunas de $\mathbf{B}$.

Note que o conjunto de equações dado por (5.32) implica

$$\mathbf{x} = \theta(0)\mathbf{b}_0 + \theta(1)\mathbf{b}_1 + ... + \theta(N-1)\mathbf{b}_{N-1} \tag{5.44}$$

isto é,

$$\mathbf{x} = \mathbf{B}\boldsymbol{\theta} = \sum_{n=0}^{N-1} \theta(i)\mathbf{b}_i \tag{5.45}$$

O vetor de entrada $\mathbf{x}$, portanto, é a soma ponderada dos vetores de base, onde os pesos são as componentes do vetor de atributos, ou seja, do vetor transformado.

Cumpre salientar que em codificação por transformada, o vetor quantizado $\mathbf{y}$ também é uma soma ponderada de vetores de base, contudo, os pesos são versões quantizadas de θ_i:

$$\mathbf{y} = \sum_{n=0}^{N-1} u(i)\mathbf{b}_i; \ u(i) = Q[\theta(i)] \tag{5.46}$$

A codificação por transformada está ilustrada na Figura 5.1, onde T denota a transformada direta do vetor $\mathbf{x} = \{x(n)\}$, $n = 0, 1, 2, ..., N-1$; $\boldsymbol{\theta} = \{\theta(i)\}$, $i = 0, 1, 2, ..., N-1$, representa o vetor de atributos (vetor transformado), Q denota a operação de quantização; $\mathbf{u} = \{u(n)\}$, $n = 0, 1, 2, ..., N-1$, representa a versão quantizada de $\boldsymbol{\theta}$; T^{-1} denota a transformada inversa e $\hat{\mathbf{x}}$ representa a versão recuperada do vetor de entrada $\mathbf{x}$.

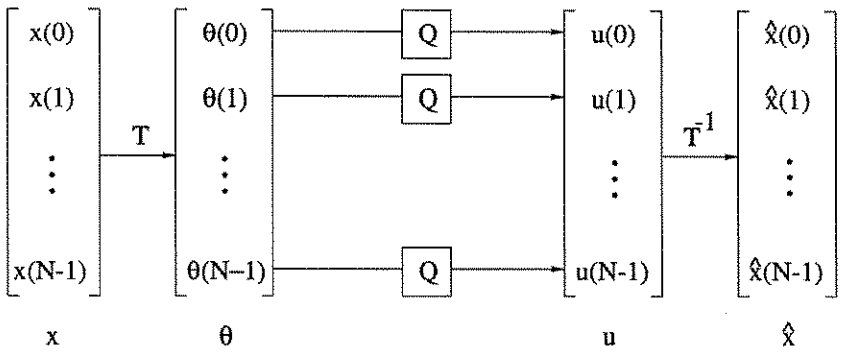


Figura 5.1: Codificação por transformada.

5.4 Transformações Ortogonais

A classe de transformações ortogonais é definida por:

$$\mathbf{A}^{-1} = \mathbf{A}^T \tag{5.47}$$

o que implica

$$\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I} \tag{5.48}$$

onde $\mathbf{I}$ é a matriz identidade de ordem N.

Uma matriz real é ortogonal se e somente se suas linhas e colunas formam um conjunto ortogonal, de modo que a transformação inversa é definida pela transposta de $\mathbf{A}$:

$$\mathbf{B} = \mathbf{A}^{-1} = \mathbf{A}^T \tag{5.49}$$

Isto implica que os vetores de base são as linhas de $\mathbf{A}$, conforme mostra a Figura 5.2, e que

$$\mathbf{b}_i^T.\mathbf{b}_j = \delta_{ij} \tag{5.50}$$

A função δ_{ij} é definida como

$$\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{5.51}$$

Ortogonalidade é uma propriedade necessária para os vetores de base serem utilizados para decompor um vetor de entrada em componentes descorrelacionadas em um espaço N-dimensional. Ortonormalidade de vetores de base é uma propriedade mais forte; isto conduz a transformadas necessárias em codificação por transformada para tornar a média das variâncias dos elementos de $\boldsymbol{\theta}$ igual a σ_x^2, a variância dos elementos de $\mathbf{x}$.

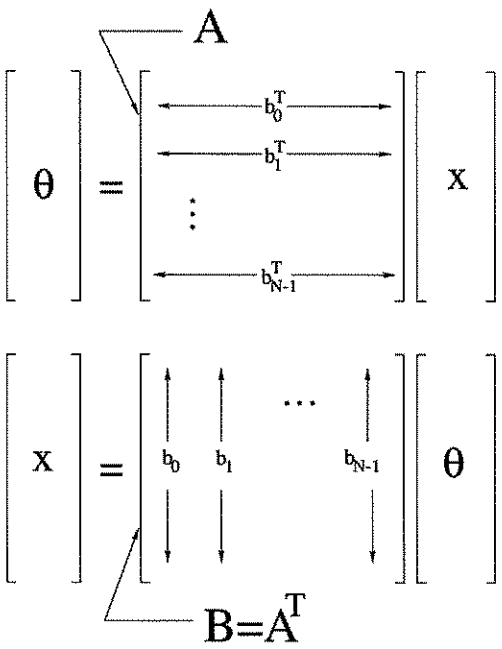


Figura 5.2: Matriz de transformação $\mathbf{A}$, matriz inversa $\mathbf{B} = \mathbf{A}^{-1} = \mathbf{A}^T$ e vetores de base $\mathbf{b}_i$ [1].

A Figura 5.3 mostra os vetores de base para várias transformadas de ordem $N = 8$. Em cada um dos quatro conjuntos de vetores de base, todo os vetores, exceto o vetor da primeira linha, são seqüências com média zero. Os vetores de base na primeira linha, são seqüências com amplitude constante, em todos o casos, exeto para a transformada de Karhunen Loeve (KLT). Os vetores de base na transformada Discreta de

Fourier (DFT) são simplesmente senos e cossenos. Na transformada discreta de Walsh Hadamard (DWHT), os vetores de base são ondas quadradas. Na transformada discreta de cossenos (DCT), os vetores de base são funções cosseno com padrões especiais de fase e amplitude. Os vetores de base nas transformadas DFT, DWHT e DCT são independentes do sinal de entrada. Por outro lado, na transformada KLT, os vetores de base, também chamados de autovetores, são definidos pela estatística do sinal de entrada [1].

