UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

AMAURI HOLANDA DE SOUZA JÚNIOR

AVALIAÇÃO DE REDES NEURAIS AUTO-ORGANIZÁVEIS PARA RECONHECIMENTO DE VOZ EM SISTEMAS EMBARCADOS

FORTALEZA
2009

AMAURI HOLANDA DE SOUZA JÚNIOR

AVALIAÇÃO DE REDES NEURAIS AUTO-ORGANIZÁVEIS PARA RECONHECIMENTO DE VOZ EM SISTEMAS EMBARCADOS

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Engenharia de Teleinformática, da Universidade Federal do Ceará, como parte dos requisitos exigidos para obtenção do grau de Mestre em Engenharia de Teleinformática.

Orientador: Prof. Dr. Guilherme de Alencar Barreto

FORTALEZA

2009

Dedico este trabalho aos meus pais

Amauri Holanda e Elita Rodrigues

pelo constante apoio, incentivo e admiração.

Agradecimentos

A Deus, acima de tudo.

Aos colegas de laboratório do GRAMA (Grupo Avançado de Aprendizagem de Máquinas) pelas discussões pertinentes e por propiciarem um ótimo ambiente de trabalho.

Ao Professor Guilherme de Alencar Barreto, pelos ensinamentos, incentivo, confiança, paciência e dedicação depositados ao longo deste trabalho.

Ao Professor Antonio Themoteo Varela, pela confiança depositada e por ter possibilitado o início desta jornada.

Aos professores e funcionários do Departamento de Engenharia de Teleinformática que de forma direta ou indireta participaram do desenvolvimento deste trabalho.

A minha amada Ariadne, pela compreensão, carinho e incentivo para que eu pudesse concluir esta importante etapa da minha vida.

À CAPES pelo suporte financeiro.

Resumo

Esta dissertação apresenta um amplo estudo comparativo do desempenho de arquiteturas de redes neurais auto-organizáveis em tarefas de reconhecimento de voz para sistemas embarcados. Em particular, avalia-se a viabilidade do uso das redes SOM (*self-organizing map*) e TS-SOM (*tree-structured SOM*) na tarefa de interesse. Os sinais de voz são parametrizados por coeficientes LPC e Cepstrais, e os testes executados envolvem reconhecimento de voz nos modos dependente e independente do locutor. O conjunto de dados utilizado neste trabalho consiste de 17 classes de elocuções, pronunciadas naturalmente, correspondendo aos dígitos e operações necessárias para o desenvolvimento de uma calculadora acionada pela voz. A aplicação é então embarcada em um smartphone N95 da Nokia. Os resultados indicam que as redes SOM e TS-SOM podem ser aplicadas na tarefa de interesse com sucesso, e apresentaram resultados similares ou superiores aos de outras técnicas clássicas quanto à taxa de reconhecimento e custo computacional.

Palavras-chave: Reconhecimento de Voz, Redes de Kohonen, Sistemas Embarcados.

Abstract

This dissertation presents a comprehensive performance comparison study of self-organizing neural network architectures applied to speech recognition in embedded systems. In particular, we assess the feasibility of using the self-organizing map (SOM) and the tree-structured self-organizing map (TS-SOM) in the task of interest. Speech signals are parameterized through LPC and Cepstral coefficients, and both, speaker dependent and speaker independent tests, are carried out. The speech data used in this work consisted of 17 classes of words, naturally pronounced, corresponding to digits and operators required to develop a voice-driven calculator. The application is then embedded into the N95 Nokia smartphone. The obtained results indicate that the SOM and TS-SOM networks can be successfully applied to the task of interest, since they perform similarly to or even better than classical speech recognition methods, concerning recognition rates and the computational costs.

Keywords: Speech Recognition, Self-Organizing Maps, Embedded Systems.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$H(z)$	*Função de transeferência do modelo AR*
$\widehat{a}(i)$	*i-ésimo coeficiente de predição linear*
$s(n)$	*Sinal de voz*
A, B e C	*Parâmetros relacionados ao limiares do algoritmo de recorte*
$CZ(m)$	*Taxa de cruzamento por zero do m-ésimo quadro de voz*
$E(m)$	*Energia do m-ésimo quadro de voz*
$G(z)$	*Função de transferência do filtro de pré-ênfase*
$\mathbf{R}_s$	*Matriz de Correlação*
P	*Ordem de predição linear*
M	*Número de neurônios nas redes SOM e TS-SOM*
$\mathbf{w}_i$	*Pesos do i-ésimo neurônio*
i^*	*Índice do neurônio vencedor*
$\mathbf{i}^*$	*Coordenada do neurônio i^* no mapa*
$h(i, i^*; t)$	*Função vizinhança da rede SOM*
E_Q	*Erro de quantização*
E_T	*Erro topológico*
N	*Número de vetores de treinamento*
η	*Taxa de aprendizagem*
ϑ	*Raio de atuação da função vizinhança na rede SOM*
ε	*Número de épocas de treinamento*
p	*Número de filhos da rede TS-SOM*
c	*Número de camadas da rede TS-SOM*
M_c	*Número de neurônios na última camada da rede TS-SOM*

Lista de Siglas

AR	*Auto-Regressivos*
ARMA	*Auto-Regressivos Médias Móveis*
DTW	*Dinamic Time Warping*
FIR	*Filtro de Resposta ao Impulso de Duração Finita*
HMM	*Hidden Markov Models*
JME	*Java Micro Edition*
LPC	*Linear Predictive Coding*
LTSD	*Long-Term Spectral Divergence*
LVQ	*Learning Vector Quantization*
MFCC	*Mel Frequency Cepstral Coefficients*
MLP	*MultiLayer Perceptron*
MSE	*Mean-Squared Error*
PCA	*Principal Components Analysis*
PDS	*Partial Distance Search*
RAV	*Reconhecimento automático da voz*
SOC	*Sum of Components*
SOM	*Self-Organizing Maps*
SOTT	*Self-Organizing Topological Tree*
TS-SOM	*Tree Structured Self Organizing Maps*
VA	*Variável Aleatória*
WTK	*Wireless ToolKit*

Sumário

Capítulo 1

Introdução

Nos últimos anos tem sido observada uma rápida evolução dos serviços e aplicações disponíveis para dispositivos móveis. Somente em 2007, o número de vendas de celulares ultrapassou 1 bilhão de unidades. Isto significa mais de 2,3 milhões de unidades vendidas por dia (MAREJKA, 2008).

Devido ao crescimento significativo da capacidade de processamento, os celulares atualmente são capazes de executar tarefas antes realizadas somente em computadores pessoais (ZAYKOVSKIY; SCHMITT, 2007). No entanto, a interface de comunicação com o usuário ainda limita a usabilidade dos dispositivos móveis. As formas convencionais de interação, como o teclado, são limitados pelo tamanho. Digitar nestes teclados é desconfortável e susceptível a erros. Um outro problema é que geralmente as pessoas precisam utilizar os celulares quando estão em trânsito e em momentos nos quais não se pode utilizar as mãos na interação com o dispositivo.

A forma mais natural para resolver tais problemas é utilizar uma interface de comunicação via voz. Falar não requer contato visual nem físico com o dispositivo. Além disso, requer pouco ou nenhum treinamento adicional para que o usuário opere o sistema. Um indício disto é que, na última década, esforços têm sido realizados para o desenvolvimento de sistemas de Reconhecimento Automático de Voz (RAV) em celulares, tais como os presentes nos trabalhos de Alhonen et al. (2007) e Olsen et al. (2008).

A reprodução direta dos algoritmos de RAV utilizados em computadores pessoais nem sempre é possível em sistemas embarcados. Devido à grande variabilidade de ambientes e recursos muito limitados, o desenvolvimento de sistemas RAV em dispositivos móveis necessita de técnicas especiais, principalmente relacionadas à otimização computacional de recursos.

Neste trabalho, será considerado um sistema embarcado qualquer

> "*...sistema computacional com propósitos específicos, normalmente construído em dimensões reduzidas, que deve funcionar de forma autônoma. Como exemplos, podem ser citados telefones celulares, equipamentos médicos especializados, alguns equipamentos de instrumentação, entre outros.*" (JUNG et al., 2005)

Projetar e implementar *software* para sistemas embarcados é diferente e, em geral, mais desafiador do que para *desktop*. Programas embarcados não só devem ser funcionais, mas também executar seguindo taxas de processamento bem especificadas, dentro de um limite de memória disponível e ainda sob restrições de consumo de potência.

Os sistemas de RAV possuem características distintas em função da tarefa a ser realizada. O conjunto destas características define o escopo para o sistema e suas possíveis limitações. Eles podem então ser caracterizados, principalmente, pelo

1. *Estilo da fala*: A elocução pode ser pronunciada de forma espontânea e por leitura. A forma espontânea é um pouco mais complexa de ser identificada devido a contrações fonéticas, que são comuns nesse estilo de fala.

2. *Tamanho do vocabulário*: Embora não existam limiares definitivos quanto ao tamanho do vocabulário, Rabiner & Juang (2008) estabelecem como vocabulário pequeno aqueles com cerca de 2 a 100 palavras; médio, de 100 a 1000; e grande, acima de 1000.

3. *Modo da fala*: Divide-se em palavras isoladas, palavras conectadas e fala contínua. Em aplicações de reconhecimento de palavras isoladas, as palavras são pronunciadas separadamente, muitas vezes seguindo um tempo pré-determinado entre as elocuções. A pronunciação de palavras conectadas exige um mecanismo automático para recorte das palavras na frase. No reconhecimento da fala contínua, o sistema deve ser capaz de identificar frases inteiras, conversação espontânea e, em geral, não há restrição quanto ao uso das palavras.

4. *Dependência do locutor*: Varia entre (*i*) sistemas dependentes do locutor, nos quais são reconhecidas, principalmente, elocuções pronunciadas por locutores previamente conhecidos, utilizados em uma etapa de treinamento; (*ii*) sistemas com adaptação ao locutor, que funcionam após uma fase de adaptação; e (*iii*) sistemas independentes do locutor, em que qualquer pessoa pode ser utilizada sem necessidade de treinamento adicional do sistema. É importante ressaltar que as arquiteturas gerais desses sistemas são as mes-

mas e o desempenho na tarefa de reconhecimento tende a ser tão pior quanto maior for a independência do locutor.

5. *Ambiente*: Sistemas RAV devem ser projetados em função do nível de ruído do ambiente em que deverão funcionar.

Além das possibilidades supracitadas, outras, tais como variações de transdutor, estado emocional do locutor, velocidade da fala, afetam consideravelmente os sistemas de reconhecimento.

Considerando a aplicação de reconhecimento de voz em dispositivos móveis, as aplicações mais comuns são

1. *Chamada por voz*: O usuário apenas pronuncia o nome de alguém da lista de telefones e então a chamada é iniciada. Uma outra forma desta aplicação é a pronunciação dos dígitos do telefone celular para quem se deseja ligar.

2. *Controle de Comandos*: Facilita a navegação pelas ferramentas disponíveis no celular. É uma forma mais fácil de ir direto à ação desejada, sem necessidade de vários passos na navegação.

3. *Fala para texto*: Também conhecida como ditado (*dictation*), esta aplicação é comum em computadores pessoais e seu desempenho tem sido continuamente melhorado. Um exemplo típico desta aplicação é o ditado de mensagens em celulares (OLSEN et al., 2008).

Dentre os problemas mencionados, talvez o principal no âmbito das aplicações para dispositivos móveis é a eliminação do ruído. De fato, o sistema deve ser capaz de selecionar adequadamente a elocução dentre vários tipos de ruído. E isto se torna mais crítico devido ao fato que celulares são usados nos mais diversos ambientes, tais como ônibus, *shopping centers* e até estádios de futebol.

Visto que há atualmente uma tendência de integração de serviços, tais como TV Digital, nos celulares, é de interesse o desenvolvimento de sistemas com baixo consumo de energia. Esta é uma das principais motivações para a pesquisa em sistemas embarcados, sendo fundamental a criação e comparação de algoritmos que possibilitem uma diminuição do número de operações aritméticas realizadas pelo celular.

1.1 Pesquisa em Reconhecimento de Voz

A pesquisa em reconhecimento de voz já existe há mais de setenta anos. Rabiner & Juang (2008) dividem esse período em quatro gerações e incluem uma quinta geração relativa aos temas de pesquisa atuais. Cada geração é caracterizada por tipos de classificadores e aplicações específicas.

A primeira geração ocorreu de 1930 a 1950 e consistia no reconhecimento de pequenos conjuntos de palavras isoladas através de métodos *ad-hoc*. Na segunda geração, década de 50, começou-se a utilizar características fonético-acústicas para reconhecimento de fonemas e palavras isoladas.

A terceira geração, que compreende o período de 1960 a 1980, iniciou a utilização de abordagens de reconhecimento de padrões na determinação de palavras conectadas para vocabulários de pequeno e médio porte. Nesta geração surgiu a utilização de métodos estatísticos para extração de características e algoritmos de programação dinâmica para alinhamento temporal dos padrões de voz. Além disso, foi neste período que se iniciou o uso de quantizadores vetoriais para redução da quantidade de padrões a serem processados.

A quarta geração, período compreendido de 1980 a 2000, caracterizou-se pela utilização de métodos estatísticos e neurais, tais como Modelos Ocultos de Markov (HMM, *Hidden Markov Model*) para modelagem da dinâmica da fala, K-Médias (MACQUEEN, 1967) e Redes Neurais Artificiais para classificação dos padrões de voz.

Na quinta geração, que vai do ano 2000 até os dias atuais, a pesquisa tem focado no reconhecimento da fala contínua utilizando combinação de HMM e características fonéticas-acústicas na detecção e correção de irregularidades linguísticas; reconhecimento de voz em ambientes ruidosos; combinação de abordagens neurais, como as redes SOM (*Self-Organizing Maps*) e MLP (*MultiLayer Perceptron*); entre outros.

A rede SOM[1] (*Self Organizing Map*), proposta por Kohonen (1982), vem sendo aplicada em diversas áreas, tais como reconhecimento de voz, robótica, mineração de dados, entre outras. O algoritmo SOM tem sido utilizado também no reconhecimento de fala contínua. Neste contexto, surgiu o Mapa Fonético-Acústico Auto-Organizável (KOHONEN, 1997), o qual provê um mapeamento com preservação topológica de fonemas de uma elocução. No entanto, em aplicações em que se dispõe de recursos computacionais bastante limitados, como em sistemas embarcados, o uso da rede SOM na forma que foi proposta inicialmente torna-se impraticável

[1] Neste trabalho, usam-se indistintamente os termos algoritmo SOM, rede SOM, rede de Kohonen e mapa auto-organizável de Kohonen.

(KOIKKALAINEN, 1994).

1.1.1 *Softwares* de Reconhecimento de Voz

Nesta seção são apresentados alguns dos principais *softwares* para reconhecimento de voz. Por se tratar de uma área de pesquisa ativa e com inúmeras aplicações nos mais diversos campos do conhecimento, é praticamente impossível listar todos os *softwares* de RAV disponíveis no mercado ou na academia.

Um dos primeiros sistemas de reconhecimento de voz comerciais foi o *Dragon System* (BAKER, 1975) da Universidade Carnegie-Mellon. Ele reconhecia palavras isoladas com taxa de acerto de 84%. Em seguida, Wilpon et al. (1982) propuseram um sistema para reconhecimento de palavras isoladas e independente do locutor com taxa de acerto entre 80% e 97%, em função dos testes que foram realizados.

Em meados da década de 80 foi desenvolvido, nos laboratórios da IBM, o *Tangora* (JELINEK, 1985), um sistema de ditado (vocabulário de 5000 palavras) dependente do locutor, que fornecia taxas de acerto em torno de 94% para a fala controlada, ou lida. O trabalho de Jelinek (1985) também foi um dos primeiros a apontar para as dificuldades do reconhecimento da fala espontânea.

Um dos *frameworks* para reconhecimento de voz mais conhecido é o *Sphinx* (LEE et al., 1989), produzido na Universidade Carnegie-Mellon e parceiras. Trata-se de um sistema de reconhecimento escrito em linguagem de programação Java. Atualmente, o *Sphinx* está na versão 4 e dá suporte ao reconhecimento de fala contínua, de palavras isoladas e de conectadas - de forma dependente e independente do locutor. Com relação ao sistema de reconhecimento independente do locutor para dígitos pronunciados isoladamente (em inglês), são reportadas taxas de acerto em torno de 99%.

Nos sistemas de RAV aplicados ao Português Brasileiro, foram alcançadas taxas de acerto de 93% (MARTINS, 1997) no reconhecimento de palavras isoladas utilizando a rede MLP e em torno de 95% (ANDREAO, 2001) no reconhecimento de palavras conectadas através do uso de HMM.

Ainda com relação à lingua portuguesa, Lima (2000) desenvolveu um sistema com taxa de acerto de aproximadamente 82% para reconhecimento de dígitos isolados utilizando o algoritmo MLP, e de 93% usando DTW (*Dynamic Time Warping*) com múltiplas referências.

De modo geral, não existem bases de dados da língua portuguesa amplamente aceitas e dis-

poníveis que permita a criação de um *benchmark* do processo de reconhecimento. Por este motivo, neste trabalho será utilizada uma base de dados própria, descrita em detalhes no Capítulo 6.

1.2 Objetivos

O objetivo geral desta dissertação, bem como seus objetivos específicos, são detalhados nas subseções a seguir.

1.2.1 Objetivo Geral

Esta dissertação tem por principal objetivo o desenvolvimento e análise de estratégias para utilização da rede auto-organizável de Kohonen para classificação de sinais da Fala/Voz, visando sua aplicação em sistemas embarcados. Dar-se-á ênfase ao estudo de técnicas de *software* para aceleração da rede SOM, na tarefa de reconhecimento de palavras isoladas e independentes do locutor.

A principal contribuição deste trabalho consiste na análise da viabilidade da aplicação da rede SOM e variantes no projeto de sistemas embarcados e, paralelamente, no reconhecimento de voz. Para avaliar esta tarefa, foi desenvolvida uma calculadora acionada pela voz.

1.2.2 Objetivos Específicos

O objetivo geral da dissertação apresentado anteriormente, por ser bastante amplo, dá margem ao surgimento de vários objetivos menores e mais específicos, a saber:

1. Implementar os principais métodos utilizados para extração de características e recorte do sinal de voz: Algoritmo de recorte de Rabiner, Coeficientes de Predição Linear e Coeficientes Cepstrais.

2. Implementar, comparar e descrever as abordagens utilizadas para aceleração do treinamento da rede SOM: Busca com Distância Parcial, Busca com Atalho, TS-SOM e uso de funções vizinhanças.

3. Analisar a complexidade computacional da rede SOM, bem como das técnicas propostas para a aceleração do seu algoritmo.

4. Implementar e comparar as principais medidas de avaliação de mapas auto-organizáveis: Erro de Quantização e Erro Topológico.

5. Realizar um estudo sistemático do desempenho de alguns dos principais algoritmos para reconhecimento de voz, traçando um comparativo entre o reconhecimento dependente e independente do locutor: DTW, MLP, K-Médias.

6. Verificar a viabilidade do uso de redes baseadas no algoritmo SOM na tarefa de reconhecimento de voz em sistemas embarcados.

7. Desenvolver um aplicativo para *smartphones* que consiste em uma calculadora acionada por voz, com o intuito de validar o estudo realizado ao longo deste trabalho.

1.3 Produção Científica

Ao longo do desenvolvimento desta dissertação os seguintes artigos científicos foram publicados:

1. **Amauri H. S. Júnior**, Guilherme A. Barreto e Antonio T. Varela (2009), "Avaliação de redes auto-organizáveis para reconhecimento de voz em sistemas embarcados", aceito para publicação no *IX Congresso Brasileiro de Redes Neurais/ Inteligência Computacional* (CBRN'2009), Ouro Preto-MG.

2. **Amauri H. S. Júnior**, Guilherme A. Barreto e Antonio T. Varela (2009), "Estudo Comparativo de Redes Auto-Organizáveis para Reconhecimento de Voz", aceito para publicação no *IX Simpósio Brasileiro de Automação Inteligente* (SBAI'2009), Brasília-DF.

1.4 Estrutura da Dissertação

O restante desta dissertação está organizada segundo os capítulos abaixo.

Capítulo 2 - Neste capítulo são apresentados os conceitos relacionados ao reconhecimento de voz, incluindo as principais abordagens utilizadas em cada etapa do sistema de reconhecimento.

Capítulo 3 - Neste capítulo a rede SOM é descrita em maior detalhe, bem como é feita sua avaliação computacional. Além disto, são estudadas as principais métricas para cálculo de similaridade em espaços vetoriais e algumas medidas de avaliação da rede.

Capítulo 4 - Neste capítulo são discutidas as principais técnicas para aceleração da rede SOM. Nesta discussão estão incluídas algumas técnicas simples que podem ser utilizadas para reduzir o número de computações no algoritmo de treinamento e teste da rede SOM.

Capítulo 5 - Neste capítulo são apresentados e avaliados os impactos das métricas para cálculo de similaridade e das técnicas de aceleração da rede SOM na tarefa de quantização vetorial.

Capítulo 6 - Neste capítulo é avaliado o desempenho da rede SOM e variantes no reconhecimento de palavras isoladas, em contraste com técnicas clássicas de reconhecimento de voz. As simulações realizadas neste capítulo são feitas fora do sistema embarcado.

Capítulo 7 - Este capítulo apresenta os resultados das simulações realizadas no sistema embarcado.

Capítulo 8 - Neste capítulo estão as conclusões do estudo realizado nesta dissertação e as propostas para trabalhos futuros.

Apêndice A - Neste Apêndice são descritos aspectos computacionais da implementação das normalizações em redes MLP.

Apêndice B - Neste Apêndice são mostradas as matrizes de confusão para dois dos classificadores avaliados: algoritmo de alinhamento temporal dinâmico e rede auto-organizável de Kohonen.

Apêndice C - Neste Apêndice é descrita a ferramenta WTK (*Wireless ToolKit*) desenvolvida pela *Sun Microsystems* para o suporte ao desenvolvimento de aplicações embarcadas utilizando a linguagem de programação Java.

Capítulo 2

Fundamentos do Reconhecimento de Voz

Neste capítulo são apresentadas as etapas básicas presentes nos sistemas de reconhecimento de voz. Basicamente, estes sistemas são divididos em três etapas: recorte ou detecção de extremos, extração de características e reconhecimento, como ilustrado na Figura 2.1.

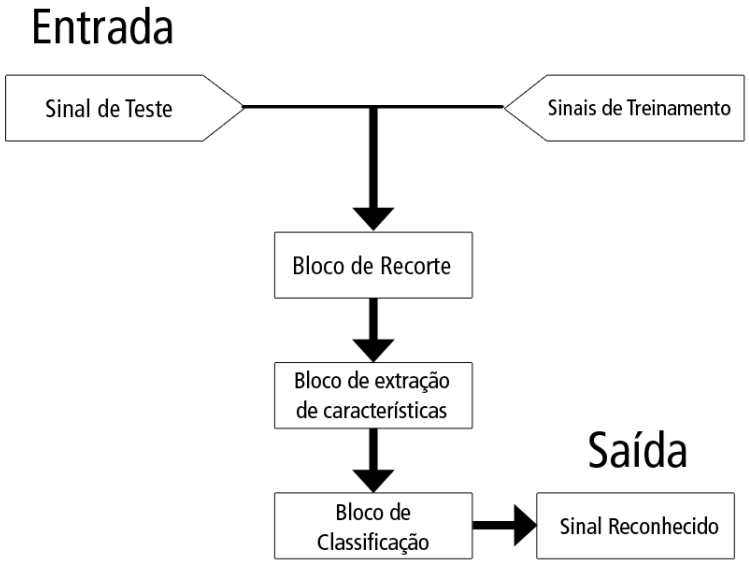


Figura 2.1: Estrutura básica dos sistemas de reconhecimento de voz.

A etapa de recorte é responsável por determinar o que é silêncio/ruído e o que de fato é informação de voz útil. Além disto, esta etapa reduz a quantidade de informação de voz que é passada para a fase de extração de características, diminuindo assim o custo computacional do sistema.

No bloco de extração de características são usados algoritmos que atuam sobre o sinal de voz a fim de representá-lo de forma mais compacta. Isto acontece pois o sinal de voz não é usado diretamente para alimentar o bloco de classificação por ser muito ruidoso, além de questões de custo de armazenamento, visto que um sinal de voz pode ter milhares de amostras.

No bloco de extração de características, o sinal de voz é representado de forma reduzida através da extração de atributos que permitam a diferenciação das elocuções.

A fase de reconhecimento pode ser dividida em treinamento e classificação. No treinamento, os vetores de características das elocuções são utilizados para determinar um modelo que represente cada classe de elocuções. A etapa de classificação usa o modelo gerado no treinamento para determinar qual elocução foi pronuncida. Em sistemas de reconhecimento independentes de locutor, as elocuções devem ser capturadas por um grande número de locutores, preferivelmente de diferentes idades, sotaques e gênero para que o sistema seja capaz de capturar variações entre locutores e então tornar-se mais abrangente.

Antes da descrição detalhada de cada etapa, é necessário uma visão geral sobre a modelagem da produção da fala.

2.1 Modelagem da Produção da Fala

Existem duas principais fontes de características da fala específicas aos locutores, as físicas e as adquiridas (ou aprendidas). As características físicas relacionam-se principalmente ao trato vocal, estrutura formada pelas cavidades que vão das pregas vocais até os lábios e o nariz. Na Figura 2.2 é ilustrado o conjunto de órgãos que formam o trato vocal e compõem o sistema de produção da fala.

Durante a produção da fala o ar vindo dos pulmões passa pelas cordas vocais, situadas na laringe, que são tensionadas e então vibram devido à passagem do fluxo de ar. O som produzido pelas pregas vocais ainda é fraco e possui poucos harmônicos, de modo que ele é então amplificado quando passa pelas chamadas cavidades de ressonância (laringe, faringe, boca e nariz), e ganha “forma” final quando é articulado através de movimentos de língua, lábios, mandíbula, dentes e palato.

À medida que as ondas sonoras passam pelas cavidades do trato vocal, o espectro da voz é alterado pelas ressonâncias do trato vocal. Estas ressonâncias formam picos de energia no espectro de frequência conhecidos como formantes. Com isso, pode-se então estimar a forma do trato vocal a partir da análise espectral da fala produzida.

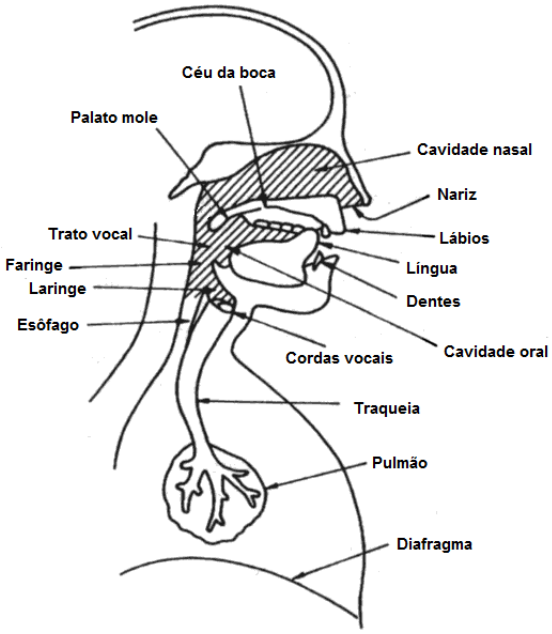
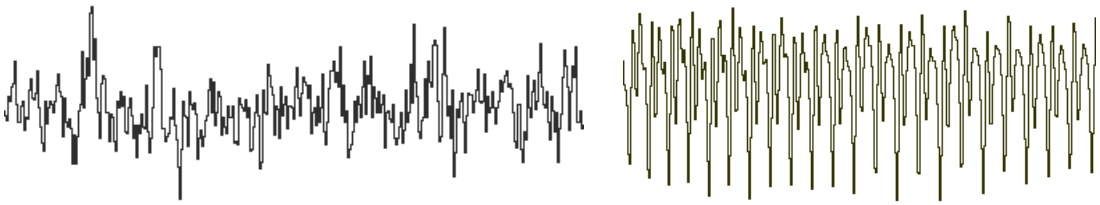


Figura 2.2: Sistema de produção da fala. (HENRIQUE, 2002)

Uma classificação comum dos eventos sonoros é quanto ao estado de vibração das cordas vocais. Adota-se uma convenção de três estados: silêncio, vozeados (sonoros) e não-vozeados (surdos). O silêncio representa a etapa em que nenhum som é produzido. Os sons ou fonemas sonoros são aqueles em que as cordas vocais são tensionadas e vibram de maneira aproximadamente periódica. Os sons surdos são produzidos quando não há vibração das cordas vocais, de modo que o som é formado basicamente nas cavidades do trato vocal, resultando em um sinal com natureza não-periódica ou aleatória. Na Figura 2.3 são ilustrados exemplos de sons vozeados e não-vozeados, em que pode-se observar na Figura 2.3(a) a natureza aleatória dos sons não-vozeados e, na Figura 2.3(b), a forma quasi-periódica de um fonema vozeado.



(a) Elocução do fonema /s/: Som não-vozeado.

(b) Fonema /z/: Som vozeado.

Figura 2.3: Exemplo de eventos sonoros vozeados e não-vozeados.

Existem ainda diversas outras caracterizações de eventos sonoros, como quanto ao modo e ponto de articulação, que fogem ao escopo deste trabalho, mas são apresentadas em Rabiner & Juang (1993) e Deller et al. (2000).

O sistema de produção da fala pode ser modelado em analogia com filtros digitais. Neste contexto, a fala é representada como resultante de um sinal de excitação, $u(n)$, convolvido com um filtro linear, $h(n)$. Esta excitação é representada por um "trem periódico" de impulsos no caso de trechos vozeados, e ruído branco nos trechos não-vozeados.

Conforme Deller et al. (2000), Rabiner (1978), durante um período de tempo no qual a voz pode ser considerada um sinal estacionário, o filtro linear pode ser caracterizado com boa aproximação por um modelo ARMA (*AutoRegressive Moving Average*), ou seja, um modelo constituído de pólos e zeros, cuja função de transferência no domínio z é dada por

$$H(z) = G\frac{1+\sum_{i=1}^{R} b(i)z^{-i}}{1-\sum_{j=1}^{P} a(j)z^{-j}}, \tag{2.1}$$

em que $a(j)$ e $b(i)$ são os coeficientes de autoregressão e de médias móveis, respectivamente. Os valores P e R representam as ordens dos modelos auto-regressivo e média móvel, e G é um fator de ganho associado à excitação $u(n)$.

2.2 Detecção de Extremos

A principal fonte de perturbações é o ruído do ambiente, causado por ar-condicionado, vozes de terceiros, sirenes e buzinas, entre outros. Perturbações tais como o sopro e o estalo da língua e dos lábio são também comuns aos sistemas de reconhecimento. O estalar da língua e dos lábios ocorre principalmente no início ou fim da elocução. O sopro consiste na respiração mais forte, e geralmente encontra-se ao término da elocução.

Detecção de extremos (*endpoints*) consiste em discriminar eventos sonoros de interesse, úteis ao sistema de reconhecimento, de perturbações do ambiente. Uma detecção errada pode degradar consideravelmente o desempenho do classificador (RABINER; JUANG, 1993).

Segmentação de palavras conectadas e detecção de extremos por si só abrangem toda uma área de pesquisa. Diversas técnicas têm sido utilizadas, entre elas pode-se citar as que utilizam energia e taxa de cruzamento por zero (RABINER; SAMBUR, 1975), (LAMEL et al., 1981), (TANYER; OZER, 2000); características espectrais (RAMíREZ et al., 2004); filtros estatísticos sobre medidas de energia (RAMíREZ et al., 2005). Um outro método rápido que se adequa para utilização em um sistema embarcado é a baseada em energia descrita em Wang et al. (2008). Uma avaliação dos principais métodos de detecção de extremos pode ser encontrada

em Tuononen et al. (2008).

Neste trabalho, será utilizada uma abordagem baseada no trabalho de Rabiner & Sambur (1975), devido a sua simplicidade, relativo baixo custo computacional e fácil compreensão. Embora grande parte dos algoritmos de recorte sejam aplicados antes da etapa extração de características, existem abordagens que se utilizam desta etapa para identificar silêncio e voz, e estão presentes em trabalhos como o de Haigh & Mason (1993) e Martin et al. (2001).

A idéia subjacente aos algoritmos que utilizam energia e cruzamento por zero consiste na diferença de energia e frequência entre a fala e o período de silêncio. A energia por quadro de um sinal de voz é maior no instante dos fonemas pronunciados do que durante o silêncio. Assim, a energia é utilizada como primeira estimativa para início e fim da elocução. Em seguida, a taxa de cruzamento por zero é usada para estender os limiares da palavra, no caso em que fonemas com baixas energias, mas altas frequências ocorrem no início ou fim da palavra pronunciada. Antes de iniciar o estudo sobre o algoritmo utilizado neste trabalho, é necessário definir as medidas de energia e taxa de cruzamento por zero.

2.2.1 Energia e Taxa de cruzamento por zero

A energia de um sinal é um conceito amplamente utilizado em diversas áreas de Processamento de Sinais. Seja $\{s(n)\}_{n=1}^{N}$ um sinal de voz de tempo discreto, sua energia é dada por

$$E(m) = \sum_{n=m-L+1}^{m} s^2(n). \tag{2.2}$$

A taxa de cruzamento por zero, $ZC(m)$, de um segmento do sinal de voz $s(n)$ representa a quantidade de vezes que o sinal cruza a mediana na escala da amplitude, normalizada pelo tamanho do quadro, L, e pode ser representada matematicamente como

$$ZC(m) = \frac{1}{L} \sum_{n=m-L+1}^{m} \frac{|sgn\{s(n)\} - sgn\{s(n-1)\}|}{2}. \tag{2.3}$$

em que m representa um índice associado a um segmento, ou quadro, do sinal de voz, e

$$sgn\{s(n)\} = \begin{cases} +1, & s(n) > 0, \\ -1, & c.c. \end{cases} \tag{2.4}$$

Se a escala de amplitude do sinal estiver simetricamente distribuída em relação à amplitude nula, então a taxa de cruzamento por zero pode ser implementada em *software* simplesmente através da multiplicação de uma amostra pela sua anterior. Caso o resultado forneça um valor

negativo, então o sinal cruzou a amplitude nula.

2.2.2 Algoritmo para Determinação de Extremos

O algoritmo de Rabiner & Sambur (1975) usa estatísticas de primeira e segunda ordem para determinação de extremos, são elas. Estas estatísticas são extraídas dos primeiros 100 a 200 ms do sinal, trecho em que se assume que o sinal consiste somente de ruído, e servem para calcular três limiares definidos a seguir.

- Limiar superior de energia : $\mu_e + A\sigma_e^2$
- Limiar inferior de energia : $\mu_e + B\sigma_e^2$
- Limiar da taxa de cruzamento: $\mu_c + C\sigma_c^2$

em que $A > B$, μ_e é a média da energia, σ_e^2 é a variância da energia, μ_c e σ_c^2 a média e a variância da taxa de cruzamentos por zero, respectivamente. Os valores de A, B e C são definidos heuristicamente e podem ser adaptados conforme a necessidade do sistema de recorte. É importante que o valor de A seja maior que o de B, pois o limiar superior de energia é a estimativa mais "grosseira" do ponto de início e fim da palavra pronunciada.

O algoritmo pode ser dividido em duas etapas. A primeira etapa do algoritmo consiste na segmentação do sinal de voz, em janelas (sem superposição) com duração entre 10 e 20 ms. Para cada segmento m, calculam-se os valores de energia $E(m)$ e taxa de cruzamento por zero $ZC(m)$. A partir destes valores pertencentes aos 10 a 20 primeiros segmentos, são extraídas as estatísticas de média e variância que são utilizadas para calcular os limiares superior e inferior de energia, e da taxa de cruzamento por zero.

A segunda e mais importante etapa, subdivide-se em três iterações. Na primeira iteração percorre-se em direção ao centro do sinal a partir dos extremos em busca de segmentos com valores de energia maiores que o limiar superior de energia. Caso encontrado, o início deste segmento é a primeira estimativa do início ou fim da palavra. Na segunda iteração, percorre-se a partir dos pontos encontrados na iteração anterior aos extremos. Se for encontrado um valor de energia menor que o limiar inferior, é marcado esse ponto e inicia-se a próxima iteração. A terceira iteração busca por taxas de cruzamento nos 20 segmentos subsequentes (em direção aos extremos) com valores maiores que o limiar da taxa de cruzamento. Caso existam três ou mais segmentos que excedam este limiar, então marca-se como segmento de recorte o último que ultrapassou o limiar. Do contrário, os pontos de recorte são os valores encontrados após a segunda iteração.

2.3 Extração de Características

A etapa de extração de características consiste na utilização de técnicas de transformação do sinal original em uma representação matemática que permita a identificação de uma dada elocução. Desse modo, o sinal é geralmente representado por um conjunto de vetores de características que podem então ser utilizados com mais eficiência na etapa de reconhecimento.

Essa etapa deve proporcionar uma redução no espaço de dados para análise sem perda significativa de informação útil. Nas subseções a seguir serão mostradas as etapas envolvidas na extração de características.

2.3.1 Pré-Processamento

Para o reconhecimento da fala, o primeiro passo é a conversão do sinal analógico em digital. A aquisição é feita através de um transdutor que, em geral, é um microfone. A conversão analógico-digital inclui tarefas de codificação e amostragem. Na codificação, utiliza-se, geralmente, 8 ou 16 bits para representar digitalmente uma amostra do sinal capturado. A amostragem é efetuada entre 8k - 44k Hertz, satisfazendo o teorema de Nyquist (OPPENHEIM et al., 1996). Em muitos sistemas é comum a aplicação de um filtro passa-baixas para limitar a banda de frequência do sinal. Com isto, pode-se eliminar o fenômeno conhecido como *aliasing*.

Observa-se que, para sinais de voz, a energia presente nas altas freqüências é pequena quando comparada com as baixas. A pré-ênfase é então necessária para que sejam obtidas amplitudes mais homogêneas das freqüências dos formantes, pois informações importantes sobre a elocução também podem estar presentes nas altas freqüências. Isto pode ser feito através de um filtro digital FIR (*Finite Impulse Response*) de primeira ordem, cuja função de transferência no domínio z é

$$G(z) = 1 - \alpha z^{-1}, \tag{2.5}$$

sendo α o parâmetro responsável pela pré-ênfase. Um valor bastante utilizado para α é 0,95 (MAFRA, 2002).

2.3.2 Análise de Curta Duração

O sinal de voz é não-estacionário e ruidoso, de modo que a analogia com filtros digitais estabelecida na Seção 2.1 só é válida para um período de tempo aproximadamente estacionário da fala, que geralmente é em torno de 10 a 30 ms. Dessa forma, todos os métodos de extração de características utilizados neste trabalho, utilizam a chamada análise de curta duração (*Short-*

Term Analysis).

Para esse fim, define-se uma janela (10-30 ms) que é movida ao longo do sinal de voz, com ou sem superposicionamento entre quadros[1] adjacentes. Existem diversas formas de implementar o janelamento do sinal, uma das janelas mais comuns é a de Hamming (OPPENHEIM et al., 1996), definida como

$$w(n) = 0,54 - 0,46\cos\left(\frac{2n\pi}{N_w - 1}\right), \tag{2.6}$$

em que N_w corresponde ao tamanho da janela. Então, a partir de cada quadro são extraídos os atributos do sinal de voz. Um esquema geral da análise de curta duração é ilustrada na Figura 2.4.

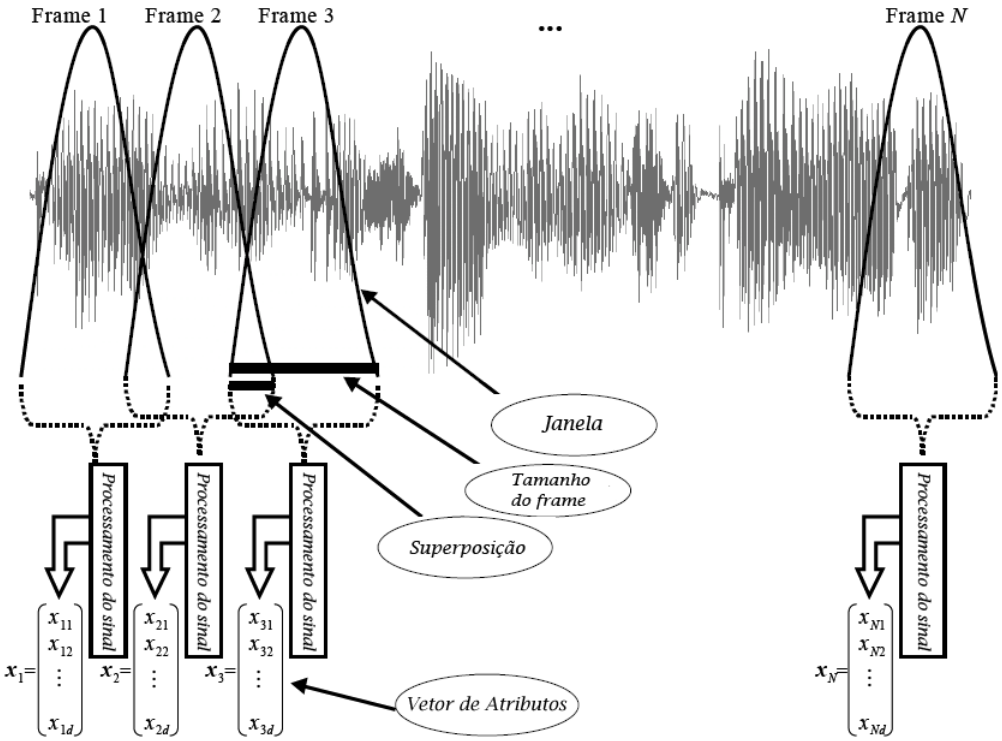


Figura 2.4: Visão geral da abordagem de análise de curta duração.

2.3.3 Codificação Linear Preditiva

Codificação Linear Preditiva (LPC, *Linear Predictive Coding*) é uma técnica de parametrização amplamente utilizada no reconhecimento de voz e locutor, pois permite uma representação do sinal por um número relativamente pequeno de parâmetros (KARPOV, 2003).

[1] Parte do sinal compreendida em uma janela.

Apesar da modelagem da fala ser representada inicialmente como um sistema ARMA, conforme discutido na Seção 2.1, as técnicas de extração de características em sistemas de RAV, utilizam, em geral, um modelo mais simples. Este, $H(z)$, corresponde a um filtro AR (*AutoRegressive*), isto é, um filtro constituído somente de pólos (*all-pole*) e cuja função de transferência é dada por

$$H(z) = \frac{1}{1 - \sum_{i=1}^{P} a(i) z^{-i}}. \tag{2.7}$$

em que P é a ordem do preditor e $a(i)$ são os coeficientes de autoregressão. A vantagem óbvia de utilizar modelos AR é a simplificação analítica do problema. Dessa forma, podem-se desenvolver métodos mais eficientes e simples para estimar os parâmetros do modelo. Além disto, na prática, zeros estão relacionadas à fase, que não tem efeito considerável na percepção. Se houver interesse somente em codificar e armazenar a fala, um modelo AR, que leva em conta somente o espectro de magnitude, não o de fase, é completamente aceitável e útil (DELLER et al., 2000).

A função de transferência do modelo AR descrito na Equação (2.7) representa o sinal de voz, $s(n)$, através da combinação linear de seus P valores passados juntamente com a excitação $u(n)$:

$$s(n) = \sum_{i=1}^{P} a(i) s(n-i) + G u(n), \tag{2.8}$$

em que $s(n)$ é o saída do modelo, $a(i)$ são os coeficientes de predição linear e P é a ordem do preditor. Visto que nas aplicações de processamento de voz o termo $u(n)$, que representa a excitação glotal e entrada do sistema, é desconhecido, geralmente este é ignorado (CAMPBELL JR, 1997), (MAKHOUL, 1975). Vale ressaltar que o termo "desconhecido" significa que não se tem acesso ao valor de $u(n)$ pois tudo que se pode medir diretamente são as amostras do sinal da voz, embora estatísticas da excitação sejam conhecidas.

A análise LPC parte do pressuposto que o sinal de voz pode ser representado pela combinação linear das amostras atrasadas. Dessa forma, o novo valor estimado do sinal $s(n)$ é dado por

$$\widehat{s}(n) = \sum_{i=1}^{P} \widehat{a}(i) s(n-i). \tag{2.9}$$

É importante lembrar que a modelagem acima é aplicada na análise de curta duração. Assim, o sinal de voz deve ser janelado, formando quadros, com ou sem superposição entre eles. A partir disso, são computados coeficientes LPC para cada quadro do sinal de voz. Na prática, utilizam-se ordens de predição que variam, em geral, de 10 a 20 coeficientes. Mais detalhes

sobre a escolha da ordem do preditor podem ser encontrados em Huang et al. (2001).

Em suma, o problema básico na análise LPC é determinar os coeficientes de predição, $\widehat{a}(i)$. Os métodos mais usados para isso são os de Yule-Walker, Burg e Mínimos Quadrados. Deller et al. (2000) resumem a complexidade computacional destes algoritmos.

Estimação pelo Método de Yule-Walker

Uma das técnicas utilizadas para encontrar os parâmetros $a(i)$ do modelo, é conhecida como método de Yule-Walker. Ele minimiza o erro quadrático médio (MSE - *Mean Squared Error*) entre o valor desejado, $s(n)$, e o valor estimado $\widehat{s}(n)$. A função custo $J(n)$ é então definida como

$$J(n) = E[(s(n) - \widehat{s}(n))^2] = E[(s(n) - \sum_{i=1}^{P} \widehat{a}(i)s(n-i))^2]. \tag{2.10}$$

Sabendo que a função custo é uma forma quadrática, o ponto de mínimo global é dado pelo ponto em que o vetor gradiente da função é igual ao vetor nulo, ou seja,

$$\nabla \mathbf{J}(n) = \mathbf{0}. \tag{2.11}$$

Resolvendo a Equação (2.11) obtêm-se o valor ótimo segundo o critério MSE, escrito na forma matricial:

$$\widehat{\mathbf{a}} = \mathbf{R}_s^{-1}\mathbf{r}_s, \tag{2.12}$$

em que

$$\mathbf{r}_s = \begin{pmatrix} r_s(1) \\ r_s(2) \\ \vdots \\ r_s(P) \end{pmatrix}, \quad \widehat{\mathbf{a}} = \begin{pmatrix} \widehat{a}(1) \\ \widehat{a}(2) \\ \vdots \\ \widehat{a}(P) \end{pmatrix}, \quad \mathbf{R}_s = \begin{pmatrix} r_s(0) & r_s(1) & \cdots & r_s(P-1) \\ r_s(1) & r_s(0) & \cdots & r_s(P-2)) \\ \vdots & \vdots & \vdots & \vdots \\ r_s(P-1) & r_s(P-2) & \cdots & r_s(0) \end{pmatrix}, \tag{2.13}$$

e $r_s(i) = E[s(n)s(n-i)]$ é a autocorrelação do sinal da voz com atraso i. A Equação (2.12) é conhecida como equação de Yule-Walker.

O problema da inversão da matriz $\mathbf{R}_s$ é simplificado devido a matriz de autocorrelação ser Toeplitz[2]. O algoritmo recursivo de Durbin-Levinson é uma das formas mais eficientes de resolver esse tipo de sistema de equações (MAKHOUL, 1975). Uma descrição detalhada do algoritmo pode ser encontrada em Haykin (1991).

[2]Matriz em que cada diagonal descendente da esquerda para a direita possui o mesmo valor.

2.3.4 Coeficientes Cepstrais

Seguindo o modelo de produção de uma elocução adotado nesta dissertação a fala é composta de uma excitação (ar vindo dos pulmões) convolvida com a resposta impulsiva do trato vocal. Esta relação pode ser representada por

$$s(n) = u(n) * h(n), \tag{2.14}$$

em que $s(n)$ é o sinal de voz, $u(n)$ é a excitação e $h(n)$ é a resposta impulsiva do trato vocal. Em geral, a separação dessas duas fontes é uma tarefa difícil (DELLER et al., 2000), uma vez que a representação no domínio da freqüência ainda não seria suficiente para separar as duas componentes pois nele as duas fontes são multiplicadas: $S(\omega) = U(\omega)H(\omega)$. Nesse contexto, surgiram os coeficientes cepstrais ou cepstrum.

O cepstrum é definido como a Transformada Inversa Discreta de Fourier (IDFT - *Inverse Discrete Fourier Transform*) do logaritmo da amplitude espectral de um sinal (SHAFER, 2008), (FURUI, 2001), matematicamente representado por

$$c_s(k) = \mathscr{F}^{-1}\{\log|\mathscr{F}\{\widehat{s}(n)\}|\}, \tag{2.15}$$

em que $\mathscr{F}^{-1}$ representa a tranformada inversa de Fourier. Assim, uma vez que

$$\log|S(\omega)| = \log|U(\omega)| + \log|H(\omega)|, \tag{2.16}$$

o *cepstrum* do sinal da voz pode ser expresso por

$$c_s(k) = c_u(k) + c_h(k), \tag{2.17}$$

ou seja, no domínio cepstral as componentes da excitação e do trato vocal são somadas. A variável independente k do *cepstrum* é chamada de *quefrency*, que corresponde a um parâmetro no domínio do tempo pois é formado a partir de uma IDFT.

A amplitude do espectro do sinal de voz é constituída pela combinação do invólucro espectral (*spectral envelope*) que está associado ao filtro linear e da estrutura fina (*fine structure*), em correspondência com a excitação glotal (FURUI, 2001), conforme mostrado na Figura 2.5. Conforme visto, a propriedade básica do cepstrum é que ele permite uma separação entre o envólucro espectral e a estrutura fina.

No domínio cepstral, a componente do trato vocal situa-se em baixas *quefrencias*, e a da excitação nas altas, conforme mostrado na Figura 2.6. Com isso, na separação das fontes pode-se utilizar um filtro (*liftro*, na nomenclatura do domínio cepstral) nas baixas *quefrencias*. Uma

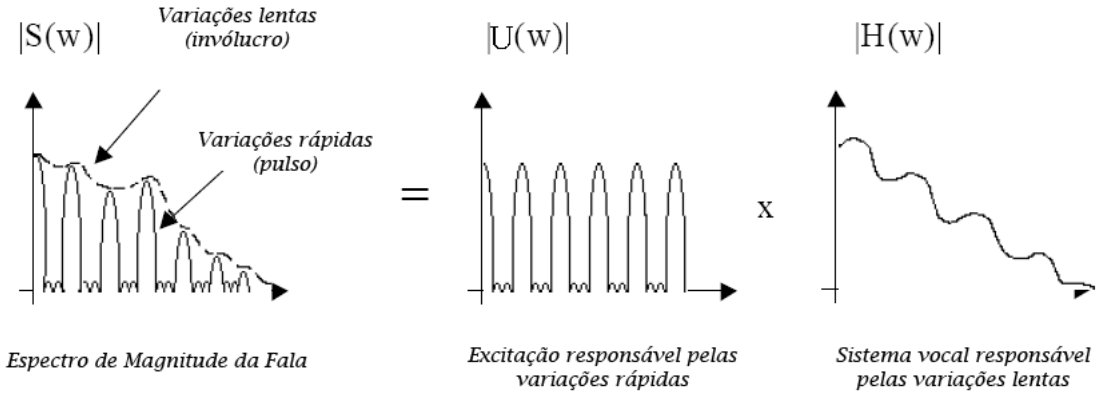


Figura 2.5: Representação dos componentes da fala no espectro de amplitude.

discussão mais profunda sobre este processo pode ser encontrada em Deller et al. (2000), Shafer (2008).