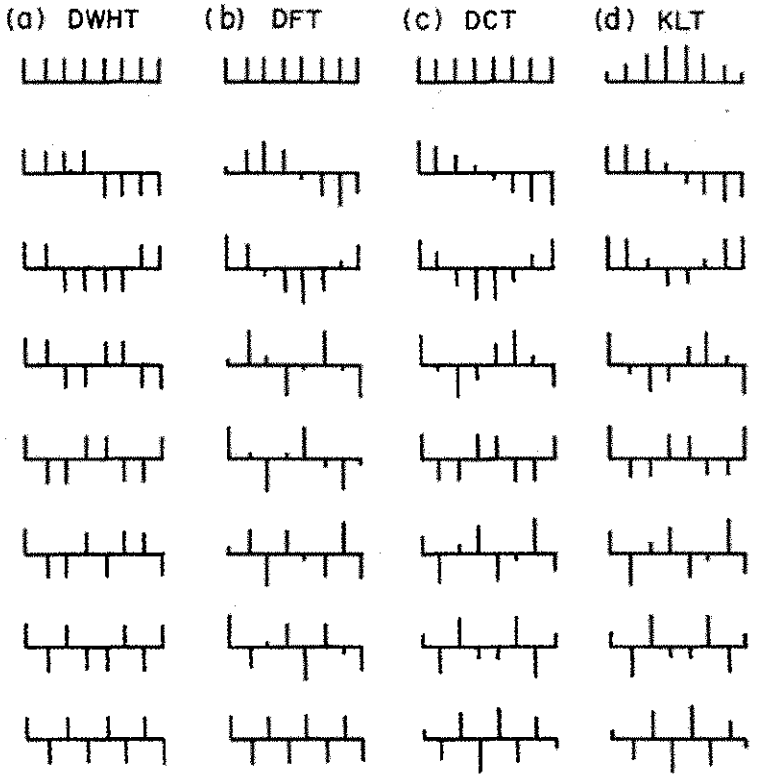


Figura 5.3: Vetores de base para $N = 8$ nas transformadas (a) DWHT, (b) DFT, (c) DCT e (d) KLT [1].

As transformadas dependentes de entrada são de implementação mais difícil, contudo, apresentam as melhores propriedades de descorrelação da entrada e ordenamento de variância. Os vetores de base da KLT também apresentam a maior semelhança

possível com os segmentos de forma de onda típicos do sinal de entrada cuja estatística foi utilizada na transformada KLT [1].

Uma conseqüência deste fato é que a dimensionalidade N necessária para um determinado erro de representação é mínima quando a representação se dá em termos de vetores de base da KLT (autovetores).

Admita-se que o vetor $\mathbf{x}$ seja tomado de uma fonte escalar estacionária de média zero. Vetores sucessivos, portanto, constituem uma fonte vetorial aleatória $\mathbf{X}$, cujos elementos são variáveis aleatórias $X(n)$; $n = 0, 1, ..., N-1$.

A seguir, será demonstrado que se A é ortogonal ($\mathbf{A}^{-1} = \mathbf{A}^T$), a média das variâncias $E[\Theta^2(i)] = \sigma^2(i)$ dos atributos $\theta(i)$ é igual à variância da entrada:

$$\begin{aligned} \frac{1}{N}\sum_{i=1}^{N-1}\sigma_i^2 &= \frac{1}{N}\sum_{i=1}^{N-1}E[\Theta^2(i)] = \frac{1}{N}E[\mathbf{\Theta}^T\mathbf{\Theta}] \\ &= \frac{1}{N}E[\mathbf{X}^T\mathbf{A}^T\mathbf{A}\mathbf{X}] = \frac{1}{N}E[\mathbf{X}^T\mathbf{X}] \\ &= \frac{1}{N}\sum_{i=1}^{N-1}E[X^2(i)] = \frac{1}{N}\sum_{i=1}^{N-1}\sigma_x^2 = \sigma_x^2 \end{aligned} \tag{5.52}$$

5.4.1 A Transformada de Karhunen-Loeve (KLT)

Para qualquer matriz $\mathbf{A}$, as variâncias dos atributos são os elementos da diagonal de $\mathbf{R}_{\theta\theta} = \{R_{\theta\theta}(i,l)\}; i, l = 0, 1, ..., N-1$:

$$\sigma^2(i) = E[\Theta^2(i)] = R_{\theta\theta}(i,i) \tag{5.53}$$

$$\mathbf{R}_{\theta\theta} = E[\mathbf{\Theta}\mathbf{\Theta}^T] = E[\mathbf{A}\mathbf{X}\mathbf{X}^T\mathbf{A}^T] = \mathbf{A}\mathbf{R}_{xx}\mathbf{A}^T \tag{5.54}$$

A transformada KLT é definida pela matriz $\mathbf{L}$, cujas linhas são os autovetores de $\mathbf{R}_{xx}$.

Conforme abordado anteriormente, associado a uma matriz $\mathbf{R}_{xx}$ existe um conjunto de autovalores $\lambda_i; i = 0, 1, ..., N-1$, e autovetores $\mathbf{l}_i; i = 0, 1, ..., N-1$, definidos por

$$\mathbf{R}_{xx}\mathbf{l}_i = \lambda_i\mathbf{l}_i \tag{5.55}$$

Como $\mathbf{R}_{xx}$ é uma matriz real e simétrica, então os autovalores são reais e existem exatamente N autovetores ortogonais, que podem ser escolhidos de modo a serem ortonormais:

$$\mathbf{l}_i^T \mathbf{l}_j = \delta_{ij} \tag{5.56}$$

A transformada KLT é definida, portanto, pela matriz $\mathbf{L}$:

$$\boldsymbol{\theta} = \mathbf{L}\mathbf{x} \tag{5.57}$$

$$\mathbf{x} = \mathbf{L}^T \boldsymbol{\theta} = \sum_{i=1}^{N-1} \theta(i)\mathbf{l}_i \tag{5.58}$$

Observa-se que o vetor de entrada $\mathbf{x}$ é uma superposição de autovetores que são obtidos a partir da estatística do sinal de entrada (via $\mathbf{R}_{xx}$) e que representam seqüências típicas do sinal de entrada. Como os vetores de base dependem do sinal de entrada, uma menor parcela desses vetores pode ser usada, em média, para aproximar uma determinada entrada, relativamente às representações que utilizam vetores de base independentes do sinal.