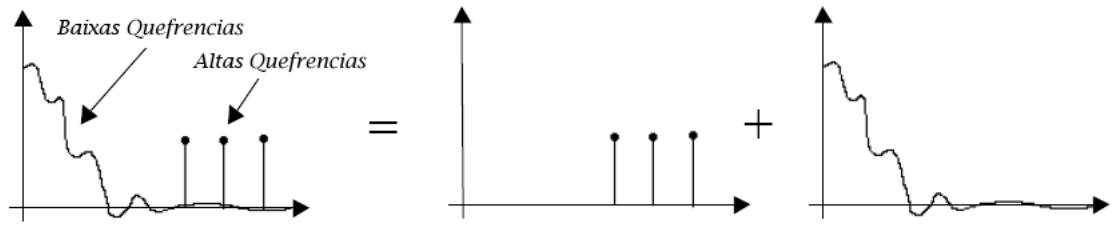


Figura 2.6: Representação dos componentes da fala no domínio cepstral.

Os coeficientes cepstrais podem também ser recursivamente estimados a partir dos coeficientes LPC. A esta abordagem dá-se o nome de LPC Cepstrum ou LPCC (*Linear Predictive Cepstral Coefficients*), sendo os coeficientes calculados como (DELLER et al., 2000)

$$\widehat{c}_h(k) = \begin{cases} \log G, & k = 0, \\ \widehat{a}(k) + \sum_{i=1}^{P-1} (i/k)\widehat{c}_h(i)\widehat{a}(k-i), & k = 1,...,P, \end{cases} \tag{2.18}$$

em que o ganho G do modelo AR pode ser estimado por $G = r_s(0) - \sum_{i=1}^{P} \widehat{a}(i) r_s(i)$ (RABINER, 1978).

Vale ressaltar que nos LPCC utilizam-se os coeficientes cepstrais da resposta ao impulso do modelo LPC. Portanto, os cepstrum estão sendo aplicados a uma seqüência na qual já se foi eliminada a excitação glotal, mencionada no início da seção. A razão para isso é que se torna

possível melhorar ainda mais a operação de separação da excitação através de manipulação desses coeficientes. Outra razão é que medidas de distâncias mais eficientes podem ser usadas. Além disso, os *cepstra* têm se mostrado uma representação eficiente do modelo da fala, e alcançado bons resultados tanto no reconhecimento do locutor quanto da fala (DELLER et al., 2000).

Diversas outras abordagens para extração de características têm sido utilizadas, entre elas, Combinação de LPC e MFCC (DELLER et al., 2000); LPC-MEL (TOKUDA et al., 1994); *Perceptual Linear Prediction Ceptrum Coefficients* (HUANG et al., 2001); Delta Cepstrum (DELLER et al., 2000), Freqüências LSP (*Line Spectrum Pairs*) (SOONG; JUANG, 1984), (GRASSI et al., 1997). Uma avaliação destas abordagens pode ser encontrada em Lima (2000).

2.4 Abordagens para Reconhecimento de Voz

Conforme visto na Seção 1.1, existem diversas abordagens e algoritmos para reconhecimento de voz. Nesta seção, será apresentada uma descrição mais detalhada de algumas delas, em particular as utilizadas neste trabalho.

2.4.1 Redes Neurais Supervisionadas

Uma forma muito comum de reconhecimento de voz consiste na utilização de redes neurais com aprendizagem supervisionada. Dentre elas, a mais amplamente utilizada é a rede neural MLP. Esta possui complexidade computacional, para cada padrão de treinamento, dada por $O(W)$, em que W representa o número de pesos sinápticos da rede. Uma discussão em detalhes do algoritmo foge do escopo deste trabalho, podendo ser encontrada em PRINCIPE et al. (2000) e HAYKIN (1994).

O principal problema na utilização de redes neurais supervisionadas está no algoritmo de normalização da elocução, necessário para fixar o número de atributos que serão apresentados ao classificador. Este problema torna-se mais evidente quando se refere ao reconhecimento de palavras pronunciadas espontâneamente. Ou seja, uma mesma palavra pronunciada por pessoas diferentes pode variar consideravelmente sua duração.

As formas mais básicas para efetuar essa normalização utilizam a limitação do número de *frames* que serão utilizados para o classificador. Assume-se a seguinte notação: ς denota a superposição entre *frames*, ϕ denota o tamanho da janela, τ define o espaçamento entre *frames* ($\phi - \varsigma$), ι é tamanho do sinal e, finalmente, ρ representa o número de *frames*. Com isso, as três

abordagens principais são descritas a seguir.

1. *Janela e superposição variáveis*: A cada nova palavra é calculado um tamanho de quadro ou janela variável, fixando-se o espaçamento em

$$\phi = \iota - (\rho - 1)\tau. \tag{2.19}$$

2. *Janela e espaçamento variáveis*: Nesta abordagem, define-se a superposição e o tamanho do quadro, de tal forma que o espaçamento é calculado por

$$\phi = \frac{\iota - \varsigma(\rho - 1)}{\rho}. \tag{2.20}$$

3. *Janela Fixa, Superposição e Espaçamento Variáveis*: Neste caso, a janela é fixada, e a superposição, bem como o espaçamento, é calculada dinâmicamente para cada palavra:

$$\varsigma = \frac{\iota - \phi\rho}{\rho - 1}. \tag{2.21}$$

Outra forma bastante comum de normalizar o sinal é utilizar um quantizador vetorial, e então usar os vetores código do quantizador com entrada da rede. O problema desta abordagem está na perda da informação temporal da elocução.

2.4.2 Alinhamento Temporal Dinâmico

A abordagem de alinhamento temporal dinâmico (DTW - *Dynamic Time Warping*) (SAKOE; CHIBA, 1978) compreende as técnicas de "encaixe no molde" (*template matching*) que utilizam princípios de programação dinâmica para determinação de uma medida de dissimilaridade entre padrões da fala de referência e teste.

A utilização do termo alinhamento se refere ao fato de que elocuções de uma mesma palavra, em geral, não possuem durações iguais. Assim, faz-se necessário a realização de uma normalização das flutuações temporais das elocuções para permitir a comparação entre elas.

Uma solução para esse problema é a normalização linear. Nela a dissimilaridade entre duas elocuções $\chi = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_I)$ e $\gamma = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_J)$, em que $\mathbf{x}_i \in \Re^P$ é o i-ésimo vetor de características da elocução χ, é definida por

$$d(\chi, \gamma) = \sum_{i=1}^{I} d(\mathbf{x}_i, \mathbf{y}_j), \tag{2.22}$$

em que $j = \frac{J}{I} i$ e $d(\mathbf{x}_i, \mathbf{y}_j)$ uma medida de dissimilaridade entre $\mathbf{x}_i$ e $\mathbf{y}_j$, tal como a distância

euclidiana. O somatório da Equação (2.22) pode ser feito em j, dependendo da direção desejada da normalização temporal.

A normalização linear implicitamente assume que a variação da taxa da fala é proporcional à duração da elocução e é independente do som pronunciado. Ou seja, pressupõe-se que fonemas dentro das elocuções também terão a mesma duração relativa. Dessa forma é necessário uma abordagem mais geral para normalização, e para isso serão explicados inicialmente os princípios de programação dinâmica que determinam o funcionamento do algoritmo DTW. Em seguida, será exemplificado o uso desta abordagem para reconhecimento de palavras isoladas, bem como as estratégias para geração de referências para formação de dicionários (*codebooks*).

Programação Dinâmica

Embora Programação Dinâmica (PD) seja utilizada na maioria das abordagens para reconhecimento de voz, sua origem refere-se à determinação de soluções ótimas em otimização combinatória no campo da matemática (MARQUES, 2005). Na comunidade científica de processamento de voz o trabalho que despertou maior interesse foi o de Sakoe & Chiba (1978).

Considere o plano cartesiano mostrado na Figura 2.7. Os pontos discretos, ou nós, de interesse são indexados por número inteiros. No contexto de reconhecimento de voz esses pontos, como (i, j), indicam uma medida de distância entre $\mathbf{x}_i$ (o i-ésimo vetor de características da elocução χ) e $\mathbf{y}_j$ (o j-ésimo vetor da elocução γ). O problema básico é encontrar o "melhor" caminho, ou com menor custo, iniciado em $(0,0)$ e finalizado em (I,J). Neste trabalho, um caminho do ponto (s,t) para (u,v) é representado pela sequência de pares ordenados: $(s,t),(i_1,j_1),(i_2,j_2),...,(u,v)$.

Como métrica para cálculo de custos associados aos caminhos é utilizado um custo de transição $d_T[(i_k,j_k)|(i_{k-1},j_{k-1})]$ associado ao(s) caminho(s) para ir de (i_{k-1},j_{k-1}) até (i_k,j_k), e um custo relacionado ao próprio nó, (i,j), dado por $d_N(i,j)$. Com estas definições, uma das medidas comumente utilizadas para ponderações de caminhos é dada pela combinação multiplicativa do custo de transição e o custo do nó:

$$d[(i_k,j_k)|(i_{k-1},j_{k-1})] = d_T[(i_k,j_k)|(i_{k-1},j_{k-1})]d_N(i_k,j_k). \tag{2.23}$$

Com isso, a distância associada ao caminho total (ir de $(0,0)$ a (I,J)) é definida como

$$D = \sum_{k=1}^{K} d[(i_k,j_k)|(i_{k-1},j_{k-1})], \tag{2.24}$$

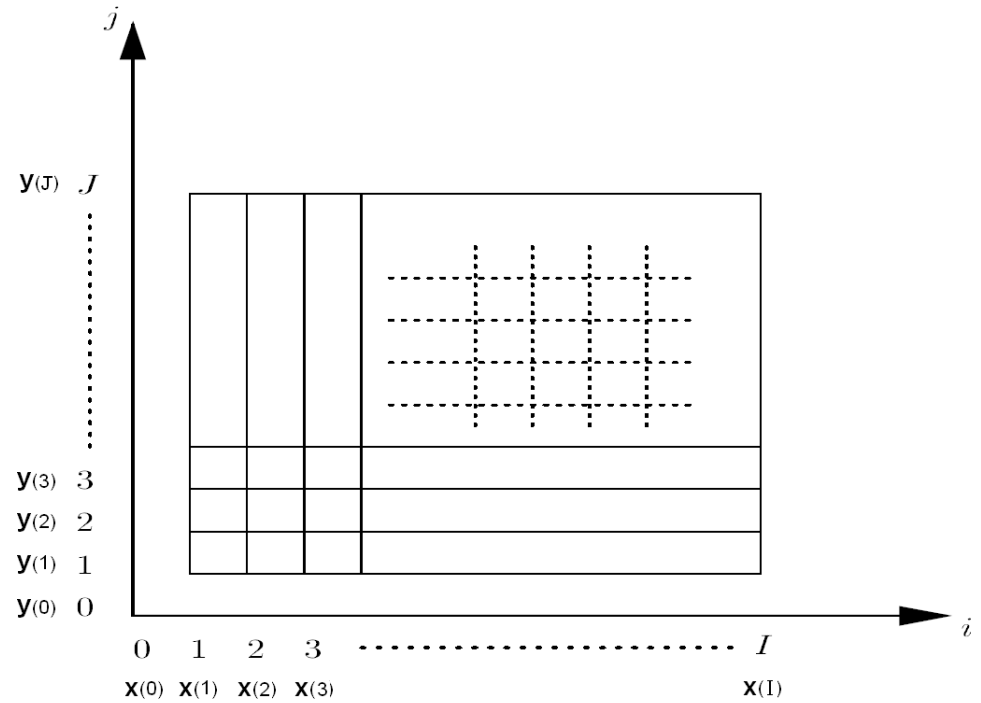


Figura 2.7: Discretização do plano utilizada no estudo do algoritmo DTW.

em que K é o número de transições no caminho. O objetivo é encontrar uma caminho que minimize a função custo D. Para isto, é utilizado o Princípio da Otimalidade de Bellman (POB) (BELLMAN, 1957). Antes de enunciar o princípio, suponha que $(s,t) \xrightarrow{*} (u,v)$ corresponde ao caminho ótimo do ponto (s,t) ao (u,v), e $(s,t) \xrightarrow{(w,x)} (u,v)$ o caminho ótimo interligando os mesmos pontos mas passando por (w,x). Com isso, segundo Deller et al. (2000) o POB pode ser definido como

$$(s,t) \xrightarrow{(w,x)} (u,v) = (s,t) \xrightarrow{*} (w,x) \oplus (w,x) \xrightarrow{*} (u,v), \tag{2.25}$$

para qualquer s,t,u,v,w e x, tal que $0 \leq s,w,u \leq I$ e $0 \leq t,x,v \leq J$, em que $\oplus$ representa a concatenação, ou união, de caminhos.

O princípio implica que

$$(0,0) \xrightarrow{(i_{k-1},j_{k-1})} (i_k,j_k) = (0,0) \xrightarrow{*} (i_{k-1},j_{k-1}) \oplus (i_{k-1},j_{k-1}) \xrightarrow{*} (i_k,j_k), \tag{2.26}$$

e este resultado é de grande importância para determinação do caminho ótimo entre dois pontos, pois tem-se como consequência direta da Equação (2.26) que

$$D_{min}[(i_k,j_k)|(i_{k-1},j_{k-1})] = D_{min}(i_{k-1},j_{k-1}) + d[(i_k,j_k)|(i_{k-1},j_{k-1})], \tag{2.27}$$

em que $D_{min}[(i_k,j_k)|(i_{k-1},j_{k-1})]$ é a distância associada a $(0,0) \xrightarrow{(i_{k-1},j_{k-1})} (i_k,j_k)$, e $D_{min}(i_{k-1},j_{k-1})$

é igual à distância no caminho $(0,0) \xrightarrow{*} (i_{k-1}, j_{k-1})$.

Observe que a Equação (2.27) não representa o caminho ótimo. Para encontrá-lo é preciso considerar o conjuntos dos melhores caminhos "chegando" de todos os possíveis pontos predecessores (i_{k-1}, j_{k-1}) e escolher aquele com menor distância. Assim a distância ótima é

$$D_{min}(i_k, j_k) = \min_{(i_{k-1}, j_{k-1})} \{D_{min}(i_{k-1}, j_{k-1}) + d[(i_k, j_k)|(i_{k-1}, j_{k-1})]\}. \tag{2.28}$$

Em suma, o POB diz que para encontrar o "melhor caminho" de $(0,0)$ a (i_k, j_k) que passa por um determinado predecessor (i_{k-1}, j_{k-1}) não é necessário re-examinar todas os caminhos parciais que partem de $(0,0)$ a (i_{k-1}, j_{k-1}). É possível simplesmente utilizar o percurso ótimo até o ponto predecessor na determinação do trajeto ótimo total (DELLER et al., 2000). Por este motivo, qualquer subcaminho, $(x,z)(w,v)$, da trajetória ótima total é também uma solução ótima entre estes dois pontos (MARQUES, 2005).

O custo total de um percurso qualquer até (i_k, j_k) pode ser definido como a soma dos custos associados a cada ponto no caminho, conforme definido na Equação (2.24). No contexto de reconhecimento de voz esse custo deve ser normalizado, permitindo assim que caminhos de diferentes tamanhos possuam uma igual oportunidade na determinação do caminho ótimo. O custo total normalizado entre duas elocuções (χ, γ) pode então ser dado por

$$D(\chi, \gamma) = \frac{\sum_{k=1}^{K} d[(i_k, j_k)|(i_{k-1}, j_{k-1})]}{\sum_{k=1}^{K} d_T[(i_k, j_k)|(i_{k-1}, j_{k-1})]}, \tag{2.29}$$

em que K é o tamanho (número de segmentos) do caminho. Será observado adiante que para determinadas restrições, o custo total de transição é igual a K, sendo ele também referido como fator de normalização de caminho. Com isso, a Equação (2.29) é referida como função custo do algoritmo DTW.

Observa-se que a minimização da Equação (2.29) é intuitivamente adequada para elocuções de uma mesma classe, mas não para classes diferentes. Isso acontece porque a minimização irrestrita dela pode levar a um "bom" encaixe (*matching*) entre duas elocuções linguisticamente diferentes, tornando o método não adequado para o propósito de reconhecimento. Isto posto, é necessário definir um conjunto de restrições consistentes com as características do problema de reconhecimento de voz.

Restrição de Fronteira

A restrição de fronteira determina os limites de início e fim do caminho percorrido no cálculo da distância entre duas elocuções, ou seja,

$$\begin{aligned}(i_1, j_1) &= (1,1),\\ (i_K, j_K) &= (I,J).\end{aligned} \tag{2.30}$$

Quando é assumido que o recorte das elocuções está bem determinado, então esta restrição não ocasiona erros significativos no alinhamento temporal. No entanto, visto que os limites de início e fim de uma elocução não são, em geral, bem definidos, alguns autores sugerem adotar uma certa flexibilidade nesta restrição (DELLER et al., 2000).

Restrição de Monotonicidade

Qualquer caminho escolhido no algoritmo DTW deve ser monotônico, significando que

$$\begin{aligned}i_{k-1} &\leq i_k,\\ j_{k-1} &\leq j_k.\end{aligned} \tag{2.31}$$

Esta restrição permite que um "candidato" a caminho ótimo não possua ciclos negativos, o que invalidaria o POB.

Restrições Globais e Locais

As restrições globais visam limitar o espaço de busca de um caminho ótimo, reduzindo significativamente o número de pontos na busca e consequentemente o custo computacional. Uma das restrições globais amplamente utilizadas é a de Itakura (1975).

Na implementação de um algoritmo DTW deve-se também escolher uma restrição de caminho local, que objetiva limitar o número de opções em torno de um determinado nó. Rabiner & Juang (1993) e Sakoe & Chiba (1978) apresentam diversos tipos de restrições locais.

Neste trabalho será utilizada a restrição local ilustrada na Figura 2.8, a qual representa que só é possível chegar a um dado ponto (i_k, j_k) através de (i_{k-1}, j_k), (i_{k-1}, j_{k-1}) ou (i_k, j_{k-1}).

Além das restrições globais e locais é necessário definir um tipo de custo de transição. As principais abordagens foram propostas por Sakoe & Chiba (1978). Nesta dissertação é utilizado

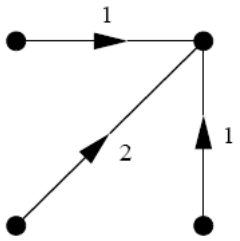


Figura 2.8: Exemplo de restrição local de caminho.

custo de transição dado por

$$d_T[(i_{k-m}, j_{k-m})|(i_{k-m-1}, j_{k-m-1})] = [i_{k-m} - i_{k-m-1}] + [j_{k-m} - j_{k-m-1}], \tag{2.32}$$

o qual fornece custos para a restrição local mostrada na Figura 2.8. Desta forma, o fator de normalização da Equação (2.29) é igual a $I+J$, que é igual a K quando considerada a restrição de fronteira. Com isso, a distância mínima referente ao caminho parcial que converge em (i_k, j_k) é dada por

$$\min \begin{cases} D_{min}(i_{k-1}, j_k) + d(i_k, j_k), \\ D_{min}(i_{k-1}, j_{k-1}) + 2d(i_k, j_k), \\ D_{min}(i_k, j_{k-1}) + d(i_k, j_k). \end{cases} \tag{2.33}$$

em que $D_{min}(i_{k-1}, j_k)$ é a distância associada ao caminho $(0,0) \xrightarrow{*} (i_{k-1}, j_k)$.

Finalmente, pode-se então elaborar um algoritmo DTW útil para o processo de reconhecimento de voz.

Algoritmo DTW para Reconhecimento de Voz

A procura da distância do caminho ótimo do algoritmo DTW usa como base uma matriz de custo entre todos os segmentos. Inicia-se a procura a partir do ponto $(0,0)$, cujo custo associado é $\mathbf{C}[0][0] = 0$. Em seguida, dentro da primeira coluna dessa matriz calculam-se as distâncias totais entre o ponto $(0,0)$ para todos os demais pontos desta coluna. É feito o mesmo para a segunda coluna e assim por diante até que se chegue ao ponto (I,J). Com isso, o valor $\mathbf{C}[I][J]$ expressa o menor custo possível e a medida de similaridade entre um padrão de teste e um padrão de referência.

No Algoritmo 1.5.1 é apresentado o pseudo-código da busca com DTW.

Algoritmo 2.4.1: DTW(**teste**, **ref**)

$I \leftarrow tamanhoTeste$

$J \leftarrow tamanhoRef$

$\mathbf{C}[0][0] = euclidiana(\mathbf{teste}[0], \mathbf{ref}[0])$

para $j \leftarrow 1$ **até** J

  faça $\{\mathbf{C}[0][j] \leftarrow \mathbf{C}[0][j-1] + euclidiana(\mathbf{teste}[0], \mathbf{ref}[j])$

para $i \leftarrow 1$ **até** I

$$\textbf{faça} \begin{cases} \mathbf{C}[i][0] \leftarrow \mathbf{C}[i-1][0] + euclidiana(\mathbf{teste}[i], \mathbf{ref}[0]) \\ \textbf{para } j \leftarrow 1 \textbf{ até } J \\ \quad \textbf{faça} \begin{cases} d \leftarrow euclidiana(\mathbf{teste}[i], \mathbf{ref}[j]) \\ \mathbf{C}[i][j] \leftarrow \min\{\mathbf{C}[i-1][j] + d, \mathbf{C}[i-1][j-1] + 2d, \mathbf{C}[i][j-1] + d\} \end{cases} \end{cases}$$

retorne $(\frac{\mathbf{C}[I][J]}{I+J})$

A utilização de programação dinâmica, por meio do POB, reduz consideravelmente a complexidade computacional do processo de minimização da Equação (2.29), reduzindo o número de operações de I^J para IJ (MARQUES, 2005).

Em sistemas RAV nos quais uma única referência por palavra é usada, a regra de decisão do Vizinho Mais Próximo (NN - *Nearest Neighbor*) (MARQUES, 2005) é comumente usada. Nos demais casos, pode-se utilizar variações desta, tal como a dos K-Vizinhos Mais Próximos (KNN - *K Nearest Neighbors*). Para este último caso, define-se

$$r_j = \frac{1}{K'} \sum_{i=1}^{i=K'} d(\chi, \chi_i^j) \tag{2.34}$$

em que $d(\chi, \chi_i^j)$ é a distância do i-ésimo vizinho mais próximo associado à palavra j, K' denota o número de elocuções de referência para uma classe de elocução. A palavra reconhecida j^* está associada a

$$r_{j^*} \leq r_j, \tag{2.35}$$

para $j = 1, 2, ..., U$, em que U representa o número de classes de palavras.

Abordagens para geração de referência do algoritmo DTW

Um dos problemas chaves na utilização do algoritmo DTW é como devem ser escolhidos os vetores de referência (*templates*) ou protótipos para uma determinada classe. A seguir são descritas duas abordagens presentes neste trabalho.

Treinamento Casual

Nas situações em que existem apenas poucas elocuções por classe e um vocabulário pequeno, o sistema de reconhecimento pode utilizar como referências os próprios vetores de treinamento. Esta abordagem é denominada treinamento casual.

Para que esta abordagem seja útil as elocuções deve ser capturadas, em geral, sob condições controladas de ruído e tempo, de modo a produzir referências fieis à elocução pronunciada. Rabiner & Juang (1993) ainda ressaltam que a abordagem pode falhar em vocabulários constituídos de elocuções similares ou confusas, ou mesmo em sistemas independentes do locutor.

Abordagem da Menor Distância Total - MDT

Lima (2000) expõe um método de determinação de referência, cuja a referência é a elocução de treinamento que apresentar o menor somatório das distâncias, utilizando DTW, em relação às outras elocuções da mesma classe.

Matematicamente, a abordagem corresponde a determinar o índice i^*da elocução vencedora:

$$i^* = \arg\min_{\forall i} \sum_{i=1}^{L} \sum_{j=1}^{L} d(\chi_i, \chi_j), i \neq j, \tag{2.36}$$

em que L é o número de elocuções disponíveis para a classe. Com isso, χ_{i^*} é a referência escolhida.

2.4.3 Quantização Vetorial

Quantização vetorial (QV) é um problema cuja a origem está na engenharia de telecomunicações, mais especificamente na transmissão codificada de informação através de canais de comunicação ruidosos e/ou com banda-passante limitada. A idéia básica de QV é reduzir a redundância no grupo original de dados (voz ou imagem) através da construção de um conjunto de vetores denominados vetores-código (*codevectors*). O conjunto de vetores-código é chamado de dicionário (*codebook*).(CRUZ, 2007)

O conceito de QV pode ser facilmente aplicado para o projeto de classificadores de padrões. Suponha U classes de elocuções para serem reconhecidas. Assim, coletam-se U conjuntos de treinamento $\{\mathbf{x}_t^{(i)}\}$, em que $i = 1, 2, ..., U$. Cada conjunto de treinamento deve conter elocuções da mesma classe. Então, U dicionários $\{\beta_i\}_{i=1}^{U}$ são projetados para minimizar uma medida de adequação destes aos dados que estão mapeando. Em geral, a métrica utilizada é o erro de quantização, E_q, definido em detalhes no Capítulo 3.

Durante o reconhecimento, os U dicionários são usados para implementar quantizadores vetoriais, tal como a Figura 2.9. Observa-se ainda que os quantizadores devem ser idênticos entre si quanto ao número total de unidades, dimensionalidade e parâmetros de treinamento.

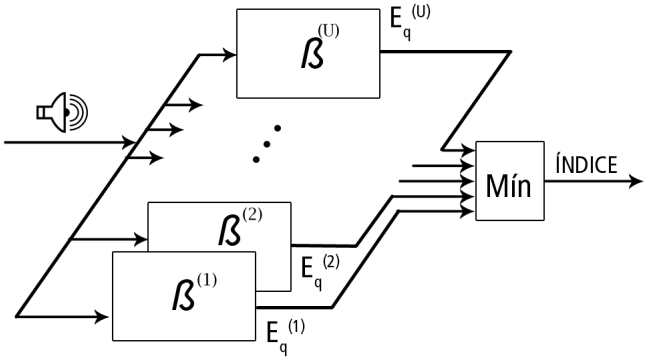


Figura 2.9: Abordagem que utiliza quantização vetorial no reconhecimento de voz.

No processo de reconhecimento, uma elocução desconhecida $\chi = \{\mathbf{x}_t\}_{t=1}^{T_n}$ é quantizada vetorialmente pelos U quantizadores, resultando em distorções $D(\beta_i)$, $i = 1,2,...,U$, em que

$$D(\beta_i) = \frac{1}{T_n} \sum_{t=1}^{T_n} d(\mathbf{x}_t, \mathbf{w}^*), \tag{2.37}$$

com $\mathbf{w}^* \in \beta_i$ satisfazendo

$$\mathbf{w}^* = \arg \min_{\forall \mathbf{w} \in \beta_i} d(\mathbf{x}_t, \mathbf{w}). \tag{2.38}$$

A palavra reconhecida i^* é a associada ao *codebook* β_{i^*}, tal que

$$D(\beta_{i^*}) = \min_{\forall i} D(\beta_i). \tag{2.39}$$

Em resumo, cada quantizador vetorial β_i é treinado somente com vetores de características extraídos de uma mesma classe de palavras, tornando-se especialista na quantização destes vetores. Quando uma nova elocução é apresentada, eles competem entre si, sendo declarado vencedor aquele que fornecer o menor erro de quantização E_q. A palavra associada ao quantizador vencedor é considerada reconhecida.

Supondo que após a etapa de extração de característica tem-se T_n vetores. Estes são quantizados por U modelos, cada um com V vetores protótipos. Dessa forma, o número de cálculos de distâncias é UVT_n, pois para encontrar o protótipo mais próximo para cada vetor de carac-

terística é necessário o cálculo de uma distância, ou equivalente, para cada protótipo de cada dicionário. Além disto, o cálculo de distância geralmente é dada com complexidade linear em função da dimensão P, $O(P)$. Portanto, a complexidade de uma abordagem convencional de quantização vetorial é dada por $O(UVT_nP)$.

Uma das possíveis abordagens que permite reduzir o tempo de processamento é proposta em Karpov (2003). Nela, os vetores de teste podem ser quantizados antes do cálculo em relação aos centróides, como ilustrado na Figura 2.10. Uma alternativa, proposta em Rabiner & Juang (1993) é usar QV como pré-processador para eliminar palavras que certamente não se encaixam com uma elocução desconhecida, com isso reduzindo o custo computacional de, por exemplo, um algoritmo DTW no processo de reconhecimento. No Capítulo 4 serão estudadas técnicas de computação que permitirão a redução do número de operações desta abordagem.

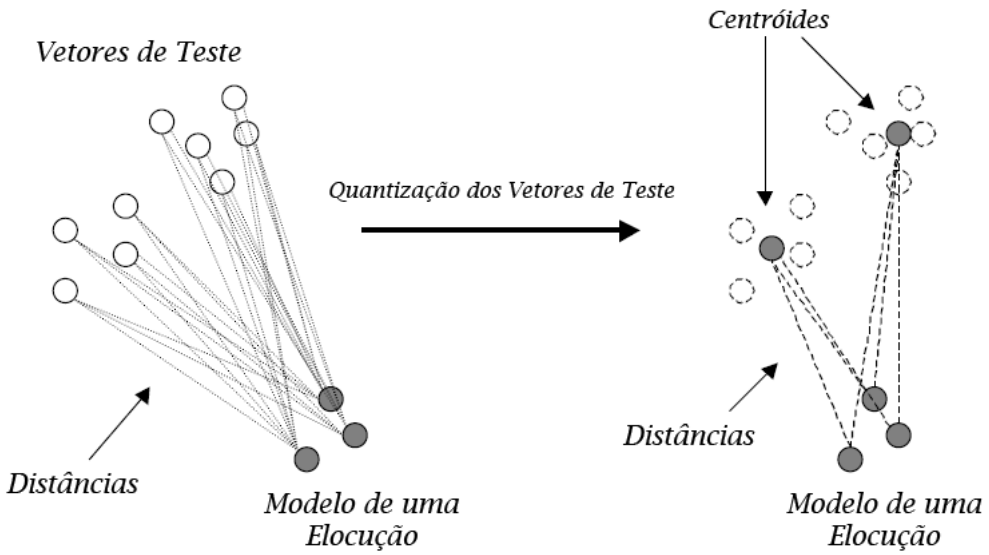


Figura 2.10: Abordagem alternativa para reconhecimento de voz.

2.5 Conclusões

Neste capítulo foram apresentados os principais componentes de um sistema básico de reconhecimento de voz. Assim, foram brevemente descritas a etapa de detecção de extremos, extração de características e o projeto do classificador de padrões, bem como a complexidade computacional temporal de alguns algoritmos relacionados a essas etapas. A principal razão para o não aprofundamento na complexidade dos algoritmos é que, em geral, eles apresentam parâmetros bem diferentes, de forma que uma avaliação mais adequada será feita no Capítulo 6. Desta forma, o intuito aqui foi fornecer somente uma primeira noção da ordem da complexidade

dos algoritmos.

Com relação à detecção de extremos será utilizada a abordagem de Rabiner & Sambur (1975). Esta etapa permite uma diminuição da quantidade de informação passada à etapa de extração de características, esta que pode ser bastante custosa computacionalmente pois envolve cálculo de funções de autocorrelações, janelamento, pré-ênfase e coeficientes de predição linear.

Conforme visto, a pesquisa em reconhecimento de voz não é muito recente, e ainda hoje inúmeros trabalhos são publicados nesta área, grande parte deles relacionados a aplicação de técnicas novas da área de reconhecimento de padrões para reconhecimento da fala contínua.

No projeto do reconhecedor serão avaliados, neste trabalho, basicamente algoritmos pertencentes ao contexto de reconhecimento de padrões, incluindo Redes Neurais Artificiais, a técnica clássica DTW, e abordagens baseadas em quantização vetorial, dentre as quais a rede auto-organizável de Kohonen.

Capítulo 3

Rede Neural Auto-Organizável de Kohonen

Redes Neurais Artificiais (RNAs) podem ser, de forma geral, divididas em duas categorias: redes com aprendizado supervisionado e não-supervisionado. No primeiro caso, cada entrada apresentada à rede vem acompanhada de uma resposta (saída) desejada e, então, os pesos sinápticos[1] da rede são ajustados de modo que a saída seja a mais próxima possível daquela desejada. No caso não-supervisionado, a rede neural detecta padrões e características estatísticas do espaço de entrada, construindo uma representação compacta desse espaço no conjunto de pesos sinápticos de seus neurônios.

Neste capítulo, será estudado o algoritmo de treinamento e teste da rede auto-organizável de Kohonen. Além disso, serão descritas métricas para cálculo de dissimilaridade entre vetores, bem como medidas de avaliação do algoritmo. Em seguida, o custo computacional do treinamento da rede será apresentado, esclarecendo as etapas envolvidas no processo de aprendizagem e visando fornecer uma visão geral do número de operações executadas em função dos principais parâmetros da rede.

3.1 Redes Neurais Não-Supervisionadas Competitivas

Grosso modo, redes neurais não-supervisionadas tentam extrair características estatísticas predominantes nos dados de entrada e constroem, de forma auto-organizada (i.e. sem auxílio externo e sem conhecimento prévio), uma representação reduzida do espaço de entrada, codificando-

[1] É assumido que o leitor está familiarizado com a nomenclatura básica utilizada em redes neurais. Caso contrário, uma introdução ao assunto pode ser encontrada em HAYKIN (1994).

a em seus pesos sinápticos.

Para utilizar uma rede neural não-supervisionada é preciso ter em mãos um número finito de N exemplos de treinamento, cada um deles representado como um vetor $\mathbf{x}(t) \in \mathbb{R}^n$, ou seja,

$$\mathbf{x}(t) = \begin{pmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{pmatrix}, \tag{3.1}$$

em que $t = 1, 2, \ldots, N$, indica o instante de apresentação deste vetor à rede durante o treinamento ou teste da mesma.

Cada componente $x_i(t)$ carrega alguma informação relevante para a análise em questão, sendo denominada normalmente de característica (*feature*) ou atributo (*attribute*). Dessa forma, um vetor $\mathbf{x}(t)$ é, normalmente, chamado de vetor de características (*feature vector*) ou vetor de atributos (*attribute vector*) no contexto de reconhecimento de padrões. Através do mapeamento dessas características é que as redes não-supervisionadas podem construir sua própria representação do espaço de entrada. Parte dos métodos abordados neste trabalho utilizam um subgrupo das redes neurais não-supervisionadas, chamado *redes neurais competitivas*.

Redes competitivas constituem uma das principais classes de redes neurais artificiais, nas quais um único neurônio ou um pequeno grupo deles, chamados *neurônios vencedores*, são ativados de acordo com o grau de proximidade entre seus vetores de pesos e o vetor de entrada atual, grau este medido segundo alguma métrica. Este tipo de algoritmo é comumente utilizado em tarefas de reconhecimento e classificação de padrões, tais como análise de agrupamentos (*clustering*), quantização vetorial e visualização de dados. Nestas aplicações, o vetor de pesos associado ao neurônio vencedor é utilizado como um vetor-protótipo representativo de um determinado grupo de vetores de entrada.

O processo de competição entre neurônios de uma rede competitiva é formulado de tal maneira que um certo neurônio i é escolhido como o neurônio vencedor para o vetor de entrada atual, se a seguinte relação for satisfeita:

$$i^*(t) = \arg\min_{\forall i} \|\mathbf{x}(t) - \mathbf{w}_i(t)\|, \tag{3.2}$$

na qual $i^*(t)$ é o índice do neurônio vencedor na rede, $\mathbf{x}(t)$ é um vetor de entrada da rede na iteração t e $\mathbf{w}_i(t) \in \mathbb{R}^n$, é o vetor de pesos associado ao neurônio i.

3.2 Mapa Auto-Organizável de Kohonen

O Mapa Auto-Organizável (*Self-Organizing Map* - SOM), proposto por Kohonen (1982), é uma rede neural não-supervisionada que possui neurônios dispostos em uma grade (*grid*) fixa, geralmente uni ou bi-dimensional, de modo que se possa definir uma relação de vizinhança espacial entre neurônios da grade.

Além da etapa de determinação do neurônio vencedor, apresentada na Seção 3.1, a rede SOM possui uma etapa cooperativa em que neurônios próximos do vencedor são atualizados em direção ao vetor de entrada, de acordo com a seguinte regra de aprendizagem:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta(t)h(i^*,i;t)[\mathbf{x}(t) - \mathbf{w}_i(t)]. \tag{3.3}$$

em que $h(i^*,i;t)$ é a função de vizinhança, geralmente do tipo gaussiana, ou seja:

$$h(i^*,i;t) = \exp\left(-\frac{\|\mathbf{i}(t) - \mathbf{i}^*(t)\|^2}{\vartheta^2(t)}\right), \tag{3.4}$$

em que $\vartheta(t)$ define o raio de influência da função de vizinhança, enquanto $\mathbf{i}(t)$ e $\mathbf{i}^*(t)$ são, respectivamente, as posições dos neurônios i e i^* no arranjo geométrico da rede.

A função de vizinhança funciona como uma janela de ponderação (*weighting window*), fazendo com que os neurônios mais próximos do neurônio vencedor atual tenham seus vetores de pesos atualizados mais intensamente que aqueles mais distantes do neurônio vencedor. Este, então, tem seus pesos reajustados com maior intensidade, visto que para ele tem-se $h(i^*,i;t) = 1$. Para todos os outros neurônios, tem-se $h(i^*,i;t) < 1$.

Por questões de convergência e estabilização do aprendizado, a função de vizinhança deve decrescer no tempo, ou seja, o raio de influência $\vartheta(t)$ decai com o decorrer do treinamento:

$$\vartheta(t) = \vartheta_0 \left(\frac{\vartheta_\varepsilon}{\vartheta_0}\right)^{(t/\varepsilon)}, \tag{3.5}$$

no qual ϑ_0 e $\vartheta_\varepsilon \ll \vartheta_0$ são os valores inicial e final de ϑ, e ε o número máximo de iterações ou épocas do treinamento. O decaimento apresentado acima é conhecido como decaimento exponencial. Em suma, a Equação (3.5) faz com que a vizinhança diminua com o passar das iterações.

Em razão de sua arquitetura peculiar e de seu algoritmo de treinamento, a rede SOM implementa uma projeção não-linear Φ do espaço de entrada, contínuo ou discreto, $\psi \subset \mathbb{R}^n$ (espaço dos dados), em um espaço de saída discreto $\mathscr{A}$, representado pelo espaço das coordenadas dos neurônios na grade, tal que $\dim(\mathscr{A}) \ll n$. Matematicamente, esta projeção pode ser simbolizada

por:

$$\Phi: \psi \rightarrow \mathscr{A} \tag{3.6}$$

A rede SOM tem tido grande utilização em aplicações de mineração de dados e reconhecimento de padrões. Grande parte do seu sucesso se deve à combinação de dois princípios essenciais de *auto-organização de sistemas*: (*i*) competição entre neurônios por recursos limitados, implementada pela Equação (3.2); e (*ii*) cooperação, implementada pela função vizinhança. O resultado da atuação destes dois princípios na rede SOM é uma projeção Φ que preserva relações de proximidade espacial entre os dados de entrada; ou seja, o mapeamento preserva a topologia do espaço de entrada no espaço de saída, conforme ilustrado na Figura 3.1. Nesta, $\dim(\psi) = n = 2$ e $\dim(\mathscr{A}) = 1$, os pontos pretos correspondem às coordenadas dos vetores de pesos do *i*-ésimo neurônio. Neurônios que são vizinhos na grade unidimensional são conectados por linhas tracejadas.

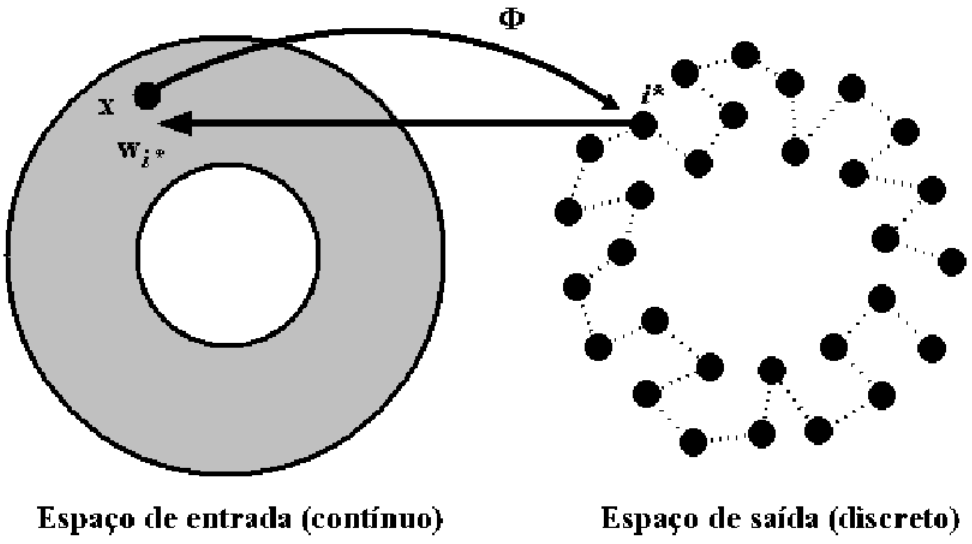


Figura 3.1: Projeção implementada pela rede SOM.

Pode-se expressar a propriedade preservação de topologia da rede SOM da seguinte forma. Sejam $\mathbf{x}_1$ e $\mathbf{x}_2$ dois vetores no espaço de entrada ψ. Sejam $\mathbf{i}_1^*$ e $\mathbf{i}_2^*$ as coordenadas dos neurônios vencedores para $\mathbf{x}_1$ e $\mathbf{x}_2$, respectivamente. Diz-se que a rede SOM, corretamente treinada, preserva a topologia do espaço de entrada se as seguintes relações forem observadas:

$$\|\mathbf{x}_1 - \mathbf{x}_2\| \rightarrow 0 \qquad \Rightarrow \qquad \|\mathbf{i}_1^* - \mathbf{i}_2^*\| \rightarrow 0, \tag{3.7}$$

ou seja, se quaisquer dois vetores estão fisicamente próximos no espaço de entrada, então eles terão neurônios vencedores espacialmente próximos na rede. As principais conseqüências desta

propriedade são listadas a seguir.

- *Aproximação do espaço de entrada:* a rede SOM constrói uma aproximação discreta do espaço de entrada, na qual cada neurônio da rede representa uma determinada região do espaço de entrada que define sua **região de atração** ou **campo receptivo** (*receptive field*). Esta região é conhecida também como **célula de Voronoi** (*Voronoi cell*). Assim, uma das principais aplicações da rede SOM é a categorização de dados não-rotulados em agrupamentos (*clusters*) e sua posterior utilização na classificação de vetores de características que não estavam presentes durante o treinamento.

- *Estimação pontual da função densidade de probabilidade:* o mapeamento da rede SOM reflete variações na estatística do espaço de entrada. Ou seja, regiões no espaço de entrada ψ de onde as amostras $\mathbf{x}$ têm uma alta probabilidade de ocorrência são povoadas com um maior número de neurônios, possuindo, conseqüentemente, uma melhor resolução do que regiões em ψ de onde amostras $\mathbf{x}$ são retiradas com baixa probabilidade de ocorrência.

O algoritmo da rede de Kohonen pode então ser sumarizado nos seguintes passos.

1. Para cada padrão de entrada $\mathbf{x}(t)$,

   (a) Busca-se pelo neurônio vencedor, $i^*(t)$, para o vetor de entrada $\mathbf{x}(t)$, usando a Equação (3.2).

   (b) Atualizam-se os vetores de pesos do neurônio vencedor e de seus vizinhos:

   $$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta(t)h(i^*, i; t)[\mathbf{x}(t) - \mathbf{w}_i(t)]. \tag{3.8}$$

   em que $0 < \eta(t) < 1$ denota o passo de aprendizagem.

2. Atualização dos parâmetros da função vizinhança e do passo de aprendizagem, segundo a Equação 3.5.

3. Se um critério de convergência predeterminado for alcançado, parar. Caso contrário, voltar para passo 1.

Em geral, os valores iniciais dos pesos são atribuídos de forma aleatória e uniforme no intervalo $[0, 1]$. Alternativamente, os vetores de pesos iniciais podem ser selecionados a partir do próprio conjunto de vetores de treinamento. Na prática, um critério de convergência bastante utilizado é o número de iterações ou épocas de treinamento ε.

3.3 Medidas Distância em Espaços Vetoriais

Existem diversas métricas para calcular o tamanho, ou norma, de um vetor $\mathbf{u} = [u_1\ u_2\ ...\ u_n]^T \in \Re^n$. Elas servem para medir similaridade (correlação) ou dissimilaridade (distância) entre vetores. Entre as mais comuns estão:

1. Norma Euclidiana
$$\| \mathbf{u} \|_2 = \sqrt{u_1^2 + u_2^2 + ... + u_n^2}. \tag{3.9}$$

2. Norma Manhattan ou Quarteirão
$$\| \mathbf{u} \|_1 = |u_1| + |u_2| + ... + |u_n|, \tag{3.10}$$

em que $|.|$ representa o valor absoluto.

No entanto, essas não são as únicas formas de definir o tamanho de um vetor. Seja p for um número real tal que $p \geq 1$, a norma L^p de $\mathbf{u}$ é definida por:

$$\| \mathbf{u} \|_p = \left(\sum_{i=1}^{n} |u_i|^p \right)^{1/p}. \tag{3.11}$$

A Equação (3.11) consiste de uma generalização também conhecida como métrica Minkowski. Quando p é igual a 1, a Equação (3.11) se reduz à Equação (3.10), a norma Manhattan, e para $p = 2$, à norma euclidiana. Observe ainda que para $p < 1$ a Equação (3.11), embora ainda seja uma métrica (para $0 < p < 1$), não satisfaz as propriedades de uma norma, definidas por

- Somente o vetor nulo tem norma zero.
- O módulo do vetor varia linearmente quando multiplicado por um escalar.
- O módulo da soma de dois vetores é menor ou igual à soma dos módulos dos vetores, propriedade também conhecida como desigualdade triangular.

A seguir são mostradas outras métricas que podem ser utilizadas para cálculo de distância e similaridade em redes neurais competitivas.

3.3.1 Correlação ou Produto Escalar

Comparação de vetores é muitas vezes baseada nas suas correlações, que é uma medida de similaridade. Assuma dois vetores, com valores reais $\mathbf{x} = [x_1\ x_2\ ...\ x_n]^T$ e $\mathbf{y} = [y_1\ y_2\ ...\ y_n]^T \in \Re^n$.

A correlação não normalizada, ou produto escalar, é definida por

$$C = \sum_{i=1}^{n} x_i y_i = \mathbf{x}^T \mathbf{y} = \mathbf{y}^T \mathbf{x}. \tag{3.12}$$

Se a informação relevante está contida somente na magnitude relativa entre suas componentes, então a similaridade pode ser medida em termos da direção co-seno definida a seguir. Dado $\mathbf{x}$ e $\mathbf{y} \in \Re^n$, então

$$\cos\theta = \frac{\mathbf{x}^T \mathbf{y}}{\| \mathbf{x} \|_2 \| \mathbf{y} \|_2} \tag{3.13}$$

é por definição a direção co-seno do ângulo, θ, entre os dois vetores, em que $\| \mathbf{x} \|_2$ é a norma euclidiana de $\mathbf{x}$. Note que se as normas dos vetores são unitárias, então a Equação (3.13) torna-se igual à Equação (3.12).

3.3.2 Distância Mahalanobis

Antes de definir a distância de Mahalanobis é necessário entender os conceitos de variância e covariância entre duas variáveis características, modeladas aqui como variáveis aleatórias (VA).

A variância de uma VA é definida como a esperança ou valor médio dos desvios quadráticos desta variável em relação à sua média. Para calcular a média deve-se conhecer a função densidade de probabilidade (ou massa de probabilidade, para o caso de variáveis discretas) da variável aleatória em questão. Como na prática em geral não se conhece essa distribuição, estes momentos estatísticos precisam ser estimados. Estimativas amostrais não-viesadas[2] para a variância e a média de uma V.A., X, são respectivamente,

$$\widehat{\sigma}_x^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \overline{x})^2 \tag{3.14}$$

e

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i, \tag{3.15}$$

em que N é o número de amostras disponíveis da variável aleatória.

A covariância entre duas variáveis aleatórias x_i e x_j mede a tendência da variação conjunta das variáveis, e é definida por:

$$E[(x_i - \overline{x}_i)(x_j - \overline{x}_j)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x_i - \overline{x}_i)(x_j - \overline{x}_j) f_{X_i,X_j}(x_i, x_j) dx_i dx_j, \tag{3.16}$$

[2]Não viesado é um termo utilizado para denotar que a média do estimador é igual à da variável aleatória estimada.

em que f_{X_i,X_j} é a densidade de probabilidade conjunta de x_i e x_j e $E[.]$ é o operador esperança matemática. Conforme explicado para a variância, a covariância pode também ser estimada por:

$$\widehat{\sigma}_{ij} = \frac{1}{N-1}\sum_{k=1}^{N}(x_i^k - \overline{x}_i)(x_j^k - \overline{x}_j), \tag{3.17}$$

As covariâncias entre várias VAs podem ser representadas na forma matricial, chamada Matriz de Covariância:

$$\Sigma = \begin{pmatrix} E[(x_1-\overline{x}_1)^2] & E[(x_1-\overline{x}_1)(x_2-\overline{x}_2)] & \cdots & E[(x_1-\overline{x}_1)(x_d-\overline{x}_d)] \\ E[(x_2-\overline{x}_2)(x_1-\overline{x}_1)] & E[(x_2-\overline{x}_2)^2] & \cdots & E[(x_2-\overline{x}_2)(x_d-\overline{x}_d)] \\ \vdots & \vdots & \vdots & \vdots \\ E[(x_d-\overline{x}_d)(x_1-\overline{x}_1)] & E[(x_d-\overline{x}_d)(x_2-\overline{x}_2)] & \cdots & E[(x_d-\overline{x}_d)^2] \end{pmatrix}. \tag{3.18}$$

A distância Mahalanobis entre dois vetores $\mathbf{x}$ e $\mathbf{y} \in \Re^n$ pode então ser definida como,

$$D_M(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^T \Sigma^{-1}(\mathbf{x}-\mathbf{y}), \tag{3.19}$$

em que Σ^{-1} é a inversa da matriz de covariância.

Existem basicamente duas formas para calcular a matriz de covariância de modo que essa métrica possa ser utilizada para cálculo de dissimilaridade na rede SOM. A primeira utiliza uma única matriz de covariância para todos os protótipos, e a segunda, uma matriz para cada protótipo.

A primeira abordagem pode ainda ter duas variantes: Utilizar a mesma matriz de covariância na forma $\sigma^2\mathbf{I}$ para todos os neurônios, em que σ^2 é uma variância média e $\mathbf{I}$ a matriz identidade. Isso é equivalente a considerar as variáveis aleatórias, formadas pelos atributos, não correlacionadas e com iguais variâncias. A segunda variante é a utilização da matriz Σ na sua forma completa, construída a partir de todo o conjunto de dados, para todos os protótipos.