Da equação 5.54,

$$\mathbf{R}_{\theta\theta} = \mathbf{L}\mathbf{R}_{xx}\mathbf{L}^T \tag{5.59}$$

Da equação 5.55,

$$\mathbf{R}_{\theta\theta} = \begin{bmatrix} \mathbf{l}_0^T \\ \mathbf{l}_1^T \\ \vdots \\ \mathbf{l}_{N-1}^T \end{bmatrix} \begin{bmatrix} \lambda_0\mathbf{l}_0 & \lambda_1\mathbf{l}_1 & \cdots & \lambda_{N-1}\mathbf{l}_{N-1} \end{bmatrix} \tag{5.60}$$

$$= [\lambda_j \mathbf{l}_i^T \mathbf{l}_j] = [\lambda_j \delta_{ij}] \tag{5.61}$$

$$= \begin{bmatrix} \lambda_0 & 0 & 0 & \dots & 0 \\ 0 & \lambda_1 & & \cdots & . \\ 0 & & \lambda_2 & \dots & . \\ \vdots & & & & \\ 0 & . & . & \dots & \lambda_{N-1} \end{bmatrix} = \boldsymbol{\lambda} \tag{5.62}$$

e, por 5.59 e 5.55,

$$\mathbf{L}\mathbf{R}_{xx} = \boldsymbol{\lambda}\mathbf{L} \tag{5.63}$$

É importante observar que os elementos de $\mathbf{R}_{\theta\theta}$ são $E[\Theta(i)\Theta(j)]$. A transformada KLT, portanto, diagonaliza $\mathbf{R}_{\theta\theta}$, ou seja, os atributos são descorrelacionados. Note também que

$$E[\Theta^2(i)] = \lambda_i;\ i = 0, 1, ..., N-1 \tag{5.64}$$

ou seja, os autovalores são as variâncias dos atributos no caso da KLT.

5.5 Projeto de Dicionários Através de Análise de Componentes Principais

O algoritmo proposto, denominado PCA (*Principal Component Analysis*), é baseado em Análise de Componentes Principais (PCA) [50, 51] de uma seqüência típica de fala e consiste dos seguintes passos:

(i) São definidos o tamanho do dicionário N e a dimensão k;

(ii) A partir da matriz de covariância $\mathbf{C_{xx}}$, de ordem k, de uma seqüência típica de fala $\mathbf{X}$, são determinados os autovalores λ_i e os autovetores $\mathbf{z}_i$; $i = 1, 2, ..., N$, definidos por:

$$\mathbf{C_{xx}}\mathbf{z}_i = \lambda_i \mathbf{z}_i \tag{5.65}$$

(iii) São definidos $L \leq K$ vetores $\mathbf{z}_i$ ao longo de cujas direções os vetores-código devem ser alocados. O número L é escolhido de acordo com o valor percentual relativo $\lambda_{i,p}$ de cada autovalor λ_i:

$$\lambda_{i,p} = \frac{\lambda_i}{\sum_{i=1}^{N} \lambda_i} \tag{5.66}$$

Apenas os L autovetores mais significativos, os quais definem as L direções principais, são escolhidos, de acordo com os valores mais significativos de $\lambda_{i,p}$.

(iv) Para cada $\mathbf{z}_i$ escolhido, é determinado um vetor $\tilde{\mathbf{z}}_i = r\mathbf{z}_i$ $(i = 1, 2, ..., L)$, onde o escalar r é o recíproco do valor absoluto da componente de maior valor absoluto. Portanto, $\tilde{\mathbf{z}}_i$ tem no mínimo uma componente com valor absoluto 1, e as demais componentes apresentam valores absolutos situados na faixa de 0 a 1;

(v) Seja N_i o número de vetores-código a serem alocados ao longo das direções definidas pelo vetor $\tilde{\mathbf{z}}_i$. Cada N_i é escolhido como uma fração de N, proporcionalmente a $\lambda_{i,p}$, de modo que

$$N = \sum_{i=1}^{L} N_i \tag{5.67}$$

(vi) Finalmente, N_i vetores-código $\mathbf{w}_{i,n_i}$ são alocados em cada i-ésima direção principal, de acordo com:

$$\mathbf{w}_{i,n_i} = f(n_i, \lambda_i)\tilde{\mathbf{z}}_i \tag{5.68}$$

onde $i = 1, 2, ..., L$, $n_i = 1, 2, ..., N_i$ e os escalares $f(n_i, \lambda_i)$ são determinados assumindo-se que cada direção principal apresenta distribuição gaussiana com variâncias ${\sigma_i}^2 = \lambda_i$: os escalares $f(n_i, \lambda_i)$ são determinados de tal forma que a área sob a função densidade de probabilidades gaussiana seja igualmente dividida.

Em suma, um conjunto de L autovetores é determinado e um número de vetores-código N_i (uma fração de N) é então alocado em cada direção principal definida pelos autovetores $\mathbf{z}_i$ $(i = 1, 2, ..., L)$ de acordo com considerações efetuadas nos autovalores (variâncias).

O método proposto difere do algoritmo descrito em [46] uma vez que introduz uma espécie de normalização nos autovetores, conforme descrito em (iv): a multiplicação dos autovetores normalizados $\tilde{\mathbf{z}}_i$ pelos escalares $f(n_i, \lambda_i)$ permite que os vetores-código resultantes incorporem simultaneamente duas características importantíssimas do sinal

de voz: as direções principais (via autovetores) e a distribuição estatística dos vetores (via autovalores) em cada direção. Convém salientar que os autovetores, ou vetores de base da transformada KLT, apresentam a maior semelhança possível com os segmentos de forma de onda típicos do sinal de entrada cuja estatística foi utilizada na transformada KLT [1].

Convém salientar que a técnica proposta, apesar de se fundamentar na transformada KLT, é bastante diferente da técnica de codificação por transformada: não é necessário cálculo da KLT em tempo real; autovalores e autovetores não são calculados para cada vetor da amostras, mas sim uma única vez, para a seqüência de voz como um todo.