A segunda abordagem utiliza uma matriz de covariância para cada protótipo, ou seja, cada protótipo i tem sua própria matriz de covariância Σ_i montada utilizando os dados mapeados no neurônio i. Younis et al. (1996) propõem inicializar toda matriz de covariância Σ_i igual a Σ, que é a matriz de covariância criada utilizando todos os dados. Em seguida, para cada padrão de entrada inserido na rede, o seu vencedor armazena o vetor $\mathbf{x}$. Após cada época, as matrizes de covariância são então re-estimadas a partir dos dados que cada protótipo mapeou ou armazenou.

Sarasamma & Zhu (2006) propuseram um algoritmo iterativo para estimar parâmetros de distribuições de probabilidade. Ele pode também ser utilizado para atualizar a matriz de covariância. Neste algoritmo, para cada novo padrão $\mathbf{x}$ é encontrado o vencedor $i^*(\mathbf{x};t)$. Caso

o vencedor seja o mesmo que o da época anterior $i^*(\mathbf{x};t-1)$ a matriz deste protótipo não é atualizada, caso contrário, $\mathbf{x}$ é utilizado para adaptar a matrix $\Sigma_{i^*(\mathbf{x};t)}$, e a contribuição de $\mathbf{x}$ na matriz de covariância $\Sigma_{i^*(\mathbf{x};t-1)}$ é subtraída.

Infelizmente, existem problemas associados ao uso dessa técnica: (i) Para avaliar a matriz de covariância para padrões de alta dimensionalidade n, uma grande quantidade de padrões ($\gg n^2$) deve ser coletado, para que uma boa estimativa da matriz possa ser alcançada, e para evitar problemas de singularidade. (ii) O custo computacional adicional para inversão da matriz.

3.4 Medidas de Avaliação da Rede SOM

Esta seção apresenta métodos para medir a qualidade da formação de mapas auto-organizáveis. Duas das propriedades que estas métricas ou medidas tentam avaliar são preservação da topologia e quantização vetorial.

3.4.1 Erro de Quantização Médio

O erro de quantização está geralmente relacionado a todos os tipos de quantizadores vetoriais e algoritmos de análise de agrupamentos. O erro de quantização é calculado determinando-se a distância média entre os vetores de entrada e seus respectivos neurônios vencedores, sendo representado matematicamente por

$$E_Q = \frac{1}{N}\sum_{i=1}^{N} \| \mathbf{x}_i - \mathbf{w}_{i^*} \|, \tag{3.20}$$

em que $\mathbf{x}_i$ é o i-ésimo padrão de entrada, $\mathbf{w}_{i^*}$ é o vetor de pesos do neurônio vencedor para o padrão $\mathbf{x}_i$, e N é o número de vetores de entrada.

Para qualquer conjunto de dados, o erro de quantização pode ser reduzido simplesmente por aumentar o número de protótipos na rede. Assim, essa métrica não deve ser utilizada na comparação entre mapas de tamanhos diferentes.

O erro de quantização ainda pode ser utilizado como medida de auxílio na escolha do número de neurônios da rede. Primeiro, plota-se um gráfico do erro de quantização por número de protótipos. Este gráfico tende a decair com o crescimento do número de protótipos. No entanto, em geral, utiliza-se uma quantidade de protótipos correspodente ao ponto em que o erro de quantização começa a estabilizar, ponto esse também conhecido com “joelho da curva”.

3.4.2 Erro de Quantização Quadrático Médio

O Erro de Quantização Quadrático Médio (MSQE - *Mean Square Quantization Error*) também pode funcionar como métrica para avaliação da rede SOM. Da mesma forma que o erro de quantização médio, esse erro é somente uma medida de adequação dos protótipos na representação dos dados de entrada e pode ser utilizado em qualquer algoritmo de quantização vetorial. O erro de quantização quadrático médio é definido como

$$E_{QM} = \frac{1}{N}\sum_{i=1}^{N} \| \mathbf{x}_i - \mathbf{w}_{i^*} \|^2. \tag{3.21}$$

3.4.3 Erro Topológico

O erro topológico (KIVILUOTO, 1996) é uma medida que busca por pontos no espaço de entrada em que o segundo vizinho mais próximo, ou segundo vencedor, não pertence à vizinhança do vencedor. Alguns autores, como Kaski & Lagus (1996), utilizam o termo continuidade para referir-se a métricas dessa natureza. O erro topológico é então definido por

$$E_T = \frac{1}{N}\sum_{i=1}^{N} u_i, \tag{3.22}$$

em que

$$u_i \;=\; \begin{cases} 1, & i^{**} \notin V_{i^*}, \\ 0, & c.c., \end{cases} \tag{3.23}$$

em que i^{**} é o segundo neurônio vencedor para o padrão $\mathbf{x}_i$, V_{i^*} é o conjunto de neurônios vizinhos imediatos do vencedor i^*, e N é o número de vetores de entrada.

Neste algoritmo, para cada vetor de entrada deve-se encontrar o segundo neurônio mais próximo. Se seu protótipo não for vizinho imediato do vencedor então é computado um erro. Ao término da apresentação de todos os padrões, a medida é normalizada dividindo-se o número de erros pelo número de vetores de entrada. Assim, o erro topológico é limitado entre zero e um, zero significando uma topologia sem erros.

3.4.4 Métrica de Kaski-Lagus

Kaski & Lagus (1996) propuseram uma métrica que busca unir uma medida de continuidade do mapeamento dos dados ao mapa (erro topológico) com uma medida de precisão do mapa na representação dos dados (erro de quantização). Esta abordagem determina quão contínuo

é o mapeamento do espaço de entrada no mapa. Na Figura 3.2 são mostrados exemplos de continuidade e descontinuidade em mapas. Quanto mais distante no *grid* estiver o segundo neurônio vencedor do primeiro vencedor para um padrão **x**, mais descontínuo será o mapa na região próxima a esse padrão (KASKI, 1997).

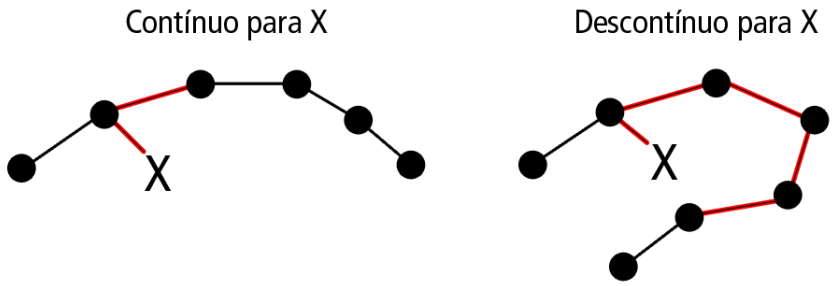


Figura 3.2: Exemplo de continuidade e descontinuidade em uma rede SOM-1D.

Considere $Q(i)$ o índice do i-ésimo neurônio ao longo do caminho Q do mapa que interliga $Q(1) = i^*$ e $Q(f) = i^{**}$, com f representando o índice do neurônio final do caminho e segundo vencedor. Sendo Q um caminho no mapa em que os neurônios $Q(i)$ e $Q(i+1)$ são vizinhos imediatos, então a métrica de Kaski & Lagus (1996) pode ser definida por

$$E_{MK} = \frac{1}{N}\sum_{i=1}^{N}(\parallel \mathbf{x}_i - \mathbf{w}_{i^*} \parallel + \min_{Q}\sum_{j=1}^{f-1} \parallel \mathbf{w}_{Q(j)} - \mathbf{w}_{Q(j+1)} \parallel), \tag{3.24}$$

em que $\mathbf{w}_{Q(j)}$ representa os pesos do j-ésimo neurônio do caminho Q.

Uma das motivações para proposição da métrica, segundo Kaski (1997), é que redes SOM de baixa dimensionalidade muitas vezes falham quando são usadas para modelar dados de alta dimensionalidade. Descontinuidades no mapeamento poderiam indicar tais falhas. O algoritmo pode ser implementado seguindo os seguintes passos.

1. Para cada vetor de entrada $\mathbf{x}(t)$

   (a) Identificar o segundo vizinho mais próximo i^{**}.

   (b) Traçar o menor caminho que interliga i^* e i^{**}, respeitando a vizinhança. Qualquer um dos neurônios pertencentes a este caminho deve ser vizinho imediato (adjacente) do neurônio anterior.

   (c) Somar a distância total do caminho anterior (encontrado em (b)) com o erro de quantização para $\mathbf{x}(t)$.

2. Calcular a média de todas as distâncias encontradas.

Caso existam descontinuidades no espaço de entrada, então a métrica terá um valor superior ao caso contínuo, pois computará uma distância maior. Assim como para o erro topológico, valores baixos da métrica Kaski-Lagus indicam bons mapas.

Kaski (1997) observou que para dados esparsos, em que poucos padrões são representados por protótipo, medidas para avaliação da preservação topológica tornam-se mais difíceis e imprecisas. Ele então sugere a utilização de um terceiro neurônio mais próximo. Outro ponto importante é que o cálculo do menor caminho entre o vencedor e o segundo vencedor pode ser uma tarefa custosa computacionalmente e, assim, técnicas de programação dinâmica podem ser utilizadas para minimizar esse custo.

Diversas outras métricas têm sido propostas na literatura relacionada à rede de Kohonen. Entre as métricas mais comuns está a Distorção da rede SOM. Ritter & Schulten (1988) mostraram que definida a função vizinhança e, para conjuntos de dados discretos, existe uma função custo que o algoritmo SOM minimiza. Essa função é conhecida como a distorção da rede SOM e, pelo mesmo motivo do erro de quantização, não deve ser utilizada na comparação de mapas de tamanhos diferentes. Outras métricas podem ser encontradas em Minamimoto et al. (1995), Villmann et al. (1997) e Venna & Kaski (2001). Por fim, Polzlbauer (2004) compara as principais métricas abordadas neste trabalho.

3.5 Complexidade Computacional

Conforme citado anteriormente, o algoritmo da rede SOM pode ser dividido em *i*) determinação do vencedor, *ii*) atualização do vencedor, *iii*) cálculo da taxa de aprendizagem e função vizinhança. A seguir é mostrada a análise computacional do número de operações (multiplicação, adição, subtração e comparação) em cada etapa do algoritmo. Durante esta seção serão utilizados pseudo-códigos para auxiliar na contagem do número de operações.

Os algoritmos 3.5.1 e 3.5.2 ilustram detalhadamente a fase da busca pelo neurônio vencedor.

Algoritmo 3.5.1: BUSCAVENCEDOR(**x**)

$i^* \leftarrow 1$

$d_{min} \leftarrow$ DISTANCIAEUCLIDIANA$(\mathbf{x}, \mathbf{w}_1)$

para $i \leftarrow 2$ **até** M

faça
- $d_i \leftarrow$ DISTANCIAEUCLIDIANA$(\mathbf{x}, \mathbf{w}_i)$
- **se** $d_i < d_{min}$
  - **então**
    - $i^* \leftarrow i$
    - $d_{min} \leftarrow d_i$

retorne (i^*)

Algoritmo 3.5.2: DISTANCIAEUCLIDIANA$(\mathbf{x}, \mathbf{w})$

para $i \leftarrow 1$ **até** n

faça $d \leftarrow d + (\mathbf{x}(i) - \mathbf{w}(i)) * (\mathbf{x}(i) - \mathbf{w}(i))$

retorne (d)

Dado que M é o número de neurônios na rede e n a dimensão dos dados de entrada, a determinação do vencedor envolve o cálculo de M distâncias e $M-1$ comparações. Cada cálculo de distância euclidiana utiliza n subtrações, $n-1$ adições e n multiplicações. Assim, para cada padrão de entrada $\mathbf{x}$, a busca pelo vencedor envolve o cálculo de Mn subtrações, multiplicações e $M(n-1)$ adições. No entanto, o vencedor é determinado para todos os padrões de entrada (N), e para cada iteração ou época do algoritmo (ε). Assim, o custo total para a determinação do vencedor no SOM é $MnN\varepsilon$ multiplicações e subtrações, $MN\varepsilon(n-1)$ adições e $(M-1)N\varepsilon$ comparações.

Algoritmo 3.5.3: GAUSSIANA(d)

retorne $(\leftarrow e^{\frac{-d*d}{2*\sigma*\sigma}})$

Algoritmo 3.5.4: ATUALIZACAO($\mathbf{x}, i^*, \eta$)

para $i \leftarrow 1$ **até** M

faça
- $d_i \leftarrow$ DISTANCIAEUCLIDIANA($\mathbf{i}, \mathbf{i}^*$)
- $f \leftarrow$ GAUSSIANA(d_i)
- $aux \leftarrow \eta * f$
- **para** $j \leftarrow 1$ **até** n
  - **faça** $w_i(j) \leftarrow w_i(j) + aux * (x(j) - w_i(j))$

O número de operações na etapa de atualização envolve o cálculo de uma distância euclidiana entre as coordenadas dos neurônios. A variável v será usada para representar a dimensão das coordenadas do neurônio. Assim, conforme os algoritmos 3.5.4 e 3.5.3, no cálculo da distância serão utilizadas v subtrações e multiplicações e $v-1$ subtrações. Além disto, é necessário o cálculo da função vizinhança. Supondo a função gaussiana, por ser amplamente utilizada, esta tarefa utilizará 3 multiplicações, 1 divisão e 1 exponenciação (considerando aqui essa operação como única). A atualização em si, utiliza n somas e subtrações e $n+1$ multiplicações. Já que estas operações serão feitas para cada protótipo da rede, vetor de entrada e época, então multiplica-se cada termo por $MN\varepsilon$ e somam-se as operações em comum. Com isto, temos um total de $MN\varepsilon(4+v+n)$ multiplicações, $MN\varepsilon(v+n)$ subtrações, $MN\varepsilon(v-1+n)$ adições e $MN\varepsilon$ exponenciações e divisões.

A forma mais comum de decaimento de taxa de aprendizagem e da base da função vizinhança, é o decaimento exponencial, mostrado na Equação (3.5). Neste decaimento são necessárias 1 divisão (para o valor do expoente da função), 2 exponenciações e multiplicações (uma para a função vizinhança e outra para taxa de aprendizagem). Aplicada a cada época, o número de operações torna-se ε divisões, e 2ε multiplicações e cálculos de exponenciais. Observa-se que existem duas divisões a mais para determinação da relação entre valores iniciais e finais tanto da função de vizinhança, quanto para a taxa de aprendizagem. No entanto, estas operações serão ignoradas por não dependerem de nenhum parâmetro. Após todas as etapas do algoritmo, somam-se as operações em comum, e tem-se o número total de operações. Na Tabela 3.1 detalha-se o número de operações realizadas durante o treinamento da rede SOM.

Tabela 3.1: Número de operação da rede SOM.

	busca	atualização	adaptação	total
multiplicação	$NnM\varepsilon$	$NM\varepsilon(4+v+n)$	2ε	$\varepsilon[2+NM(2n+v+4)]$
divisão	-	$NM\varepsilon$	ε	$\varepsilon(NM+1)$
adição	$NM\varepsilon(n-1)$	$NM\varepsilon(v-1+n)$	-	$NM\varepsilon(2n+v-2)$
subtração	$NnM\varepsilon$	$NM\varepsilon(v+n)$	-	$NM\varepsilon(2n+v)$
comparação	$(M-1)N\varepsilon$	-	-	$N\varepsilon(M-1)$
exponenciação	-	$NM\varepsilon$	2ε	$\varepsilon(2+NM)$
total	$N\varepsilon(3Mn-1)$	$MN\varepsilon(3n+3v+5)$	5ε	

Como pode ser visto na Tabela 3.1, a etapa de atualização dos pesos é a que requer um maior número de operações. Observa-se ainda que tanto a busca quanto a atualização possuem complexidade linear em função do número de neurônios na rede. Assim, é comum referenciar a complexidade computacional da rede SOM como $O(M)$.

Como será visto adiante, não é necessário atualizar todos os protótipos da rede a cada novo padrão de entrada. Isto aconteceu aqui, pois a função vizinhança gaussiana é assintótica e, dessa forma, mesmo os neurônios mais distantes são atualizados. Outros tipos de função vizinhança e adaptação da taxa de aprendizagem podem ser utilizadas para reduzir o custo computacional da rede. Com isto, a etapa de busca pelo vencedor torna-se computacionalmente dominante e grande parte das técnicas para aceleração computacional do algoritmo SOM se concentram na otimização dessa busca.

3.6 Conclusões

Neste capítulo foram apresentados os aspectos fundamentais do algoritmo de treinamento e teste da rede auto-organizável de Kohonen. Além disto, foram descritas métricas para cálculo de dissimilaridade na rede, bem como medidas de avaliação do algoritmo.

Uma vez que um dos objetivos deste trabalho consiste em utilizar a rede SOM em sistemas embarcados, o custo computacional do algoritmo de treinamento da rede foi discutido detalhadamente, em termos do número de operações realizadas.

Conforme visto, as etapas de busca pelo vencedor e atualização da função vizinhança demandam mais esforço computacional durante o treinamento e teste da rede SOM. No capítulo a seguir serão estudadas técnicas para aceleração computacional dessas etapas.

Capítulo 4

Implementação Computacional Eficiente da Rede SOM

A rede SOM é amplamente utilizada em Engenharia e tarefas de análise de dados, mas raramente utilizada em problemas de tempo real. A razão para isso está no custo computacional associado à rede (SAGHEER et al., 2006). Existem duas abordagens principais para acelerar o treinamento e desempenho da rede SOM. A primeira utiliza exclusão de protótipos durante a busca, sendo chamadas aqui de técnicas baseadas na exclusão de protótipos. A segunda abordagem acelera a rede SOM com base no arranjo arquitetural da busca.

Kaski (1999) propôs um método que utiliza a decomposição dos vetores em subspaços de dimensão menor que a original e produto interno no cálculo das similaridades. Esta abordagem se mostra eficiente principalmente quando os vetores de entrada possuem alta dimensionalidade. O problema dessa abordagem está em definir a dimensão do subespaço, além da necessidade de computações *offline* de resíduos e projeções utilizadas pelo algoritmo.

Cheung & Constantinides (1993) compararam diversos algoritmos de busca rápida pelo vizinho mais próximo. Entre eles estão métodos de busca com distância parcial(PDS, *Partial Distance Search*) (BEI; GRAY, 1985); soma de componentes (SOC, *Sum Of Components*) (RA; KIM, 1991); teste de componentes (CT, *Component Testing*) (CHENG; GERSHO, 1986). Todos estes estão inseridos em uma categoria de métodos que excluem a busca por certos protótipos da rede. Além disto, todos os métodos são ótimos na determinação do protótipo vencedor.

Grande parte dos métodos de aceleração da busca foram propostos no contexto de quantização vetorial e podem ser então aplicados a qualquer algoritmo de quantização vetorial, não somente

à rede SOM. Exemplos desses métodos são as Árvores *K-D* (FRIEDMAN et al., 1977), Árvores SR (KATAYAMA; SATOH, 1997), Árvores-X (BERCHTOLD et al., 2001). A limitação destas abordagens é que são aplicáveis apenas a problemas de baixa dimensionalidade (WEBER et al., 1998). Outros métodos subótimos utilizados para reduzir o custo computacional da rede SOM, tais como Kushilevitz et al. (2000), Indyk & Motwani (1998), Kleinberg (1997), requerem grande quantidade de vetores de treinamento para promover uma quantidade aceitável de redução computacional (NISKANEN et al., 2002). Estes algoritmos fazem parte de técnicas baseadas na arquitetura da busca.

As principais abordagens baseadas na arquitetura da busca e adaptadas completamente à rede SOM, ou seja, que levam em conta as propriedades de quantização e preservação da topologia, são o mapa auto-organizável estruturado em árvore (TS-SOM, *Tree-Structured Self-Organizing Map*) (KOIKKALAINEN, 1994) e a arvóre topológica auto-organizável (SOTT, *Self-Organizing Topological Tree*) (XU; CHANG, 2004), (XU et al., 2005).

O algoritmo proposto por Kusumoto & Takefuji (2006) é composto de subdivisão e busca binária, além de não possuir função vizinhança. Su & Chang (2000) combinam a rede SOM com estrátegias baseadas em heurísticas. Sagheer et al. (2006) utilizam Análise dos Componentes Principais (PCA - *Principal Components Analysis*) (JOLLIFFE, 2002) para acelerar o treinamento da rede SOM.

Diante do vasto número de metodologias e algoritmos para acelerar a rede SOM, este trabalho analisará e avaliará apenas uma parte das abordagens. Com relação às técnicas baseadas na exclusão de protótipos serão abordadas a busca com distância parcial (BEI; GRAY, 1985), e a abordagem proposta por Kohonen (1997). Além disto, o algoritmo TS-SOM (KOIKKALAINEN, 1994) será avaliado. No final do capítulo serão expostos alguns aspectos importantes que podem ser utilizados na redução do número de operações no treinamento e teste da rede SOM.

4.1 Mapa Auto-Organizável Estruturado em Árvore

O mapa auto-organizável estruturado em árvore, ou simplesmente TS-SOM (*Tree-Structured Self-Organizing Map*), foi inicialmente proposto por Koikkalainen & Oja (1990). Em seguida, Koikkalainen (1994) incorporou melhorias e considerações importantes ao algoritmo. O TS-SOM surgiu como uma alternativa mais rápida para determinação do protótipo vencedor na rede SOM.

Koikkalainen & Oja (1990) tiveram como base o trabalho de Friedman et al. (1977), o qual propõe uma estrutura em árvore para resolução do problema da busca pelo protótipo vencedor

com uma complexidade temporal de $O(logM)$, em que M é o número de protótipos. Abordagem baseada em árvore para quantização vetorial foi desenvolvida anteriormente por Buzo et al. (1980), conhecida como TSVQ (*Tree-Structured Vector Quantization*). Mas, de fato, o algoritmo TSVQ não incorporou a característica de preservação topológica presente no algoritmo SOM, essencial em diversas aplicações (KOIKKALAINEN, 1994).

4.1.1 Algoritmo de Treinamento da Rede TS-SOM

A rede TS-SOM difere da rede SOM original em alguns pontos. Assim, é necessário definir alguns conceitos, como vizinhança de um neurônio, N_i; os filhos de um neurônio, S_i; e o conjunto de busca, SN_i.

Considere i o índice de um neurônio qualquer e G um grafo que interliga os neurônios da rede. A vizinhança N_i do neurônio i é definida como o conjunto $\{i\} \cup \{$os vizinhos imediatos de i, conforme definido por $G\}$, sendo que N_i não muda ao longo do treinamento. Essa é uma das principais diferenças com relação à rede SOM convencional. Na rede TS-SOM não há necessidade de redução da função vizinhança pois os mesmos efeitos podem ser atingidos através do procedimento de inicialização de novas camadas, nas quais os neurônios herdam os pesos sinápticos dos seus pais[1] (KOIKKALAINEN, 1994). Observe ainda que N_i pertence à mesma camada de i.

Se i denota um neurônio e H é uma p-árvore de neurônios, ou seja, uma árvore na qual cada neurônio tem p filhos. Os filhos do neurônio i, S_i, definem o conjunto de todos os neurônios filhos de i, conforme definido por H.

O conjunto de busca, SN_i, é definido como o conjunto formado por $\{S_g \cup S_i\}$, em que $g \in N_i$. Isto é, SN_i é o conjunto dos neurônios que são filhos do neurônio i e filhos dos vizinhos imediatos de i. Com isso, pode-se agora entender o algoritmo de treinamento da rede TS-SOM.

O algoritmo de treinamento é baseado na estrutura em árvore mostrada na Figura 4.1, em que os neurônios de uma mesma camada estão conectados via conexões laterais. Assim, cada camada contém, na verdade, redes SOM com diferentes resoluções. Os neurônios são adaptados camada por camada. Após o treinamento de uma camada, os pesos dos neurônios desta camada são "congelados", ou seja, seus pesos sinápticos não são alterados. A seguir mostram-se os principais passos do algoritmo.

1. Iniciar o nó raiz da árvore com valores aleatórios.

[1] Na nomenclatura utilizada em estrutura de dados em árvore, um nó pai é aquele relacionado, através de grafos, com nós da camada inferior.

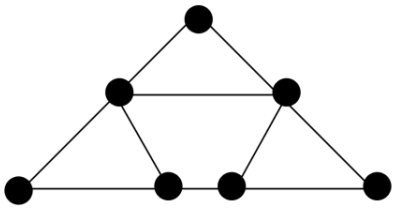


Figura 4.1: Estrutura em árvore binária.

2. Treinar a camada atual da árvore até que um critério de convergência seja alcançado. Durante este treinamento, os seguintes passos são realizados:

   (a) Escolher aleatoriamente um vetor de entrada do conjunto de dados de treinamento.

   (b) Encontrar o neurônio vencedor i^* na camada atual;

   (c) Atualizar os pesos de i^* e seus vizinhos N_{i^*} em direção ao vetor de entrada selecionado.

3. Iniciar os pesos dos neurônios da próxima camada com os valores dos pesos dos seus pais e treinar a nova camada (voltar para passo 2).

A etapa que mais difere da rede SOM original é a busca pelo neurônio vencedor. Na busca em árvore comum a busca vai de um nó da árvore ao seu filho mais similar ao padrão de entrada - filho vencedor. Já na rede TS-SOM, o conjunto de busca inclui os filhos do neurônio vencedor e de seus vizinhos imediatos. Na Figura 4.2 é ilustrado o conjunto de busca a partir de um dado nó, i. A inclusão dos filhos dos vizinhos imediatos do neurônio vencedor da camada anterior à adaptativa é bastante econômica computacionalmente, pois não depende do número de neurônios na camada. Este procedimento é vital para o funcionamento do algoritmo (KOIKKALAINEN; OJA, 1990).

A atualização na rede TS-SOM é bastante similar à rede SOM original. O neurônio vencedor i^*, e seus vizinhos topológicos imediatos, são ajustados em direção ao vetor de entrada. Se N_{i^*} é a vizinhança do neurônio i^*, η a taxa de aprendizagem e t um índice temporal, então uma regra típica de atualização é

$$\mathbf{w}_i(t+1) \quad = \quad \begin{cases} \mathbf{w}_i(t) + \eta[\mathbf{x}(t) - \mathbf{w}_i(t)], & i \in N_{i^*} \\ \mathbf{w}_i(t), & c.c. \end{cases} \tag{4.1}$$

A inicialização de uma nova camada engloba duas etapas seguintes.

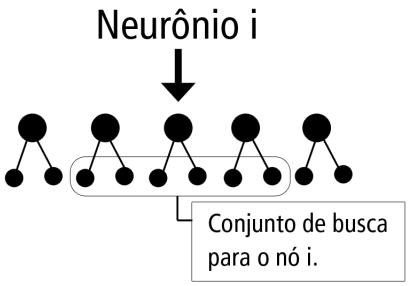


Figura 4.2: Conjunto de busca a partir do neurônio i.

1. Todos os neurônios de uma nova camada herdam os pesos dos seus pais.

$$\mathbf{w}_i = \mathbf{w}_g, \quad i \in S_g. \tag{4.2}$$

2. Os neurônios são movidos na direção dos seus vizinhos para ordenamento topológico. Assume-se que $\bar{\mathbf{w}}_g$ denota o vetor de pesos médio dos vizinhos imediatos ao nó g, e λ um número tal que $0 < \lambda << 1$. Após a inicialização da etapa anterior, os neurônios da nova camada são adaptados conforme a seguinte regra:

$$\mathbf{w}_i = \mathbf{w}_i + \lambda \bar{\mathbf{w}}_g, \quad g \in N_i. \tag{4.3}$$

4.1.2 Complexidade Computacional da Rede TS-SOM

Para a avaliação computacional da rede TS-SOM, considere M o número de neurônios na rede; M_c a quantidade de neurônios na última camada; c o número de camadas e p o número de filhos por nó [2]. Com definições acima e supondo que a árvore seja estática e completa, as seguintes relações podem ser facilmente verificadas.

$$M = \sum_{j=0}^{c-1} p^j. \tag{4.4}$$

$$M_c = p^{c-1}. \tag{4.5}$$

A partir das relações acima pode-se determinar o número de camadas, ou altura da árvore,

[2] Observe que p também representa o número máximo de vizinhos imediatos de um neurônio. Este é sempre igual ao número de filhos por nó em árvores estáticas e completas.

em função do número de nós M como

$$c = \lceil \log_p \{M(p-1)+1\} \rceil, \tag{4.6}$$

já que em alguns casos a árvore pode não estar completa, ou seja, nem todo nó possuir o número máximo de filhos ou nenhum, então foi adicionado à Equação (4.6) a função de arredondamento para cima, $\lceil . \rceil$, ou inteiro superior, fornecendo assim o valor exato do número de camadas.

A complexidade computacional da busca em árvore é $O(\log_p M)$, pois após a árvore estar balanceada[3] o número de nós buscados é, no pior caso, igual ao número de camadas c. No entanto, em cada camada da rede TS-SOM são buscados, no pior caso, $(p+1)p$ neurônios: $p+1$ representa o vencedor e seus p vizinhos imediatos, enquanto p, o número de filhos por nó da árvore. Ou seja, em uma rede TS-SOM (1D) com dois filhos, o número máximo de neurônios buscados por camada é igual a 6 (os filhos do vencedor e os filhos dos vizinhos imediatos ao vencedor). Como o número de camadas em uma árvore é igual $\lceil \log_p \{M(p-1)+1\} \rceil$, então o número de buscas por neurônios é $(p^2+p)\lceil \log_p \{M(p-1)+1\} \rceil$ e, por isso, a rede TS-SOM possui complexidade $O(\log_p M)$.

Como na rede TS-SOM não existe preservação topológica inter-camadas, a camada principal é a última, pois ela é a utilizada no teste da rede. As camadas superiores auxiliam a formação topólogica das inferiores e são essenciais para encontrar o vencedor na última camada de forma mais rápida. Com isto, o custo computacional da rede TS-SOM deve ser determinado também em função da quantidade de neurônios na última camada, para uma melhor comparação com a rede SOM convencional. Em suma, deve-se escrever a complexidade $O(\log_p M)$ em função de M_c.

Das Equações (4.4) e (4.5), tem-se que

$$M = \sum_{j=0}^{\log_p M_c} p^j \tag{4.7}$$

então

$$\log_p M = \log_p \left[\sum_{j=0}^{\log_p M_c} p^j \right]. \tag{4.8}$$

Percebe-se que $\sum_{j=0}^{\log_p M_c} p^j$ é a soma de uma progressão geométrica de razão p. Assim, esta soma pode ser reescrita como $p^0 \frac{p^{\log_p (M_c)+1}-1}{p-1} = \frac{pM_c-1}{p-1}$. Com isso, utilizando as propriedades de

[3] Uma árvore binária é dita balanceada se, para cada nó, as alturas de suas subárvores diferem de, no máximo 1.

funções logarítmicas, tem-se agora

$$\log_p M = \log_p (pM_c - 1) - \log_p (p - 1). \tag{4.9}$$

Ignorando o segundo termo da Equação (4.9), pois p é pequeno e, em geral, fixo. Além disto, observa-se que $\log_p (p-1) < 1$, então chega-se a

$$O(\log_p M) = O(\log_p (pM_c - 1)) = O(\log_p M_c). \tag{4.10}$$

Logo, demonstra-se que a complexidade da rede TS-SOM também varia logaritmicamente com o número de neurônios na última camada. A complexidade computacional em função de M_c, fundamental para comparação com o algoritmo SOM convencional, não é discutida na literatura.

4.1.3 Acelerando a Rede TS-SOM

Koikkalainen (1994) notou que em algumas situações a busca em árvore poderia utilizar uma tabela para associar a cada vetor de entrada $\mathbf{x}(t)$, um neurônio i pertencente à penúltima camada da rede e, então, iniciar a busca a partir deste neurônio associado, sem precisar percorrer toda a árvore. Desta forma, a busca só percorre um número fixo de neurônios, determinado pelo número de filhos por nó e pela quantidade de vizinhos topológicos imediatos, que conforme dito são grandezas de mesmo valor, chamada aqui de p. Assim, a complexidade computacional do algoritmo torna-se $O(1)$, pois a cada época e vetor de entrada a busca pelo vencedor é realizada sobre $(p^2 + p)$ neurônios.

É importante ressaltar ainda que durante o treinamento, para cada vetor de entrada $\mathbf{x}(t)$ é necessária a atualização de i (pertencente à penúltima camada), de forma que referencie o pai do neurônio vencedor quando $\mathbf{x}(t)$ é a entrada. Além disto, observe que esta estratégia para aceleração só pode ser aplicado no treinamento, não no teste, da rede.

4.2 Métodos de Exclusão de Protótipos

Esta seção irá expor alguns dos principais métodos que utilizam a exclusão de protótipos durante a busca pelo vencedor. É importante ressaltar que o termo *exclusão* não significa que os protótipos serão retirados da rede, mas somente desconsiderados durante a etapa de busca. A análise computacional assintótica não será feita para esses algoritmos, pois conforme será visto adiante, exceto para a busca com atalho, a busca pelo vencedor depende da estimativa

inicial para uma distância mínima. Nestes casos, uma análise em termos de valor médio é mais importante e viável, e será feita no Capítulo 5.

4.2.1 Busca com Distância Parcial

Buscando reduzir o número de operações nos algoritmos de quantização vetorial, Bei & Gray (1985) propuseram o algoritmo busca com distância parcial (PDS, *Partial Distance Search*). Este método reduz o custo computacional no cálculo da distância entre vetores pela metade ou mais, sem perda de eficiência (BEI; GRAY, 1985).

O algoritmo PDS é uma forma simples de reduzir o número de operações de multiplicação, adição e subtração na busca pelo neurônio vencedor. Nesta abordagem, quando a distância parcial quadrática entre um vetor de entrada e o i-ésimo vetor-protótipo excede a distância quadrática total encontrada até o instante atual na busca, este protótipo pode ser descartado. Isto é, pode-se descartar o protótipo em que a distância acumulada nas primeiras j ($j < n$) amostras é maior que a menor distância encontrada na busca até o presente momento. O pseudo-código 4.2.1 ilustra o funcionamento do algoritmo.

Algoritmo 4.2.1: PDS($\mathbf{x}$)

$i^* \leftarrow 1$
$d_{min} \leftarrow \text{DISTANCIAEUCLIDIANA}(\mathbf{x}, \mathbf{w}_1)$
para $i \leftarrow 2$ **até** M
faça
- $descarte \leftarrow$ **falso**
- $d \leftarrow 0$
- **para** $j \leftarrow 1$ **até** n
  - **faça**
    - $aux \leftarrow \mathbf{x}(j) - \mathbf{w}_i(j)$
    - $d \leftarrow d + aux * aux$
    - **se** $(d > d_{min})$
      - **então**
        - $descarte \leftarrow$ **verdadeiro**
        - **sair**
- **se** $(descarte =$ **falso**$)$
  - **então**
    - $i^* \leftarrow i$
    - $d_{min} \leftarrow d$

retorne (i^*)

Na busca com descarte antecipado de protótipos (PDS), o tempo adicional gasto para a

comparação $(d > d_{min})$ após o cálculo da distância em cada dimensão é, em média, menor que o tempo gasto na busca completa pelo vencedor (NISKANEN et al., 2002; BEI; GRAY, 1985).

4.2.2 Busca com Atalho

Kohonen (1997) propôs a busca com atalho (*Shortcut Winner Search*, em inglês) com o intuito de reduzir o custo computacional na busca pelo vencedor na rede SOM.

A solução encontrada por Kohonen é parecida com a aceleração da rede TS SOM. No entanto, a busca com atalho pode ser aplicada a qualquer arquitetura SOM e não necessita de divisão em camadas ou níveis hierárquicos.

Com o decorrer das iterações de treinamento de uma rede SOM, o mapa vai se tornando suavemente ordenado, e a largura da função vizinhança e amplitude da taxa de aprendizagem diminuem. Neste cenário, a probabilidade de o neurônio vencedor para um padrão $\mathbf{x}$ estar na vizinhança do neurônio vencedor para o mesmo padrão $\mathbf{x}$ na época anterior é alta.

Kohonen (1997) então sugere armazenar uma referência, ou ponteiro, relacionando um vetor $\mathbf{x}$ ao protótipo vencedor i^* na iteração t. Na iteração $t+1$ a busca pode ser feita na vizinhança imediata do neurônio i^* e, somente se um protótipo mais próximo de $\mathbf{x}$ for encontrado nessa vizinhança a busca continua na vizinhança desse novo neurônio e assim sucessivamente, até que o vencedor esteja no centro do domínio da busca. Após o vencedor ter sido identificado, a referência ao vetor $\mathbf{x}$ é então atualizada. É importante ressaltar que a cada nova vizinhança a ser procurada, somente os protótipos que não foram testados anteriormente precisam ser examinados.

Como uma forma de utilizar a idéia acima, Costa (1999) sugere a utilização de uma função raio de busca $R(i^*, i; t)$ que decresce ao longo do treinamento. Teoricamente, pode-se utilizar qualquer função monotônica decrescente, no entanto é preferível funções suaves e que se mantenham sempre superior ao raio de ação da função vizinhança ($R(i^*, i; t) > h(i^*, i; t)$) (COSTA, 1999).

Ainda em Kohonen (1997) e Costa (1999) é exposto um método de adição de neurônios dinamicamente. Esta inserção é realizada entre os neurônios existentes já treinados, através de interpolação linear, isto é, fazendo-se uma média dos pesos dos neurônios na vizinhança imediata de cada neurônio inserido.

Com isso, pode-se iniciar o treinamento de uma rede com $B \times Z$ neurônio. Após um número alto de épocas, t_1, são inseridos neurônios na rede atráves da interpolação linear, resultando em

uma rede de $(2B-1)\times(2Z-1)$ neurônios. Então, o treinamento continua até t_2, e assim sucessivamente até a rede atingir o tamanho desejado. Fazendo com que $t_1 > t_2 > ... > t_f$, em que f representa a última interpolação, pode-se obter um ganho computacional considerável, pois o custo computacional do treinamento convencional é de $O(\varepsilon M)$ para uma rede SOM com M neurônios, enquanto que pelo método em questão, o custo tem ordem $O(t_1M_1 + t_2M_2 + ... + t_fM_f) = O(t_fM_f)$, desde que $M = M_f$ e $\varepsilon = \sum_{j=1}^{f} t_j$.

4.3 Dicas para Redução do Número de Operações

Apesar da grande quantidade de técnicas para aceleração da rede SOM, outras ações simples podem reduzir significativamente o número de operações no treinamento e teste da rede. Nesta seção estudaremos algumas dessas ações.

4.3.1 Função Raiz Quadrada

Para calcular a distância euclidiana entre dois vetores uma função raiz quadrada é utilizada. No entanto, quando somente as relações entre as distâncias realmente importam, essa raiz não precisa ser calculada. Esta situação é exatamente o caso da busca pelo vencedor. Assim, na utilização da distância euclidiana como métrica para cálculo de distâncias entre dois vetores $\mathbf{x}$ e $\mathbf{y} \in \Re^n$, será utilizado na verdade o quadrado da distância euclidiana $d(\mathbf{x},\mathbf{y})^2$. Observe que no Capítulo 3, na análise computacional da rede SOM, a raiz quadrada já foi omitida.

4.3.2 Função Vizinhança

A função vizinhança mais utilizada na rede SOM é a função gaussiana. Porém, esta função é custosa do ponto de vista computacional, pois envolve o uso de exponenciais. Além disso, com o uso da função gaussiana, todos os neurônios da rede são atualizados, pois ela é uma função assintótica. Nesse contexto, Kohonen (1997) define uma vizinhança do tipo retangular, ou seja, dado um raio $r(t)$ do neurônio vencedor no instante t, $N_{i^*}(t)$ é o conjunto de neurônios tal que $\| \mathbf{i}^* - \mathbf{i} \| < r(t)$, então $h(i^*,i;t) = 1$ para todos os neurônios $i \in N_{i^*}(t)$ e $h(i^*,i;t) = 0$ caso contrário.

Outro tipo de função vizinhança que pode ser utilizada para reduzir o custo computacional da rede SOM é a função vizinhança triangular. Essa função deve ser centrada em zero, e como a distância entre dois neurônios no mapa é sempre um valor positivo, então só é necessária a utilização da região positiva da função. A função triangular aplicada à distância entre o neurônio

i^* e i, é definida como

$$h(i^*, i; t) = -\frac{b}{\vartheta(t)} \parallel \mathbf{i} - \mathbf{i}^* \parallel + b, \tag{4.11}$$

em que b representa o ponto do gráfico que toca o eixo das ordenadas, e deve ser igual a 1. $\vartheta(t)$ é o raio de atuação da função.

A grande vantagem computacional na utilização dessas abordagens é que somente um pequeno número de neurônios precisa ter seus pesos alterados para um determinado padrão $\mathbf{x}$, e além disso, não há cálculo de exponenciais, que são computacionalmente mais custosas do que somas, subtrações e multiplicações.

Uma outra possível função vizinhança é a gaussiana truncada. Neste caso, são calculadas as funções gaussianas para todos os neurônios, mas somente os neurônios que estiverem acima de um limiar são atualizados.

4.3.3 Decaimento Linear da Taxa de Aprendizagem

A utilização do decaimento conforme a Equação (3.5) pode ser bastante custosa computacionalmente, pois envolve o cálculo de ε divisões e 2ε multiplicações e exponenciações, incluindo o decaimento dos parâmetros da função vizinhança e da taxa de aprendizagem. Alternativamente ao decaimento exponencial, está o linear, definido conforme

$$\eta(t) = \eta(1) + (t-1)\frac{\eta(\varepsilon) - \eta(1)}{\varepsilon - 1}. \tag{4.12}$$

em que $\eta(\varepsilon)$ e $\eta(1)$ são os valores final e inicial da taxa de aprendizagem respectivamente.

Observe que o valor $\frac{\eta(\varepsilon)-\eta(1)}{\varepsilon-1}$ é calculado uma única vez (requerendo duas subtrações e uma divisão), antes do início do treinamento, sendo armazenado para utilização no decorrer do treinamento. Para cada época são necessárias 1 subtração, 1 multiplicação e 1 adição. Assim, o número total de operações envolvidas em ε épocas são $2+\varepsilon$ subtrações, 1 divisão, ε adições e ε multiplicações. A grande vantagem de utilizar decaimento linear está na não utilização de exponenciais, que são operações computacionalmente mais custosas, principalmente em sistemas embarcados. Observe ainda que o mesmo decaimento pode ser aplicado aos parâmetros da função vizinhança.

4.4 Conclusões

Este Capítulo descreveu uma série de metodologias e técnicas utilizadas para acelerar o treinamento da rede SOM. Elas podem ser divididas em métodos que utilizam exclusão de protótipos da busca e aqueles baseados em estrutura de dados ou arquiteturas. Dentre as diversas técnicas existentes, foram avaliadas a rede TS-SOM, a Busca com Distância Parcial, Busca com Atalho, além de algumas dicas para reduzir o custo computacional do treinamento/teste da rede.

No próximo capítulo, essas metodologias, assim como o algoritmo SOM convencional, serão avaliadas quanto à quantização vetorial, preservação da topologia e tempo de processamento.

Capítulo 5

Resultados: Técnicas de Aceleração da Rede SOM

Neste capítulo são apresentados os resultados da avaliação do desempenho de técnicas voltadas para a redução do custo computacional da rede SOM. Para isto, foi utilizado um conjunto de dados gerado artificialmente. Ele possui 600 padrões, com 50 dimensões, distribuídos uniformemente entre 0 e 1.

Todas as simulações foram desenvolvidas utilizando a linguagem de programação Java (*Java Development Kit* 6), em um computador com processador Intel Core 2 Duo de 2,33 GHz, utilizando o ambiente de desenvolvimento *Eclipse*.

Uma vez que as tarefas que dominam o processamento na rede SOM são a busca pelo vencedor e a atualização dos neurônios, estas serão as etapas avaliadas neste capítulo.

5.1 Metodologia de Simulação

As abordagens de aceleração da busca pelo vencedor testadas neste trabalho incluem as métricas euclidiana (com e sem o cálculo da raiz quadrada) e quarteirão. Além disto, os algoritmos PDS, Busca com Atalho, SOM-1D e TS-SOM-1D também foram testados. Para facilitar o entendimento e avaliação destes algoritmos a busca pelo vencedor foi dividida em três níveis, conforme mostrado na Figura 5.1. O primeiro nível se refere à etapa em que os neurônios são testados como possíveis vencedores, os algoritmos relativos a esta etapa são a busca exaustiva e a busca com atalho. O segundo nível se refere ao modo de calcular a distância entre os vetores de teste e os pesos dos neurônios. Este cálculo pode envolver todas as componentes do vetor,

completo, ou utilizando PDS. O terceiro nível consiste na utilização das métricas para cálculo de distâncias.

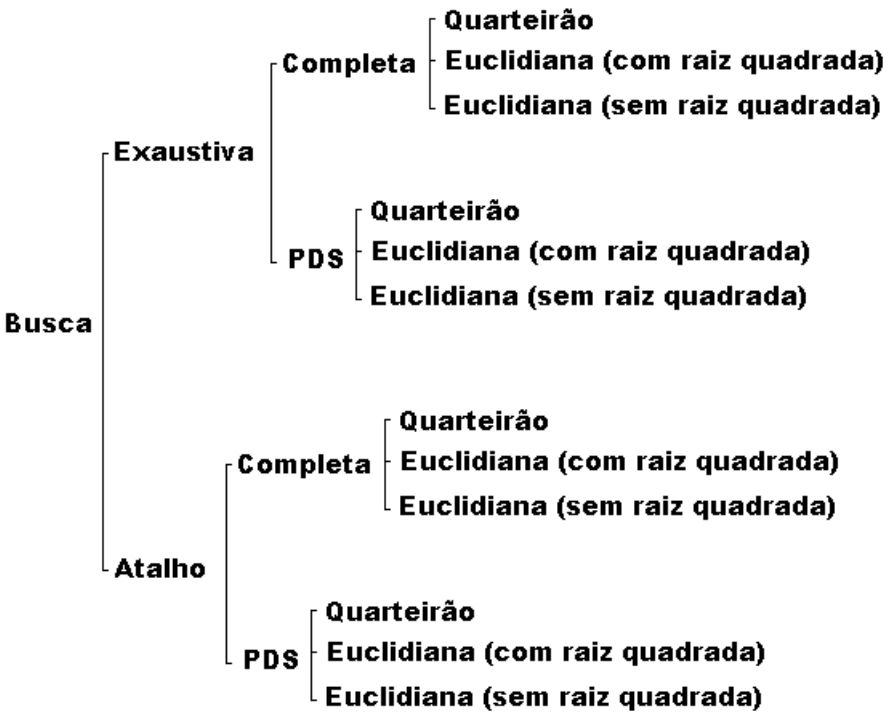


Figura 5.1: Divisão em níveis da etapa de busca pelo neurônio vencedor.

Os algoritmos foram avaliados com relação ao tempo médio de uma busca, tempo médio da etapa completa de busca pelo vencedor, erro médio de quantização, erro topológico médio, e os respectivos desvios-padrão.

O tempo médio de uma busca representa a média do tempo gasto para cálculo de uma operação de busca pelo vencedor. O tempo médio da estapa completa de busca (chamado também de tempo total) é a soma do tempo gasto a cada vez que se busca pelo vencedor. Ou seja, é o tempo total gasto com a busca durante todo o treinamento da rede.

Todas as medições de tempo realizadas neste trabalho utilizaram a função Java *System.nanoTime();*. Com esta função é possível obter o tempo, em nanosegundos, atual da máquina. Assim, para medir uma determinada etapa, essa função é colocada no início e fim da mesma, e então o tempo inicial é subtraído do tempo final e tem-se a duração da etapa.

Para que não houvesse interferência da etapa de atualização dos pesos nas medidas que avaliam o tempo de busca, a etapa de atualização foi mantida fixa, usando sempre a função vizinhança gaussiana. Exceto para a rede TS-SOM-1D, os parâmetros dos algoritmos implementados são os seguintes:

- Número de épocas ($\varepsilon = 400$).
- Raio Vizinhança inicial ($\vartheta_o = 30$).
- Raio Vizinhança final ($\vartheta_\varepsilon = 0,001$).
- Taxa de aprendizagem inicial ($\eta_o = 0,1$).
- Taxa de aprendizagem final ($\eta_\varepsilon = 0,001$).
- Decaimento da taxa de aprendizagem:

$$\eta_o(\frac{\eta_\varepsilon}{\eta_o})^{\frac{t}{\varepsilon}}, \tag{5.1}$$

  em que t representa a iteração atual.

- Decaimento do raio da função vizinhança:

$$\vartheta_t = \vartheta_o(\frac{\vartheta_\varepsilon}{\vartheta_o})^{\frac{t}{\varepsilon}}, \tag{5.2}$$

  em que t representa a iteração atual.

Devido à própria estrutura de treinamento da rede TS-SOM-1D seus parâmetros diferem dos apresentados acima. Com isto, os parâmetros utilizados para esta rede foram os seguintes:

- Número de camadas ($c = 9$).
- Número de filhos por neurônio ($p = 2$).
- Topologia Unidimensional.
- Taxa de aprendizagem ($\eta = 0,05$).
- Número de épocas ($\varepsilon = 4000$ e 10000).

Dentre as abordagens para atualização do vencedor e seu vizinhos, a principal etapa é a escolha da função de vizinhança. Nesta seção, são avaliadas a função Gaussiana, Retangular e Gaussiana Truncada, por serem as mais amplamentes utilizadas (KOHONEN, 1997). Além disto, foi avaliada a etapa de atualização da rede TS-SOM-1D. A etapa de busca foi mantida fixa ao longo da avaliação desta seção.

Seguindo a notação utilizada na seção anterior, a busca foi realizada, exceto para a rede TS-SOM, seguindo os níveis *Exaustiva+PDS+Euclidiana*. Os parâmetros e métodos fixados nestas simulações são listados a seguir:

- Métrica para cálculo de distância (Euclidiana sem raiz quadrada).
- Busca pelo vencedor (Exaustiva e PDS).
- Taxa de aprendizagem inicial ($\eta_o = 0,1$).
- Taxa de aprendizagem final ($\eta_\varepsilon = 0,001$).
- Número de épocas ($\varepsilon = 400$).
- Raio vizinhança inicial ($\vartheta_o = 30$).
- Raio vizinhança final ($\vartheta_\varepsilon = 0,001$).

Na descrição dos métodos foram utilizadas notações especiais. *Gaussiana* significa que foi utilizada uma função gaussiana na atualização dos pesos dos neurônios. Como ela é uma função assintótica, todos os neurônios são atualizados, mesmo aqueles mais distantes do vencedor. O termo *Gaussiana Truncada(v)* significa que somente os neurônios que obtiveram valor na função vizinhança gaussiana maior que *v* serão atualizados.

Neste trabalho, o decaimento utilizado para o raio de atuação $r(t)$ da função vizinhança Retangular foi linear até $r(t) = 1$. Desta forma, a notação *Retangular(v)* significa um decaimento linear de $r(t)$ de 30 até 1 em *v* épocas. Ou seja, a partir da *v*-ésima época, a função vizinhança só inclui os neurônios vizinhos imediatos. Da mesma forma da seção anterior, serão reportados o tempo médio da atualização e o tempo médio da total etapa de atualização durante todo o treinamento.