5.6 Simulações Realizadas com o Algoritmo PCA

O primeiro experimento consistiu em proceder uma avaliação da coerência do método proposto. Para tanto, obteve-se o gráfico de um sinal de voz típico, correspondente a 10 frases foneticamente balanceadas [71], apresentado na Figura 5.4, e do dicionário projetado através do algoritmo PCA, ilustrado na Figura 5.5. Conforme se pode observar, o algoritmo alocou corretamente os vetores-código ao longo da direção correspondente à componente principal do sinal. Além disso, é importante observar que um maior número de vetores-código foi corretamente alocada na região correspondente às baixas amplitudes.

Como parte dos experimentos, foi avaliado o desempenho do algoritmo proposto para diversas taxas de uma quantização vetorial com dimensão $k = 4$. Para avaliação do algoritmo, utilizou-se uma seqüência de teste relativamente longa, constituído de 29120 amostras (3,64s), correspondente ao conjunto de setenças "*O sol ilumina a fachada de tarde. Trabalhou mais do que podia*". Conforme se observa na Figura 5.6, para quase todas as taxas consideradas, o algoritmo PCA apresenta desempenho superior ao apresentado pelo algoritmo LBG e para algumas taxas PCA apresenta quase o mesmo desempenho do algoritmo KMTAN. Conforme se observa na Figura 5.7, para dimensão $k = 8$, o algoritmo PCA supera o LBG para a taxa compreendida entre 0,45 e 0,75 $bit/amostra$; entretanto, para todas as taxas o algoritmo KMTAN é superior ao PCA para $k = 8$.

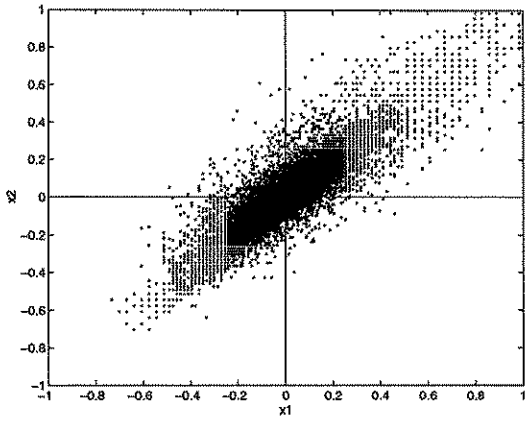


Figura 5.4: Sinal de voz utilizado para determinação da matriz de covariância, correspondente ao conjunto de 10 frases foneticamente balanceadas (18,76s, 75040 vetores). x_1 e x_2 representam, respectivamente, a primeira e a segunda componente do vetor $\mathbf{x} \in R^2$.

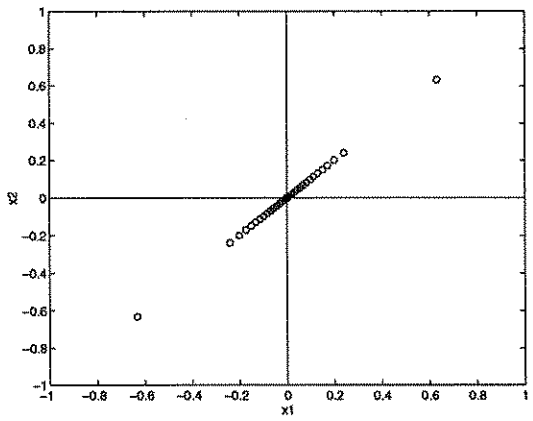


Figura 5.5: Dicionário obtido com o algoritmo PCA: $k = 2$ e $N = 32$. x_1 e x_2 representam, respectivamente, a primeira e a segunda componente do vetor $\mathbf{x} \in R^2$.

Um outro conjunto de experimentos consistiu em avaliar o desempenho do algoritmo para diferentes dimensões à taxa de 1,0 *bit/amostra*. Conforme se observa na Figura 5.8, para as dimenões 3, 4 e 5, PCA apresenta desempenho semelhante ao apresentado pelo algoritmo KMTAN, superanto o algoritmo LBG. Para dimensões 6, 7 e 8, entretanto, PCA é superado pelos algoritmos LBG e KMTAN.

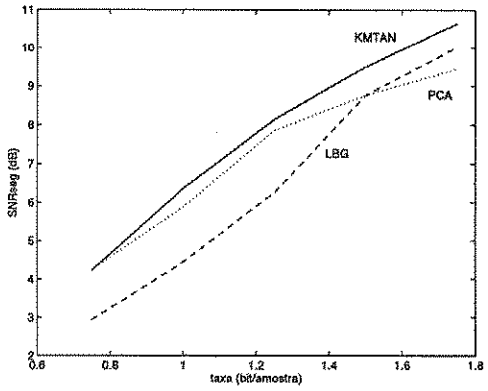


Figura 5.6: Comparação dos algoritmos LBG, KMTAN e PCA para compressão de voz para dimensão $k = 4$.

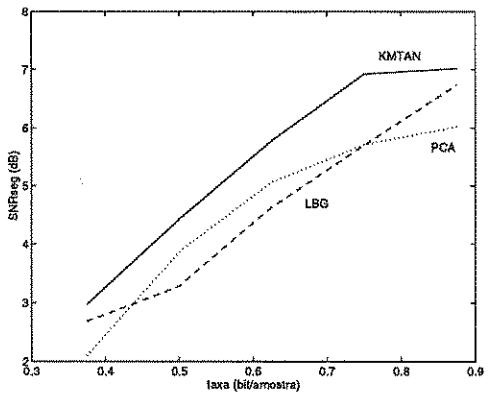


Figura 5.7: Comparação dos algoritmos LBG, KMTAN e PCA para compressão de voz para dimensão $k = 8$.

É importante observar que o método proposto gera dicionários estruturados, con-

forme se observa na Figura 5.9, ao contrário dos dicionários gerados pelos algoritmos LBG e KMTAN, sem estrutura alguma, conforme se verifica na Figura 5.10.