É importante ressaltar que tanto na avaliação da busca pelo vencedor quanto da etapa de atualização dos neurônios, foram realizadas 20 simulações independentes, sob as mesmas condições, para extrair as estatísticas das medidas de avaliação utilizadas neste trabalho. Para fins de simplificação, os resultados médios são ilustrados com relação ao melhor resultado, este marcado em negrito nas Tabelas deste capítulo. Isto torna mais clara a visualização relativa entre as abordagens testadas.

5.1.1 Conjunto de Dados Gerados para Simulação

Para avaliar as redes SOM e suas variantes, foi utilizado um conjunto de dados gerado artificialmente, com distribuição uniforme entre 0 e 1. O Código 1 ilustra a rotina Java usada para geração dos dados.

```
1 public void gerarDados(int nAmostras, int dimensao){
2         for(int i = 0; i < nAmostras; i++){
3                 for(int j = 0; j < dimensao; j++){
4                         System.out.print(Math.random() + " ");
5                 }
6                 System.out.println("");
7         }
8 }
```

Código 1: Função Java utilizada para gerar os dados aleatórios.

Os dados gerados possuem 600 amostras (variável *nAmostras*) e 50 dimensões (variável *dimensão*). A função responsável por gerar o número aleatório foi a *Math.random();*, pertencente ao pacote *java.lang*.

Com a utilização de dados uniformemente distribuidos se está considerando que a distribuição de probabilidade dos dados não afeta o desempenho do erro topológico e de quantização, desde que todas as redes sejam aplicadas ao mesmo conjunto de dados.

5.2 Resultados: Busca pelo Vencedor

Nesta seção, são reportados os resultados de desempenho dos algoritmos na etapa de busca pelo vencedor. Foram utilizadas algumas abreviações, de modo que *Exa.* significa Exaustiva; *Comp.*, Completa e *Eucli.*, Euclidiana. Na Tabela 5.1 são apresentados os resultados. Nesta Tabela observa-se que cada método, exceto a rede *TS-SOM*, é completamente identificado pelos três níveis. O número entre parênteses no método *Atalho* representa a época ou iteração na qual se iniciou a busca com Atalho. Isto acontece porque o método deve ser utilizado com o mapa já parcialmente ordenado, conforme explicado no Capítulo 4.

Conforme pode ser visto na Tabela 5.1, a rede *SOM 1D:Exa+PDS+Eucli* obteve o melhor desempenho quanto ao erro de quantização, com valores menores que os obtidos por redes bi-dimensionais. Os melhores resultados médios são mostrados em negrito.

As redes *TS-SOM* apresentaram os maiores erros de quantização médio. Observa-se ainda que o erro de quantização foi equivalente quando se utilizou *Atalho(200)* e *Atalho(250)*. Além disso, o algoritmo *PDS* não prejudica a propriedade de quantização das redes, uma vez que a combinação *Exa+PDS* é ótima na determinação do vencedor.

Tabela 5.1: Desempenho das técnicas de aceleração do SOM na busca pelo vencedor.

Métodos	**Erro Quantização**		**Erro Topológico**		**Tempo (ms)**		**Tempo Total (ms)**	
	Média	Desvio	Média	Desvio	Média	Desvio	Média	Desvio
Exa+Comp+Eucli+Raiz	+0,049	$1{,}43\ 10^{-2}$	+0,225	$3{,}87\ 10^{-2}$	+0,03629	$2{,}47\ 10^{-4}$	+10301,274	59,39
Exa+Comp+Eucli	+0,053	$1{,}03\ 10^{-2}$	+0,201	$3{,}04\ 10^{-2}$	+0,03047	$1{,}19\ 10^{-4}$	+8902,996	28,47
Exa+Comp+Quarteirão	+0,075	$1{,}11\ 10^{-2}$	+0,291	$3{,}49\ 10^{-2}$	+0,03106	$1{,}15\ 10^{-4}$	+9046,646	27,60
Exa+PDS+Eucli	+0,045	$9{,}33\ 10^{-3}$	+0,221	$2{,}92\ 10^{-2}$	+0,02503	$1{,}38\ 10^{-4}$	+7598,624	33,15
SOM 1D:Exa+PDS+Eucli	**1,310**	$5{,}15\ 10^{-3}$	+0,583	$1{,}99\ 10^{-2}$	+0,02291	$9{,}94\ 10^{-5}$	+7088,722	23,85
Atalho(250)+Comp+Eucli	+0,051	$8{,}86\ 10^{-3}$	+0,202	$4{,}22\ 10^{-2}$	+0,01765	$1{,}09\ 10^{-4}$	+5826,630	26,17
Atalho(250)+PDS+Eucli	+0,050	$1{,}14\ 10^{-2}$	+0,202	$4{,}15\ 10^{-2}$	+0,01620	$8{,}65\ 10^{-5}$	+5480,183	20,75
Atalho(200)+PDS+Eucli	+0,053	$1{,}08\ 10^{-2}$	+0,201	$3{,}91\ 10^{-2}$	+0,01312	$6{,}66\ 10^{-5}$	+4741,038	15,99
TS-SOM(4000 épocas)	+0,326	$2{,}93\ 10^{-3}$	+0,003	$8{,}13\ 10^{-3}$	**0,007799**	$2{,}21\ 10^{-4}$	**280,79**	7,97
TS-SOM(10000 épocas)	+0,304	$5{,}07\ 10^{-3}$	**0,0505**	$9{,}16\ 10^{-3}$	+0,00018	$9{,}40\ 10^{-5}$	+437,549	8,46

Com relação ao erro topológico, as redes bidimensionais novamente obtiveram valores similares. Dentre estas, o uso da norma *Quarteirão* provocou um aumento considerável no erro topológico. Os melhores resultados ficaram com as redes *TS-SOM*, e o pior com a rede *SOM 1D*. Isto já era esperado uma vez que redes unidimensionais possuem uma menor quantidade de vizinhos imediatos; assim, existe uma maior probabilidade de o segundo neurônio vencedor não ser vizinho do primeiro. No entanto, nas redes *TS-SOM*, essa limitação é superada pela etapa de inicialização de novas camadas.

Com o intuito de avaliar conjuntamente as diferentes medidas utilizadas na Tabela 5.1, foi gerado o gráfico de barras da Figura 5.2. Nele está ilustrado o desempenho médio dos algoritmos. Cada conjunto de barras representa um algoritmo, de modo que a comparação entre os algoritmos deve ser feita em relação às mesmas medidas. E como tais medidas possuem ordem de grandeza diferentes elas foram normalizadas, entre 0 e 1, pela soma. Ou seja, cada medida de um algoritmo é dividida pela soma da mesma medida dos outros algoritmos.

Da Figura 5.2 pode ser visto que a combinação das técnicas Busca com Atalho e PDS (Algoritmos 6, 7 e 8) reduziu consideravelmente o tempo da busca pelo vencedor, sem comprometer o desempenho na quantização vetorial e preservação da topologia. Vê-se também claramente o alto valor do erro de quantização da rede SOM-1D, e uma redução de tempo significativa da busca com a utilização das distâncias Euclidiana (sem raiz quadrada) e Quarteirão.

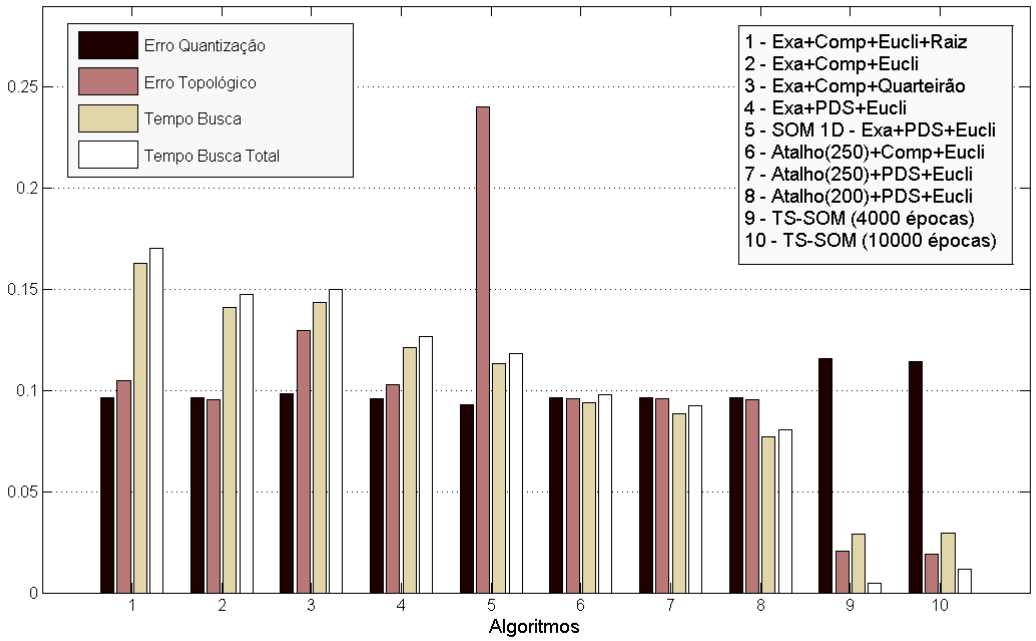


Figura 5.2: Comparação dos algoritmos na etapa de busca em função dos valores médios.

As redes TS-SOM obtiveram os melhores resultados tanto no tempo de processamento, quanto no erro topológico. Conforme constatado por Koikkalainen (1994), a rede apresenta um desempenho inferior aos demais com relação ao erro de quantização. Isto acontece pois a arquitetura TS-SOM implementa um método subótimo na determinação do vencedor. No entanto, em aplicações em que o tempo de processamento é um fator crítico esta rede é uma excelente alternativa. A rede *TS-SOM(4000 épocas)* obteve uma redução de 97,34% no tempo total de treinamento, quando comparado com o pior caso (*Exa.+Comp+Euclidiana+Raiz*). Observa-se ainda que uma maior quantidade de épocas de treinamento não implicou em uma redução do erro de quantização.

O gráfico de pizza da Figura 5.3 ilustra o tempo relativo, com normalização pela som, da busca dos algoritmos avaliados. Conforme ilustrado, uma busca em uma rede TS-SOM é cerca de 5 vezes mais rápida que em *Exa.+Comp+Euclidiana+Raiz*. Além disso, a utilização do algoritmo *Atalho(200)+PDS+Eucli* reduziu pela metade o tempo de processamento.

Observa-se ainda, na Figura 5.3, que a utilização da métrica quarteirão é equivalente à retirada da raiz quadrada na métrica euclidiana, quanto ao tempo da busca pelo vencedor.

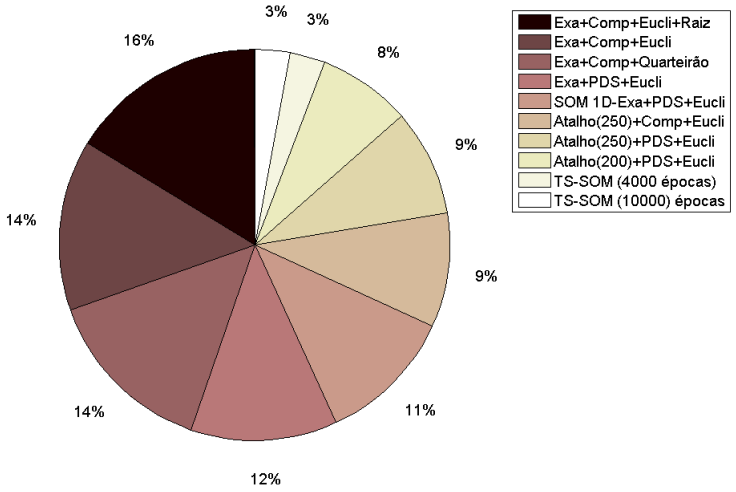


Figura 5.3: Tempos médios de busca pelo neurônio vencedor.

5.3 Resultados: Atualização dos Neurônios

Nesta seção são apresentados os resultados para os testes referentes à etapa de atualização dos neurônios da rede SOM. Na Tabela 5.2 estão mostrados os resultados dos métodos avaliados, com estatísticas geradas de 20 simulações independentes.

Tabela 5.2: Desempenho das técnicas de aceleração da rede SOM na atualização dos neurônios.

Métodos	**Erro Quantização**		**Erro Topológico**		**Tempo (ms)**		**Tempo Total (ms)**	
	Média	Desvio	Média	Desvio	Média	Desvio	Média	Desvio
Gaussiana	+0,049	$9{,}33\ 10^{-3}$	+0,221	$2{,}92\ 10^{-2}$	+0,337	$5{,}93\ 10^{-3}$	+81272,875	1423,02
SOM 1D	+0,004	$5{,}15\ 10^{-3}$	+0,583	$1{,}99\ 10^{-2}$	+0,267	$5{,}33\ 10^{-3}$	+64369,311	1279,31
GaussianaTruncada(0,001)	+0,060	$1{,}09\ 10^{-2}$	+0,189	$4{,}69\ 10^{-2}$	+0,295	$3{,}00\ 10^{-3}$	+71203,356	720,51
GaussianaTruncada(0,01)	+0,051	$1{,}26\ 10^{-2}$	+0,209	$4{,}74\ 10^{-2}$	+0,295	$3{,}24\ 10^{-3}$	+71241,815	777,52
Retangular(100)	**1,307**	$9{,}39\ 10^{-3}$	+0,331	$2{,}86\ 10^{-2}$	+0,247	$4{,}53\ 10^{-3}$	+59733,312	1088,38
Retangular(200)	+0,100	$4{,}48\ 10^{-3}$	+0,187	$1{,}92\ 10^{-2}$	+0,254	$4{,}80\ 10^{-3}$	+61359,911	1152,95
TS-SOM(4000 épocas)	+0,329	$2{,}93\ 10^{-3}$	+0,003	$8{,}13\ 10^{-3}$	$\mathbf{1{,}71\ 10^{-3}}$	$1{,}77\ 10^{-4}$	**62,037**	6,37
TS-SOM(10000 épocas)	+0,307	$5{,}07\ 10^{-3}$	**0,0505**	$9{,}16\ 10^{-3}$	$\mathbf{1{,}71\ 10^{-3}}$	$1{,}57\ 10^{-4}$	+92,311	14,14

O melhor desempenho médio com relação ao erro de quantização foi atingido utilizando a vizinhança *Retangular(100)*. O pior resultado ficou com a rede *TS-SOM*, mantendo coerência com os resultados da seção anterior. As redes com piores resultados quanto ao erro topológico

foram a *Gaussiana Retangular(100)* e *SOM 1D*. A rede com menor erro topológico foi a rede *TS-SOM*.

O gráfico em barras da Figura 5.4 exibe o desempenho dos métodos avaliados. Observa-se uma redução de 81334,912 ms (*Gaussiana*) para 62,037 ms com a rede *TS-SOM*, significando uma redução de 99% no tempo total.

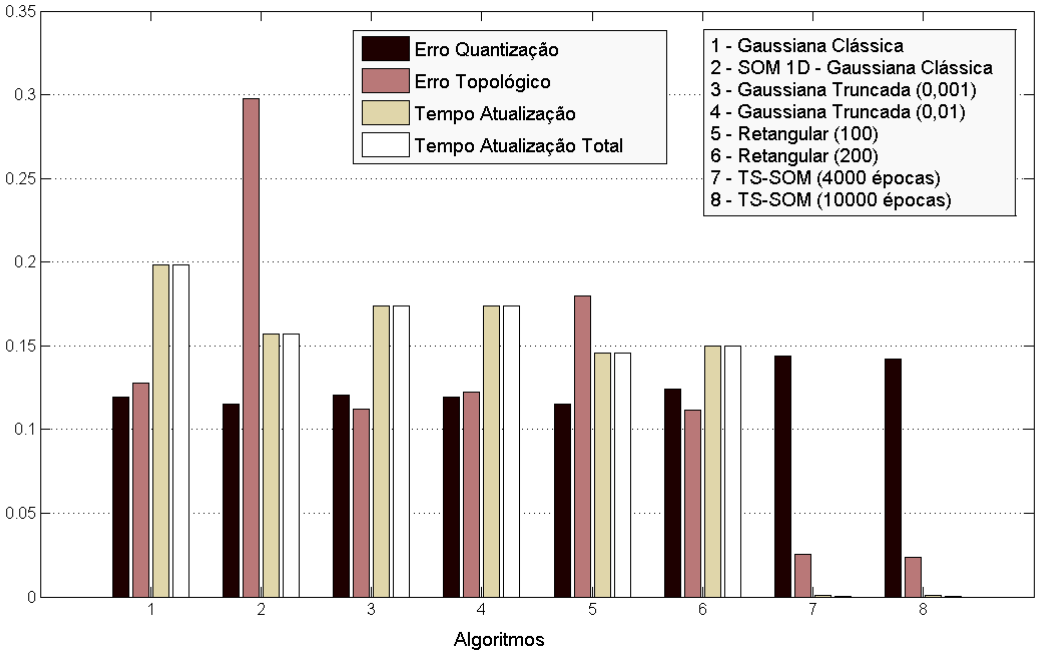


Figura 5.4: Comparação dos algoritmos na etapa de atualização.

Mesmo com o uso das funções *Gaussiana Truncada* e *Retangular* conseguiu-se diminuir consideravelmente o tempo. Isto comprova que, de fato, não há necessidade de que todos os neurônios tenham seus pesos alterados na etapa de atualização.

Muitos dos resultados observados nesta seção podem ser validados com base na teoria subjacente. Constata-se que quando se começa a utilizar um rápido decaimento na função retangular (*Retangular(100)*) há um aumento no erro topológico, decorrente do fato de somente os vizinhos imediatos terem seus pesos atualizados. Além disso, como na rede TS-SOM 1D sempre são atualizados somente três neurônios a cada vetor de entrada, o tempo de processamento dessa etapa é bem menor que os das demais abordagens.

Além dos métodos citados foi avaliada também a utilização da distância Mahalanobis. No entanto, ela não está na análise principal pois seu tempo de processamento é muito alto quando comparado com os demais algoritmos. A utilização da distância Mahalanobis como testada

aqui, dificulta utilização em sistemas embarcados com limitação de memória e processador. Dessa forma, são necessárias técnicas de otimização computacional dos cálculos das matrizes de covariância e multiplicação matricial, assunto que foge ao escopo atual deste trabalho. Para ilustrar este fato, a Tabela 5.3 ilustra o desempenho da distância Mahalanobis. Uma única matriz de covariância foi calculada usando todo o conjunto de dados. Os resultados foram extraídos de 10 simulações independentes, e os parâmetros do algoritmo são os mesmos da rede SOM 2D com treinamento convencional (Exa.+Comp.+Gaussiana), com 100 épocas de treinamento.

Tabela 5.3: Desempenho da rede SOM com uso da métrica Mahalanobis.

Método	**Erro Quantização**		**Erro Topológico**		**Tempo (ms)**		**Tempo Total (ms)**	
	Média	Desvio	Média	Desvio	Média	Desvio	Média	Desvio
Busca	1,76	$2{,}0\ 10^{-3}$	0,412	0,014	151,761	0,088	9105678,28	5296,76
Atualização	1,76	$2{,}0\ 10^{-3}$	0,412	0,014	0,091	0,001	5459,84	70,70

Observa-se na Tabela 5.3 que os erros de quantização e topológico são iguais na etapa de busca pelo neurônio vencedor e atualização pois as medidas foram retiradas de uma mesma simulação. O que de fato é interessante observar é que o uso da distância Mahalanobis no cálculo de dissimilaridade em redes SOM é extremamente custoso computacionalmente. Observa-se que o custo decorrente dessa distância vem da multiplicação entre os vetores e a matriz de covariância, uma vez que esta, bem como sua inversa, é calculada apenas uma vez no treinamento. Na etapa de atualização, o tempo de processamento é equivalente às demais abordagens, pois não é utilizada a matriz de covariância nessa etapa. Além disso, não houve redução no erro de quantização nem no erro topológico que justificasse a busca por algoritmos de otimização computacional da distância de Mahalanobis.

5.4 Conclusões

Neste capítulo foram avaliadas as técnicas de aceleração computacional da rede SOM. A avaliação foi realizada considerando-se o erro topológico, o erro de quantização e o tempo de processamento. Nessa avaliação, as redes SOM 1D apresentam piores resultados para o erro topológico, enquanto a rede TS-SOM com relação ao erro de quantização.

A arquitetura TS-SOM é, sem dúvida, a abordagem mais rápida dentre as avaliadas, permitindo uma redução de mais de 95% do tempo de execução do treinamento da rede SOM. É importante ressaltar que esta redução ocoreu principalmente devido ao uso de mapas com

muitos neurônios.

As funções vizinhança Retangular e Gaussiana Truncada apresentaram-se como uma boa alternativa à Gaussiana Convencional, devido sua aceleração computacional. Além disso, a Busca com Atalho e a Busca com Distância Parcial também apresentaram reduções significativas no tempo da etapa de busca pelo vencedor, sem comprometer o desempenho na quantização vetorial e preservação topológica. No próximo capítulo, alguns dos algoritmos avaliados nesta seção terão seus desempenhos, em uma tarefa de reconhecimento de voz, comparados.

Capítulo 6

Resultados: Reconhecimento de Voz

Neste capítulo são apresentados os resultados do estudo comparativo de desempenho entre as redes auto-organizáveis e métodos clássicos de reconhecimento de voz para identificação de dígitos e operações faladas em um dispositivo celular móvel, com o intuito de desenvolver uma aplicação de calculadora acionada pela voz. Ao longo deste capítulo são expostos o conjunto de dados utilizado para o reconhecimento de voz, a metodologia de simulação e avaliação dos algoritmos, bem como os critérios de desempenho dos classificadores.

6.1 Conjunto de Dados

Para montagem de um conjunto de dados com palavras pronunciadas isoladamente, foram capturadas elocuções de 14 pessoas (duas do sexo feminino), pronunciando cada palavra 3 vezes, variando-se a distância para o microfone. O transdutor utilizado foi um microfone para computadores pessoais da marca XPC. As palavras enunciadas consistiram nos dígitos de 0 a 9, e as palavras $\{$*vezes*, *dividido*, *limpar*, *voltar*, *resultado*, *mais*, *menos*$\}$. No total, o conjunto de dados possui 17 classes de palavras.

Para auxiliar na captura das elocuções foi desenvolvida a aplicação mostrada na Figura 6.1. Ela foi desenvolvida utilizando a linguagem de progamação Java e banco de dados MySQL. A aplicação possui quatro botões: *Limpar*, *Resultado*, *Gravar* e *Atualizar Rede*; um campo de texto; dois *checkboxes* - *Cadastrar* e *Consultar* e um quadro para exibir o sinal de voz. O botão *Limpar* esvazia o campo de texto, enquanto que o *Resultado* executa as operações existentes no campo de texto. Essa aplicação funcionou como protótipo para os testes iniciais, validando a utilização de redes auto-organizáveis no reconhecimento de voz.

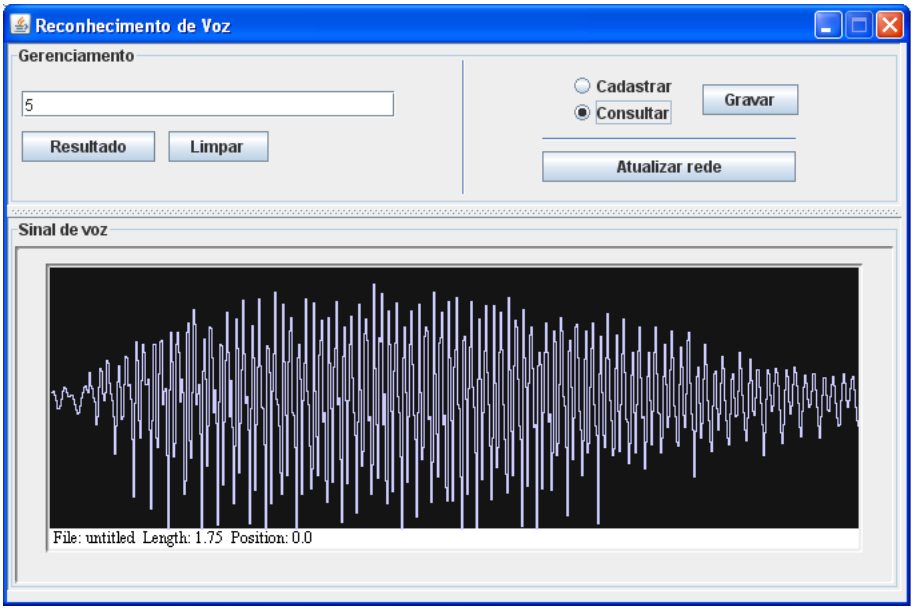


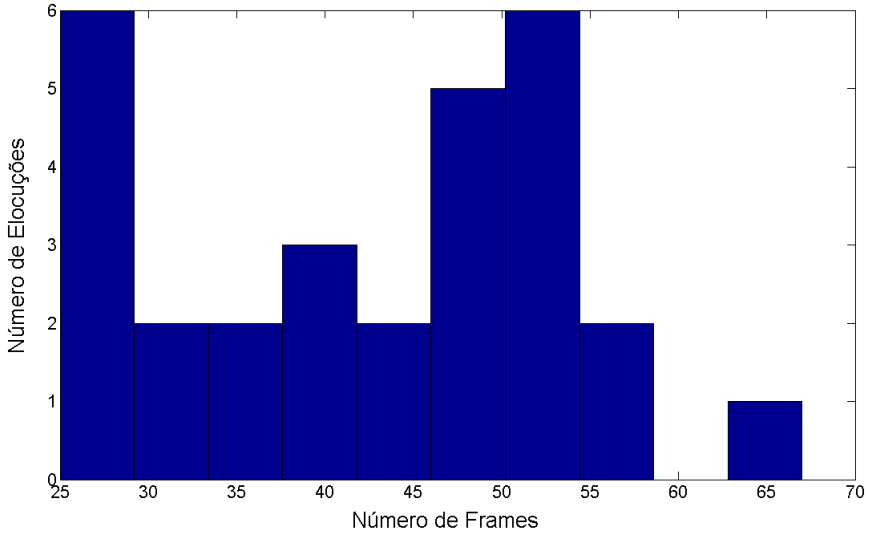
Figura 6.1: Aplicativo gráfico para auxílio na captura de elocuções.