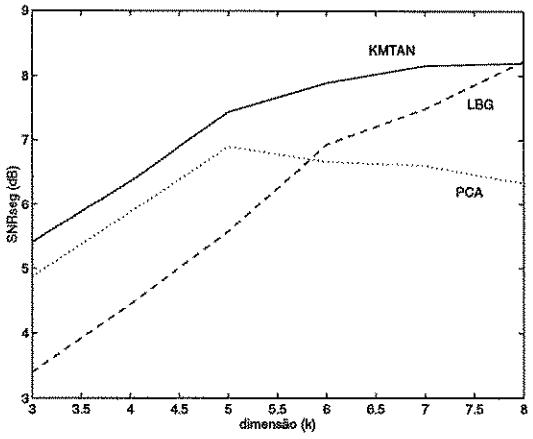


Figura 5.8: Comparação dos algoritmos LBG, KMTAN e PCA para compressão de voz a 1,0 *bit/amostra*.

Dentre as principais vantagens apresentadas pelo algoritmo proposto, podem ser citadas:

- simplicidade e rapidez;
- geração de dicionários estruturados, o que pode ser utilizado para facilitar a etapa de codificação;
- independência de dicionário inicial;
- inexistência de parâmetros a serem ajustados;

De fato, o algoritmo proposto é bastante simples e intuitivo. Os vetores-código são calculados, de acordo com a estatística do sinal, ao contrário de abordagens tradicionais, como por exemplo os algoritmos LBG e Kohonen, que necessitam de uma seqüência de treino para atualizar iterativamente os vetores-código. O algoritmo descrito não apresenta o incômodo problema de ajuste de parâmetros. Além disso, o algoritmo PCA não requer a definição de um dicionário inicial.

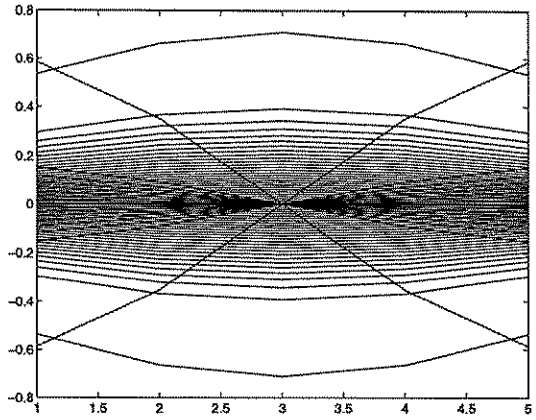


Figura 5.9: Dicionário de padrões produzido pelo algoritmo PCA: $k = 5$ e $N = 76$. Cada uma das 76 curvas corresponde à ligação dos pontos correspondentes às componentes (amostras) dos vetores de dimensão 5.

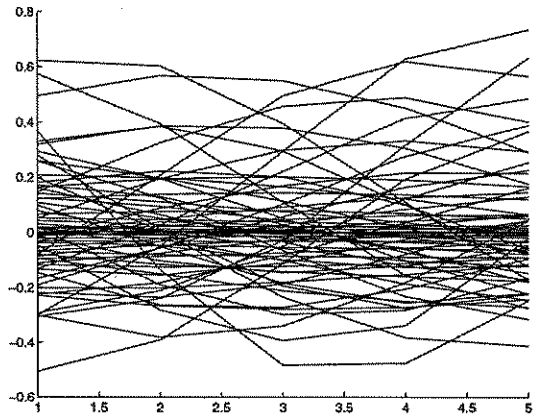


Figura 5.10: Dicionário de padrões produzido pelo algoritmo KMTAN: $k = 5$ e $N = 76$. Cada uma das 76 curvas corresponde à ligação dos pontos correspondentes às componentes (amostras) dos vetores de dimensão 5.

Capítulo 6

Conclusão

Como técnica de compressão de sinais, a quantização vetorial tem sido bastante estudada para aplicações envolvendo voz e imagem, apresentando excelente desempenho, isoladamente ou como parte integrante de sistemas mais complexos, permitindo elevadas taxas de compressão.

Durante o desenvolvimento deste trabalho, foram propostos e avaliados alguns algoritmos para projeto de quantizadores vetoriais. Neste contexto, foram utilizadas duas abordagens: a utilização de modificações introduzidas no algoritmo de treinamento da rede neural de Kohonen e o uso de uma técnica baseada em Análise de Componentes Principais.

A avaliação dos algoritmos KMTAU e KMTAN, resultantes da introdução de modificações no algoritmo de Kohonen, foi realizada através de simulações envolvendo sinais com distribuição gaussiana e de Gauss-Markov, sinais de voz e imagens.

Nos experimentos envolvendo as distribuições mencionadas, observou-se o enorme potencial da quantização vetorial, que apresentou desempenho bastante próximo dos limites teóricos fornecidos pela Teoria da Distorção Versus Taxa, formulada por Shannon. Nesses experimentos, ficou caracterizada a eficiência do algoritmo KMTAU: os dicionários projetados incorporaram as características estatísticas do sinal de entrada e apresentaram excelentes resultados em termos de entropia normalizada e desempenho superior ao apresentado pelo algoritmo LBG.

Nas simulações envolvendo forma de onda de voz, após algumas investigações,

justificou-se a opção por se utilizarem seqüências de treino curtas nos algoritmos modificados de Kohonen, como estratégia de redução do tempo de processamento. Nessas simulações, observou-se que, ao contrário do algoritmo LBG, os algoritmos propostos não são sensíveis ao dicionário inicial. Durante os experimentos que envolveram a compressão de 8,0 *bit/amostra* (sinal original) para 1,0 *bit/amostra* (sinal reconstruído), ficou bastante clara a superioridade da quantização vetorial sobre a escalar, tanto em relação à reprodução da forma de onda do sinal original como no tocante à qualidade subjetiva do sinal quantizado. Com relação à quantização vetorial a baixas taxas, observou-se alguma limitação com relação à faixa dinâmica, problema que pode ser amenizado através da incorporação de esquemas de quantização adaptativa. Em todos os experimentos, os algoritmos propostos, além de incorporarem importantes características estatísticas do sinal de voz, apresentaram desempenho superior ao apresentado pelo algoritmo LBG. Em se tratando do algoritmo KMTAU, em particular, investigou-se o comportamento da relação sinal-ruído segmental (SNR_{seg}) em função da iteração, chegando-se a uma importante conclusão: o algoritmo é capaz de escapar de máximos locais ruins, tendo em vista o comportamento não assintótico de SNR_{seg}, de modo que um bom dicionário pode ser obtido em iterações intermediárias, não necessariamente na última iteração.

Com relação às simulações envolvendo imagens, os algoritmos propostos também apresentaram desempenho superior ao apresentado pelo algoritmo LBG. Nos experimentos que envolveram a compressão de 8,0 *bpp* (imagem original) para taxas compreendidas entre 0,312 *bpp* e 0,625 *bpp* (imagem reconstruída), o algoritmo KMTAU apresentou excelente desempenho no tocante à avaliação subjetiva das imagens processadas.

A avaliação do algoritmo PCA, baseado em Análise de Componentes Principais, foi realizada através de simulações realizadas em sinais de voz. Observou-se a capacidade que esse algoritmo apresenta para alocação de vetores-código ao longo das direções principais do sinal de voz. Em grande parte dos experimentos, fixada a dimensão, para uma determinada faixa de taxas (em bit/amostra), o algoritmo PCA superou o LBG e apresentou desempenho bastante próximo do apresentado pelo algoritmo KMTAN. Esse comportamento se manteve para as dimensões 3, 4 e 5 à taxa de 1,0 *bit/amostra*.

Durante as simulações, constatou-se que o algoritmo PCA, ao contrário do LBG e do KMTAN, produziu dicionários estruturados, o que pode facilitar a etapa de codificação dos vetores do dicionário. É possível concluir, portanto, que o algoritmo PCA trata-se de uma técnica simples, rápida e eficiente. Ao contrário dos algoritmos LBG, KMTAU e KMTAN, não é necessária a utilização de uma seqüência de treino para atualizar iterativamente os vetores-código. Estes são calculados de acordo com a estatística de um sinal de voz típico; além disso, o algoritmo PCA não oferece parâmetros a serem ajustados. Convém salientar que a introdução da etapa de normalização dos autovetores no algoritmo originalmente proposto em [46] implica incorporação simultânea de duas características importantes do sinal de voz: as direções principais (via autovetores) e a distribuição estatística adequada dos vetores-código (via autovalores) em cada direção.

Para continuação das atividades de pesquisa desenvolvidas, podem ser citadas as seguintes sugestões:

- investigação da complexidade da quantização vetorial, visando a otimização do processo de procura do vetor-código mais próximo (fase de codificação);
- incorporação de esquemas adaptativos às técnicas de quantização vetorial, objetivando melhorar o desempenho em termos da faixa dinâmica do quantizador;
- avaliação da influência do comprimento das seqüências de treino e de teste nos algoritmos decorrentes da introdução de modificações no algoritmo de Kohonen, conforme fundamentação descrita em [72];
- realização de uma avaliação subjetiva mais formal e consistente em imagens e sinais de voz submetidos à compressão.

Cumpre salientar que durante o desenvolvimento deste trabalho foram obtidas importantes conclusões, ressaltando o enorme potencial e a grande eficiência das técnicas propostas para projeto de quantizadores vetoriais voltados para compressão de sinais de voz e imagens.

Bibliografia

[1] Jayant, N. S. and Noll, P. *Digital Coding of Waveforms.* Prentice-Hall, Englewood Cliffs, NJ, 1984.

[2] Beale, R. & Jackson, T. *Neural Computing: An Introduction.* Institute of Physics Publishing, Bristol and Philadelphia, 1990.

[3] Haykin, S. *Digital Communications.* John Wiley & Sons, 1988.

[4] Gersho, A. and Gray, R. M. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, Boston, MA, 1992.

[5] Gray, R. M. Vector Quantization. *IEEE ASSP Magazine*, pages 4–29, April 1984.

[6] Berger, T. *Rate Distortion Theory: A Mathematical Basis for Data Compression.* Prentice-Hall, Englewood Cliffs, NJ, 1971.

[7] Zeger, K., Vaisey, J. and Gersho, A. Globally Optimal Vector Quantizer Design by Stochastic Relaxation. *IEEE Transactions on Signal Processing, Vol. 40, No.2*, pages 310–322, February 1992.

[8] Soleymani, M. R. and Morgera, S. D. An Efficient Nearest Neighbor Search Method. *IEEE Transactions on Communications, Vol. COM-35, No.6*, pages 677–679, June 1987.

[9] Soleymani, M. R. and Morgera, S. D. A Fast MMSE Encoding Technique for Vector Quantization. *IEEE Transactions on Communications, Vol. 37, No.6*, pages 656–659, June 1989.

[10] Paliwal, K. K. and Ramasubramanian, V. Effect of Ordering the Codebook on the Efficiency of the Partial Distance Search Algorithm for Vector Quantization. *IEEE Transactions on Communications, Vol. 37, No.5*, pages 538–540, May 1989.

[11] Jayant, N. S. Coding Speech at Low Bit Rates. *IEEE Spectrum*, pages 58–63, August 1986.

[12] Linde, Y., Buzo, A. and Gray, R. M. An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications, Vol. COM - 28, No.1*, pages 84–95, January, 1980.

[13] Kohonen, T. *Self-Organization and Associative Memory (3rd ed)*. Springer-Verlag, Berlin, 1989.

[14] Kohonen, T. The Self-Organizing Map. *Proceedings of the IEEE, Vol. 78, No. 9*, pages 1464–1480, September 1990.

[15] Haykin, S. *Neural Networks - A Comprehensive Foundation.* IEEE Press, Englewood Cliffs - NJ, 1994.

[16] Hertz, J. and Krogh, A. and Palmer, R. G. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City,CA, 1992.

[17] Freeman, J. A. and Skapura, D. M. *Neural Networks - Algorithms, Applications and Programming Techniques.* Addison-Wesley, Reading,MA, 1991.

[18] Editor - Kosko, B. *Neural Network for Signal Processing.* Prentice-Hall International, Englewood Cliffs, NJ, 1992.

[19] Wang, L. On Competitive Learning. *IEEE Transactions on Neural Networks, Vol. 8, No.5*, pages 1214–1218, September 1997.

[20] Karayiannis, N. B. and Pai, P.-I. Fuzzy Vector Quantization Algorithms and Their Applications in Image Compression. *IEEE Transactions on Image Processing, Vol. 4, No. 9*, pages 1193–1201, September 1995.

[21] Karayiannis, N. B. and Pai, P.-I. Fuzzy Algorithms for Learning Vector Quantization. *IEEE Transactions on Neural Networks, Vol. 7, No. 5*, pages 1196–1211, September 1996.

[22] Pan, J. S., McInnes, F. R. and Jack, M. A. VQ Codebook Design Using Genetic Algorithms. *Electronics Letters, Vol. 31, No. 17*, pages 1418–1419, 17th August 1995.

[23] Equitz, W. H. A New Vector Quantization Clustering Algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 37, No. 10*, pages 1568–1575, October 1989.

[24] Gray, R. M. and Karnin, E. D. Multiple Local Optima in Vector Quantizers. *IEEE Transactions on Information Theory, Vol. IT-28, No.2*, pages 256–261, Mar 1982.

[25] Chen, C.-Q., Koh, S.-N. and Sivaprakasapillai, P. Codebook Generation for Vector Quantisation. *Electronics Letters, Vol. 31, No. 7*, pages 522–523, 30th March 1995.

[26] Chen, C.-Q., Koh, S.-N. and Sivaprakasapillai, P. VQ Codebook Design Algorithm Based on Partial GLA. *Electronics Letters, Vol. 31, No. 21*, pages 1803–1806, 12th October 1995.

[27] Lee, D., Baek, S. and Sung K. Modified K-means Algorithm for Vector Quantizer Design. *IEEE Signal Processing Letters, Vol. 4, No.1*, pages 2–4, January 1997.

[28] Yair, E., Zeger, K. and Gersho, A. Competitive Learning and Soft Competition for Vector Quantizer Design. *IEEE Transactions on Signal Processing, Vol. 40, No.2*, pages 294–309, February 1992.

[29] Deller Jr., J. R., Proakis, J. G. and Hansen, J. H. L. *Discrete-time Processing of Speech Signals.* MacMillan Publishing Co., 1993.

[30] Aguiar Neto, B. G. *Processamento e Transmissão Digital de Voz (Apostila).* Curso de Pós Graduação em Engenharia Elétrica, Campina Grande - PB, 1995.

[31] Dimolitsas, S. Subjective Quality Quantification of Digital Voice Communication Systems. *IEE Proceedings-I, Vol. 138, No. 6*, pages 585–595, December 1991.

[32] Dimolitsas, S. Objective Speech Distortion Measures and Their Televance to Speech Quality Assesments. *IEE Proceedings, Vol. 136, Pt. I, No. 5*, pages 317–324, October 1989.

[33] Eskicioglu, A. M. and Fischer, P. S. Image Quality Measures and Their Performance. *IEEE Transactions on Communications, Vol.43, No. 12*, pages 2959–2965, December 1995.

[34] Gray, R. M. and Linde, Y. Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources. *IEEE Transactions on Communications, Vol. COM-30, No.2*, pages 381–389, February 1982.

[35] Vilar França, R. M. and Aguiar Neto, B. G. A Self-Organizing Algorithm for Waveform Vector Quantization. *Records of SBT/IEEE International Telecommunication Symposium*, pages 136–140, August,1994.

[36] Simpson, P. K. *Neural Networks Applications - IEEE Technology Update Series.* II series, Technical Activities Board, Inc., New York, 1996.

[37] Simpson, P. K. *Neural Networks Theory, Technology and Applications - IEEE Technology Update Series.* III series, Technical Activities Board, Inc., New York, 1996.

[38] Fehn, H. G. and Noll, P. Multipath Search Coding of Stationary Signals with Applications to Speech. *IEEE Transactions on Communications, Vol. COM-30, No.4*, pages 687–701, April 1982.

[39] Kaouri, H. A. and McCanny, J. V. Reduced-Memory Multirate Vector Quantisation. *Electronics Letters, Vol. 25, No. 7*, pages 471–473, 30th March 1989.

[40] Makur, A. and Subbalakshmi, K. P. Variable Dimension VQ Encoding and Codebook Design. *IEEE Transactions on Communications, Vol. 45, No.8*, pages 897–899, August 1997.

[41] Abut, H., Gray, R. M. and Rebolledo, G. Vector Quantization of Speech and Speech-Like Waveforms. *IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-30, No.3*, pages 423–435, June 1982.

[42] Gersho, A. and Cuperman, V. Vector Quantization: A Pattern-Matching Techinique for Speech Coding. *IEEE Communications Magazine*, pages 15–20, December 1983.

[43] Nandi, A. K., Aburdene, M. F., Constantinides, A. G. and Dologlou, I. Variation of Vector Quantisation and Speech Waveform Coding. *IEE Proceedings-I, Vol. 138, No.2*, pages 76–80, April 1991.

[44] Vilar França, R. M. and Aguiar Neto, B. G. Voice Waveform Vector Quantization Using a Competitive Algorithm. *"Records of the IEEE GLOBECOM'94"*, pages 872–875, November,1994.

[45] Vilar França, R. M. and Aguiar Neto, B. G. A Modified Kohonen's Algorithm for Voice Waveform Vector Quantization. *"Records of the XI Simpósio Brasileiro de Inteligencia Artificial"*, pages 351–366, October,1994.

[46] Vilar França, R. M. and Aguiar Neto, B. G. Designing Codebooks for Voice Waveform Vector Quantization Based on the Karhunen-Loève Transform. *"Records of the IEEE International Telecommunications Symposium ITS'96 "*, pages 44–48, October,1996.

[47] França, R. M., Madeiro, F. and Aguiar Neto, B. G. Building Voice Patterns Inside Vector Quantization Codebooks with a Modified Kohonen's Algorithm. *Proceedings of IEEE ICSP'97*, pages 291–295, August,1997.

[48] França, R. M. V., Madeiro, F. and Aguiar Neto, B. G. Transformada de Karhunen-Loève Aplicada à Compressão de Sinais de Voz". *Anais do XX CNMAC*, pages 508–509, Setembro,1997.

[49] Chen, J.-H. and Gersho, A. Gain-Adaptive Vector Quantization with Applications to Speech Coding. *IEEE Transactions on Communications, Vol. COM-35, No.9*, pages 918–930, September 1987.

[50] Diamantsaras, K. J. & Kung, J. S. *Principal Component Neural Networks - Theory and Applications.* John Wiley & Sons, New York - NY, 1996.

[51] Fukunaga, J. *Statistical Pattern Recognition.* Academic Press, New York, 1990.

[52] Xu, L. and Yuille, A. L. Robust Principal Componente Analysis by Self-Organizing Rules Based on Statistical Physics Approach. *IEEE Transactions on Neural Networks, Vol. 6, No. 1*, pages 131–143, January 1995.

[53] Meyer, Y. *Wavelets: Algorithms and Applications.* Society of Industrial and Applied Mathematics, Philadelphia, 1993.

[54] Parker, J. R. *Algorithms for Image Processing and Computer Vision.* John Wiley & Sons, Inc., New York, 1997.

[55] Young, R. K. *Wavelet Theory and its Applications.* Kluwer Academic Publishers, Massachusetts, 1995.

[56] Bruce, A., Donoho, D. and Gao, H.-Y. Wavelet Analysis. *IEEE Spectrum*, pages 26–35, October 1996.

[57] Kubrick, A. and Ellis, T. Classified Vector Quantisation of Images: Codebook Design Algorithm. *IEE Proceedings, Vol. 137, Pt. I, No. 6*, pages 379–386, December 1990.

[58] Shanbehzadeh, J. and Ogunbona, P. O. On the Computational Complexity of the LBG and PNN Algorithms. *IEEE Transactions on Image Processing, Vol. 6, No.4*, pages 614–616, April 1997.

[59] Goldberg, M. and Sun, H. Image Sequence Coding Using Vector Quantization. *IEEE Transactions on Communications, Vol. COM-34, No. 7*, pages 703–710, July 1986.

[49] Chen, J.-H. and Gersho, A. Gain-Adaptive Vector Quantization with Applications to Speech Coding. *IEEE Transactions on Communications, Vol. COM-35, No.9*, pages 918–930, September 1987.

[50] Diamantsaras, K. J. & Kung, J. S. *Principal Component Neural Networks - Theory and Applications.* John Wiley & Sons, New York - NY, 1996.

[51] Fukunaga, J. *Statistical Pattern Recognition.* Academic Press, New York, 1990.

[52] Xu, L. and Yuille, A. L. Robust Principal Componente Analysis by Self-Organizing Rules Based on Statistical Physics Approach. *IEEE Transactions on Neural Networks, Vol. 6, No. 1*, pages 131–143, January 1995.

[53] Meyer, Y. *Wavelets: Algorithms and Applications.* Society of Industrial and Applied Mathematics, Philadelphia, 1993.

[54] Parker, J. R. *Algorithms for Image Processing and Computer Vision.* John Wiley & Sons, Inc., New York, 1997.

[55] Young, R. K. *Wavelet Theory and its Applications.* Kluwer Academic Publishers, Massachusetts, 1995.

[56] Bruce, A., Donoho, D. and Gao, H.-Y. Wavelet Analysis. *IEEE Spectrum*, pages 26–35, October 1996.

[57] Kubrick, A. and Ellis, T. Classified Vector Quantisation of Images: Codebook Design Algorithm. *IEE Proceedings, Vol. 137, Pt. I, No. 6*, pages 379–386, December 1990.

[58] Shanbehzadeh, J. and Ogunbona, P. O. On the Computational Complexity of the LBG and PNN Algorithms. *IEEE Transactions on Image Processing, Vol. 6, No.4*, pages 614–616, April 1997.

[59] Goldberg, M. and Sun, H. Image Sequence Coding Using Vector Quantization. *IEEE Transactions on Communications, Vol. COM-34, No. 7*, pages 703–710, July 1986.

[60] Ramamurthi, B. and Gersho, A. Classified Vector Quantization of Images. *IEEE Transactions on Communications, Vol. COM-34, No. 11*, pages 1105–1115, November 1986.

[61] Nasrabadi, N. M. and King, R. A. Image Coding Using Vector Quantization: A Review. *IEEE Transactions on Communications, Vol. 36, No. 8*, pages 957–971, August 1988.

[62] Kim, D. S. and Lee, S. U. Image Vector Quantizer Based on a Classification in the DCT Domain. *IEEE Transactions on Communications, Vol. 39, No. 4*, pages 549–556, April 1991.

[63] Wu, Y. and Coll, D. C. BCT-VQ-DCT Hybrid Coding of Digital Images. *IEEE Transactions on Communications, Vol. 39, No. 9*, pages 1283–1287, September 1991.

[64] Galli, I., Mecocci, A. and Cappellini, V. Improved Colour Image Vector Quantisation by Means of Self-Organising Neural Networks. *Electronics Letters, Vol. 30. No. 4*, pages 333–334, 17th February 1994.

[65] Weingessel, A., Bischof, H., Hornik, K. and Leisch, F. Adaptive Combination of PCA and VQ Networks. *IEEE Transactions on Neural Networks, Vol. 8, No. 5*, pages 1208–1211, September 1997.

[66] De Natale, F. G. B., Fioravanti, S. and Giusto, D. D. DCVRQ: A New Strategy for Efficient Entropy Coding of Vector-Quantized Images. *IEEE Transactions on Communications, Vol. 44, No. 6*, pages 696–706, June 1996.

[67] Haykin, S. Neural Networks Expand SP's Horizons. *IEEE Signal Processing Magazine*, pages 24–49, March,1996.

[68] Tai, S. C., Lai, C. C. and Lin, Y. C. Two Fast Nearest Neighbor Searching Algorithms for Image Vector Quantization. *IEEE Transactions on Communications, Vol. 44, No. 12*, pages 1623–1628, December 1996.

[69] Lightstone, M. and Mitra, S. K. Image-Adaptative Vector Quantization in an Entropy-Constrained Framework. *IEEE Transactions on Image Processing, Vol. 6, No. 3*, pages 441–450, March 1997.

[70] Castro, D., Klautau, A. and Alcaim, A. Quantização Vetorial Preditiva Aplicada à Codificação de Imagens Decompostas Através de Transformada Wavelet. *Anais do XV Simpósio Brasileiro de Telecomunicações*, pages 599–603, 1997.

[71] Alcaim, A., Solewicz, J. A. and Moraes, J. A. Freqüência de Ocorrência dos Fones e Listas de Frases Foneticamente Balanceadas no Português Falado no Rio de Janeiro. *Revista da Sociedade Brasileira de Telecomunicações*, 1990.

[72] Kim, D. S., Kim, T. and Lee, S. U. On Testing Trained Vector Quantizer Codebooks. *IEEE Transactions on Image Processing, Vol. 6, No. 3*, pages 398–406, March 1997.