A aplicação permite o reconhecimento de palavras isoladas. Com o *checkbox Cadastrar* selecionado, a voz gravada clicando-se no botão *Gravar* é armazenada no banco MySQL. Além disso, é necessário que o usuário escreva um rótulo para a palavra pronunciada no campo de texto, pois este será também armazenado. Para testar a aplicação, o usuário deve selecionar a opção *Consultar* e *Gravar* sua elocução. O rótulo da elocução reconhecida será inserido no campo de texto.

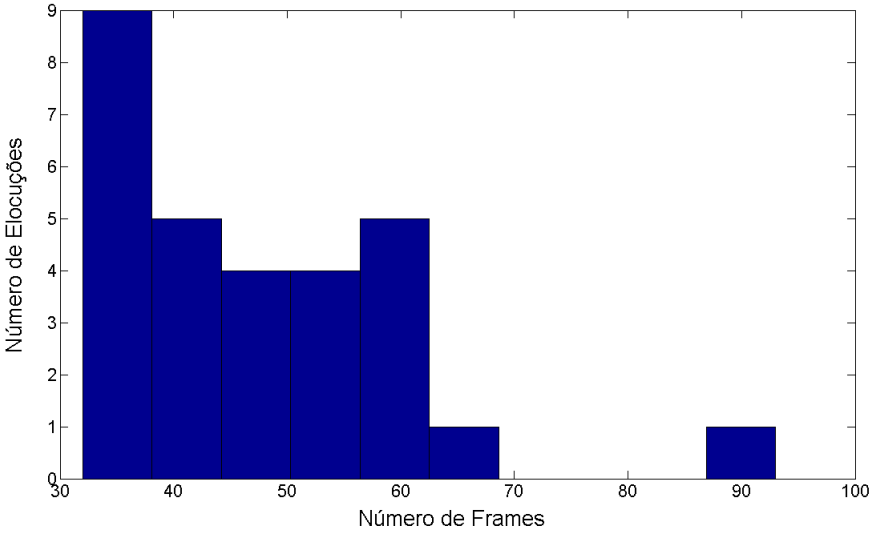
A aplicação possibilitou a gravação do sinal de voz, digitalizado com taxa de amostragem de 8KHz e 8 *bits* na quantização. Com isso, pode-se testar diferentes métodos de extração de características, bem como um ajuste fino dos parâmetros dos algoritmos relativos a essa etapa.

Todas as elocuções foram gravadas em uma sala fechada com ruído proveniente de condicionadores de ar e de forma espontânea. Isto significa que não houve um critério rigoroso para cadastro das palavras. Dessa forma, elocuções de uma mesma palavra podem diferir consideravelmente em duração e, consequentemente, em número de amostras. Para ilustrar esse problema são mostrados os histogramas do número de *frames* das elocuções das palavras *dividido* e *resultado*, nas Figuras 6.2(b) e 6.2(a), respectivamente.

Como pode ser observado na Figura 6.2, a variabilidade do número de *frames* é alta. Essa análise pode ser útil, por exemplo, para detectar *outliers*. Estes podem ser identificados como as elocuções em que a quantidade de *frames* é muito alta ou muito baixa, em função do número médio de *frames*.



(a) Palavra *dividido*.



(b) Palavra *resultado*.

Figura 6.2: Exemplo de histogramas de elocuções capturadas.

Com base no exposto, este trabalho utiliza uma base de dados com 714 elocuções, i.e. 17 palavras pronunciadas por 14 pessoas, com cada palavra sendo pronunciada 3 vezes.

6.2 Metodologia de Simulação

Para avaliar as abordagens estudadas no reconhecimento de voz em sistemas embarcados foram realizadas simulações *offline* e *online*. O termo *offline* se refere ao fato de que os testes foram efetuados fora do sistema embarcado, em um computador pessoal da marca *Dell* com

processador de 2,33 GHz e 1 GB de memória RAM rodando sistema operacional Windows XP. Todas as simulações *offline* foram realizadas utilizando-se a linguagem de programação Java e o ambiente de desenvolvimento *Eclipse*. O intuito da avaliação *offline* é realizar um estudo estatístico de algoritmos de reconhecimento de voz, e a partir da análise desses resultados, determinar quais as opções mais adequadas ao sistema celular móvel.

As simulações *online* referem-se àquelas realizados no próprio sistema embarcado. Para isso, foi utilizado o dispositivo celular N95 da Nokia. Os testes *online* serão apresentados no Capítulo 7.

Durante todos os testes realizados neste trabalho foi utilizado o algoritmo de recorte de Rabiner (RABINER; SAMBUR, 1975), com os limiares iguais a $A = 42$, $B = 32$ e $C = 22$, determinados experimentalmente. Quanto à etapa de extração de características, foram realizados testes com coeficientes LPC e Cepstral. O número de coeficientes variou entre 10, 15 e 20. A notação $LPCX$ denota X coeficientes LPC, enquanto $CepstralY$ denota Y coeficientes Cepstrais.

Os classificadores avaliados são a rede SOM e variantes, rede TS-SOM, rede MLP, algoritmo DTW e K-Médias. Os parâmetros de treinamento de cada algoritmo serão mostrados na discussão dos resultados de cada um deles. O desempenho de cada algoritmo é ilustrado em função do tipo de reconhecimento - dependente ou independente do locutor e do método de extração de características.

Para avaliar os classificadores, o conjunto de dados é dividido em conjunto de treino e teste. Nos testes dependentes do locutor, todas as elocuções são embaralhadas e 80% delas são atribuídas ao conjunto de treinamento, enquanto os 20% restantes serão utilizados no teste. No teste para o modo independente do locutor, são separadas as elocuções de três locutores. Estas elocuções compõem o conjunto de teste, e as elocuções restantes, o conjunto de treinamento.

A avaliação dos classificadores será feita com base nas taxas de acerto média, máxima, mínima e desvio-padrão. Além disso, o tempo total médio de treinamento e de teste são mostrados. O tempo total médio de teste representa o tempo médio gasto pelo algoritmo na classificação de uma elocução.

Além das métricas supracitadas foi definido um índice, chamado aqui de Índice I1, que considera a taxa média de acerto, o desvio-padrão e o tempo de treinamento ou teste de cada algoritmo. Este índice é definido como

$$I1 = \frac{(T_{ok} - \sigma)^2}{\mathscr{T}}, \tag{6.1}$$

em que T_{ok} é a taxa média de acerto, σ é o desvio-padrão da taxa de acerto e $\mathscr{T}$ é o tempo de

treinamento ou teste. É importante ressaltar que o índice só leva em consideração o tempo de treinamento ou o de teste. Dessa forma, é possível traçar um paralelo das melhores técnicas para treinamento e teste no próprio sistema embarcado.

Outro ponto relevante é que o índice I1 pode privilegiar algumas abordagens. Isto acontece quando a ordem da variável tempo é muito menor ou muito maior do que a da taxa de acerto. Assim, ele é mais adequado na comparação de técnicas que possuam ou taxas de acerto próximas ou tempos próximos.

6.3 Resultados do Desempenho *Offline*

Nesta seção são discutidos os resultados da análise do desempenho *offline* obtidos para os algoritmos DTW, K-Médias, rede SOM e TS-SOM na aplicação de reconhecimento de dígitos e operações.

6.3.1 Alinhamento Temporal Dinâmico - DTW

Os testes com o algoritmo DTW foram realizados utilizando-se apenas uma referência (*template*) para cada classe de palavra. Todos os algoritmos podem ser identificados por meio do tipo de método de extração de característica e da abordagem para geração da referência (Casual ou MDT).

Na Tabela 6.1 é mostrado o desempenho do algoritmo DTW com testes na forma dependente do locutor. Os resultados foram extraídos de 20 simulações independentes.

O melhor desempenho classificatório médio foi atingido com o uso de 20 coeficientes Cepstrais e abordagem Casual para geração de referências (*Cepstral20-Casual*). Este alcançou uma taxa média de 91,1% e máxima de 95,7%. Ele ainda gastou em média cerca de 4s para classificar uma elocução. Observa-se que o tempo de treinamento dos algoritmos que utilizam a abordagem *Casual* são muito pequenos. Isto acontece porque não há treinamento, somente uma atribuição das elocuções separadas para treino ao conjunto de referências. Com relação aos desvios-padrão, todos ficaram na mesma ordem, variando de 2,43 a 5,14%. Além disso, como era de se esperar, o tempo de teste aumenta à medida que número de coeficientes cresce.

Tabela 6.1: Resultados para algoritmo DTW - Teste Dependente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
Cepstral10-Casual	89,8	93,6	82,9	3,02	**0,0012**	3.266,43
Cepstral15-Casual	90,8	**95,7**	**87,2**	**2,43**	**0,0012**	3.809,20
Cepstral20-Casual	**91,1**	**95,7**	83,7	2,86	**0,0012**	4.312,58
LPC10-Casual	79,9	84,4	75,9	3,72	0,0014	3.293,29
LPC15-Casual	72,9	76,6	67,4	3,41	0,0013	3.874,83
LPC20-Casual	69,9	76,6	62,4	4,89	0,0014	4.352,28
Cepstral10-MDT	46,9	55,3	35,5	5,14	120.692,82	**112,05**
Cepstral15-MDT	48,3	53,2	42,5	3,36	144.105,47	143,22
Cepstral20-MDT	44,6	51,8	38,3	4,40	174.836,63	169,48
LPC10-MDT	29,9	32,6	26,2	2,54	119.074,33	178,16
LPC15-MDT	27,0	33,3	21,9	3,36	146.225,74	215,55
LPC20-MDT	25,4	32,6	17,7	4,36	170.391,49	249,39

Em geral, o uso da abordagem MDT provocou uma queda da classificação média em cerca de 40%. Essa abordagem reduz consideravelmente o tempo de teste de uma elocução, da ordem de unidades de segundos para a ordem de centenas de milissegundos. Isso acontece porque no algoritmo MDT é gerada apenas uma referência para cada classe, enquanto que na abordagem Casual o número de referências é igual ao do conjunto de dados de treinamento. Embora exista uma redução no tempo de teste, o tempo de treinamento é alto, da ordem de centenas de segundos, inviabilizando o treinamento no próprio celular; isso sem levar em conta o custo de armazenamento dessas referências. Além disso, o desempenho classificatório do algoritmo DTW utilizando MDT não é aceitável. Assim, a Figura 6.3 ilustra os índices I1 dos classificadores que utilizaram abordagem Casual.

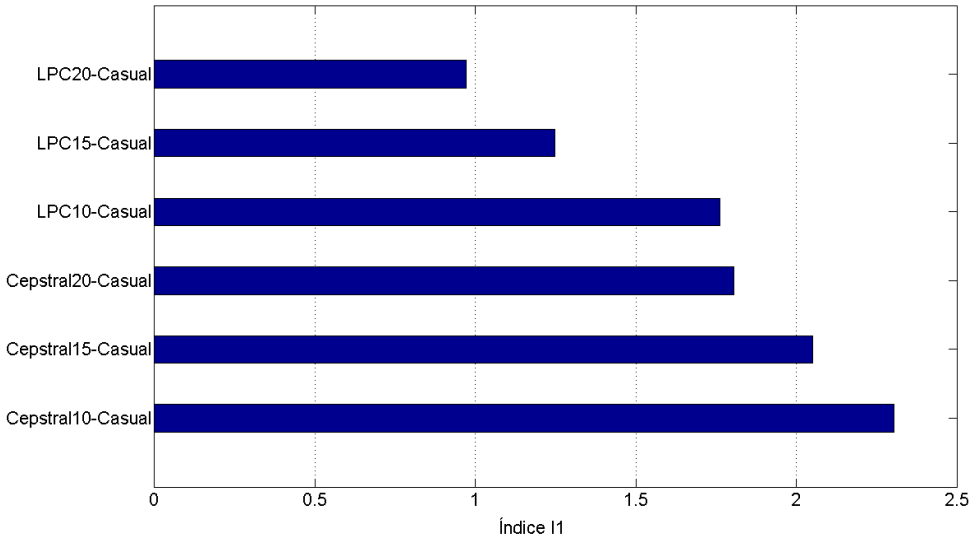


Figura 6.3: Índices I1 do algoritmo DTW (Casual) - Caso Dependente do Locutor.

Conforme pode ser visto na Figura 6.3, o melhor algoritmo sugerido pelo índice I1 foi o *Cepstral10-Casual*. Observa-se ainda que as abordagens que utilizaram coeficientes LPC obtiveram os piores resultados. Mesmo o menor tempo gasto com a utilização de apenas 10 coeficientes LPC não propiciou um índice maior que com a utilização de 20 coeficientes Cepstrais. A utilização de coeficientes LPC reduz cerca de 10% a taxa média de acerto, em comparação com o uso de coeficientes Cepstrais.

A Tabela 6.2 expõe os resultados obtidos com o algoritmo DTW para o caso independente do locutor.

Tabela 6.2: Resultados para algoritmo DTW - Teste Independente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
Cepstral10-Casual	**72,3**	**80,4**	**62,8**	4,92	0,0014	3.190,16
Cepstral15-Casual	70,9	78,4	61,5	4,78	**0,0013**	3.776,73
Cepstral20-Casual	67,2	73,8	56,1	4,91	0,0014	4.314,08
LPC10-Casual	37,9	50,9	31,7	7,05	0,0015	3.236,82
LPC15-Casual	34,8	42,5	22,9	7,10	0,0015	3.752,70
LPC20-Casual	28,9	38,6	18,4	6,62	0,0015	3.953,20
Cepstral10-MDT	46,6	56,7	35,9	7,08	116.492,60	**117,33**
Cepstral15-MDT	43,2	48,4	31,7	5,80	140.462,14	144,21
Cepstral20-MDT	44,3	51,6	34,6	4,64	163.625,31	170,24
LPC10-MDT	27,8	31,6	25,2	**2,54**	118.765,45	175,12
LPC15-MDT	24,1	33,1	20,3	3,36	145.692,74	217,98
LPC20-MDT	22,9	33,9	16,9	4,67	167.682,23	258,34

Os melhores resultados médios de acerto são 72,3% e 70,9%, obtidos com coeficientes *Cepstral10* e *Cepstral15*, e abordagem *Casual*. Um fato importante é que no caso independente do locutor, o uso dos coeficientes LPC resultou em taxas médias de acerto menores até do que as obtidas com uso da abordagem MDT, ao contrário do que ocorreu no caso dependente no locutor. Observa-se ainda, analisando os resultados do algoritmo DTW como um todo, um decréscimo de aproximadamente 20% na taxa de acerto para testes independentes do locutor.

Com relação ao tempo de treino e teste, não houve variação significativa para o caso dependente do locutor. Assim, as abordagens em que a dimensão dos vetores que compõem uma elocução são maiores, fornecem tempos de processamento maiores.

6.3.2 Rede MLP

Os parâmetros escolhidos da rede MLP, cujo os resultados do reconhecimento são apresentados nesta seção, são os seguintes: taxa de aprendizagem de 0,01, 50 neurônios ocultos, 1 camada oculta e função de ativação sigmóide logística. A saída foi codificada na forma binária, em que somente um bit é igual a 1, representando uma determinada classe, e todos os outros são iguais a zero. Assim, a rede possui 17 neurônios de saídas.

Na avaliação da rede MLP foram testados quatro métodos de normalização do sinal de voz. O termo normalização refere-se ao fato de se produzir para as diferentes elocuções, um mesmo número de coeficientes LPC ou Cepstrais. Será utilizada a notação *Normalização1* para denotar aquela com tamanho da janela e superposição variáveis. *Normalização2* representa a normalização com janela e espaçamento variáveis. *Normalização3* é aquela em que é definida um tamanho de janela, e o espaçamento e superposição são variáveis. Por fim, a notação *Quantização* denota o uso da normalização por meio de um quantizador vetorial. Nesta abordagem foi utilizado o algoritmo K-Médias com 50 protótipos e treinado durante 400 épocas. As técnicas de normalização supracitadas foram descritas na Seção 2.4.1 e têm suas implementações discutidas no Apêndice A.

A Tabela 6.3 exibe os resultados obtidos com o uso da rede MLP e coeficientes LPC.

Tabela 6.3: Resultados para a rede MLP e Coeficientes LPC - Teste Dependente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
10atr+Normalizacao1	65,5	72,4	57,1	4,82	67.457,60	0,22
15atr+Normalizacao1	60,1	68,3	51,0	5,49	99.199,67	0.32
20atr+Normalizacao1	55,3	66,3	45,9	6,26	126.973,51	0,40
10atr+Normalizacao2	53,9	61,5	47,4	3,61	89.623,36	0.21
15atr+Normalizacao2	48,7	54,0	42,9	3,64	132.008,88	0,31
20atr+Normalizacao2	45,7	51,1	37,0	4,43	174.433,66	0,41
10atr+Normalizacao3	**68,9**	73,0	**65,9**	**2,31**	93.691,38	0,21
15atr+Normalizacao3	64,0	70,9	58,9	3,60	137.900,26	0.30
20atr+Normalizacao3	62,3	66,7	56,0	3,53	181.465,93	0,43
10atr+Quantizacao	58,9	**73,7**	44,7	7,56	**21.582,00**	**0,18**
15atr+Quantizacao	51,8	64,0	47,6	6,47	29.764,51	0,25
20atr+Quantizacao	47,1	55,3	34,2	6,00	41.666,94	0,34

A maior taxa de acerto média foi obtida com uso de 10 coeficientes LPC e a técnica *Normalizacao3*. O pior resultado foi alcançado com *20atr+Quantizacao*, com acerto de 47,1%. Observa-se que ao contrário do que aconteceu com o algoritmo DTW, o aumento do número de coeficientes não resultou em aumento da taxa média de acerto.

Comparando os tempos de treinamento e teste das abordagens que utilizaram o mesmo

número de coeficientes mas diferentes normalizações, percebe-se que eles diferem consideravelmente. No entanto, todas as abordagens (exceto *Quantização*) utilizam o mesmo número de *frames* por elocução. A diferença no tempo decorre do fato de que algumas normalizações são mais restritivas do que outras, ou seja, elas não conseguem normalizar certos sinais que não satisfazem as condições de normalização e, então, são descartadas.

A Tabela 6.4 mostra os resultados da rede MLP com coeficientes cepstrais para o teste dependente do locutor.

Tabela 6.4: Resultados para a rede MLP e Coeficientes Cepstrais - Teste Dependente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
10atr+Normalizacao1	75,6	85,7	71,4	4,12	54.312,21	0,18
15atr+Normalizacao1	71,2	79,6	65,3	5,28	96.061,40	0,31
20atr+Normalizacao1	69,5	75,5	65,3	**3,42**	127.074,57	0,42
10atr+Normalizacao2	67,2	73,3	56,3	5,04	89.529,54	0,21
15atr+Normalizacao2	62,8	69,3	48,9	5,97	128.874,48	0,31
20atr+Normalizacao2	58,1	66,3	39,2	6,34	171.498,67	0,40
10atr+Normalizacao3	**81,1**	**86,5**	**72,3**	4,09	93.642,47	0,21
15atr+Normalizacao3	76,6	81,3	63,5	3,96	129.309,12	0.31
20atr+Normalizacao3	72,0	79,8	61,1	4,12	175.475,33	0,41
10atr+Quantizacao	66,9	76,3	57,9	6,10	**21.533,39**	**0,18**
15atr+Quantizacao	61,5	74,0	44,7	8,73	27.155,73	0,25
20atr+Quantizacao	58,9	71,7	40,2	6,87	37.934,84	0,34

Novamente, a maior taxa de acerto médio ocorreu com uso de 10 coeficientes e da técnica *Normalizacao3*. Em geral, as técnicas que utilizaram essa normalização foram melhores. A utilização de coeficientes cepstrais resultou em um aumento de cerca de 10% na taxa de acerto. Além disso, as abordagens de quantização, embora mais rápidas, produziram as taxas de acerto mais baixas. Os testes realizados para o modo independente do locutor e coeficientes cepstrais é mostrado na Tabela 6.5.

Tabela 6.5: Resultados para rede MLP e Coeficientes Cepstrais - Teste Independente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
10atr+Normalizacao1	66,8	76,7	54,5	6,91	61.993,67	0,21
15atr+Normalizacao1	63,6	71,8	52,3	6,72	92.497,12	0,31
20atr+Normalizacao1	58,8	74,2	51,9	6,99	124.503,43	0,41
10atr+Normalizacao2	56,2	66,4	40,5	9,65	87.662,28	0,21
15atr+Normalizacao2	57,8	65,3	48,9	5,47	128.970,84	0,30
20atr+Normalizacao2	50,8	61,6	38,9	5,85	171.177,83	0,40
10atr+Normalizacao3	70,5	**77,7**	**63,4**	5,02	91.861,45	0,21
15atr+Normalizacao3	**70,6**	74,5	61,5	**3,77**	135.302,52	0,31
20atr+Normalizacao3	68,0	74,5	59,5	5,15	177.955,33	0,41
10atr+Quantizacao	60,5	70,0	49,2	7,06	**19.095,90**	**0,18**
15atr+Quantizacao	52,0	66,0	36,9	9,73	27.446,65	0,26
20atr+Quantizacao	51,9	65,8	35,2	9,49	37.181,84	0,34

Como se pode observar comparando as Tabelas 6.5 e 6.4, as taxas de acerto das técnicas quando aplicadas ao reconhecimento independente do locutor diminuem entre 5 e 15%. Os algoritmos com melhores desempenhos médios quanto à taxa de acerto foram *15atr+Normalizacao3* e *10atr+Normalizacao3*. A Tabela 6.6 apresenta os resultados com uso de coeficientes LPC e teste independente do locutor. Observa-se que os resultados obedeceram a mesma tendência encontrada para os testes com as redes MLP, com a maior taxa obtida com o algoritmo *10atr+Normalizacao3* e, as piores, com as abordagens que utilizam a técnica *Quantização*.

Tabela 6.6: Resultados para a rede MLP e Coeficientes LPC - Teste Independente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
10atr+Normalizacao1	51,5	63,6	42,0	7,33	62.872,67	0,21
15atr+Normalizacao1	42,2	51,8	36,0	5,43	90.935,30	0,30
20atr+Normalizacao1	40,8	48,9	32,7	4,76	126.626,12	0,41
10atr+Normalizacao2	42,6	46,9	34,5	**3,46**	88.064,97	0,21
15atr+Normalizacao2	40,7	47,4	33,1	3,88	128.724.78	0,31
20atr+Normalizacao2	34,7	38,9	28,3	2,87	170.852,88	0,41
10atr+Normalizacao3	**59,5**	**65,3**	**52,9**	3,53	93.039,51	0,21
15atr+Normalizacao3	55,4	60,8	49,7	3,63	136.191,57	0,31
20atr+Normalizacao3	45,2	52,9	40,5	3,83	178.849,04	0,43
10atr+Quantizacao	45,9	57,1	29,6	10,84	**19.367,08**	**0,17**
15atr+Quantizacao	44,2	58,0	31,6	9,53	27.798,55	0,26
20atr+Quantizacao	37,5	52,0	32,3	5,59	36.294,55	0,37

6.3.3 Abordagens Baseadas em Quantização Vetorial

Esta seção compara o desempenho de algoritmos de quantização vetorial no reconhecimento de voz. Em outras palavras, é associado um quantizador vetorial (K-Médias, rede SOM ou rede TS-SOM) a cada classe de elocuções. A fase de treinamento consiste em separar do conjunto de dados somente elocuções de uma determinada classe e, então, treinar os quantizadores com esses dados. Assim, os quantizadores se especializam em uma determinada classe. Na fase de identificação de uma nova elocução (teste), a palavra reconhecida é aquela associada ao quantizador que fornecer o menor erro de quantização para a elocução. A seguir são expostos os resultados do algoritmo K-Médias, e das redes SOM e TS-SOM no reconhecimento de voz.

K-Médias Sequencial

Esta seção exibe os resultados obtidos com a utilização do algoritmo K-Médias Sequencial descrito em Cruz (2007). O algoritmo foi treinado por 400 épocas. Os testes foram realizados utilizando 10 e 15 coeficientes Cepstrais.

O algoritmo K-Médias pode ter seus protótipos iniciados de forma aleatória ou amostrados

do próprio cojunto de dados de treinamento. A estas abordagens dá-se o nome de inicialização aleatória e com os dados, respectivamente. Na Tabela 6.7 são mostrados os valores das taxas de acerto média, máxima, mínima e desvio-padrão para um total de 20 simulações e 400 épocas de treinamento. Estes resultados foram alcançados utilizando-se 10 coeficientes Cepstrais.

Tabela 6.7: Resultados para o algoritmo K-Médias - Inicialização Aleatória e com Dados.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio
K=100–Aleatória	40,2	50,3	31,2	4,75
K=100–Dados	86,3	92,9	79,4	3,48
K=256–Aleatória	40,5	52,5	34,0	4,23
K=200–Dados	88,4	92,9	82,3	2,31

Conforme pode ser visto na Tabela 6.7, a inicialização com amostras extraídas do conjunto de treinamento produz taxas de acerto cerca de 45% mais altas do que usando inicialização aleatória. Diante disso, todos os algoritmos K-Médias avaliados neste trabalho terão seus protótipos iniciados com amostras extraídas do conjunto de treinamento. A Tabela 6.8 mostra os resultados obtidos nos testes com o algoritmo K-Médias no modo dependente do locutor.

Tabela 6.8: Resultados para algoritmo K-Médias - Teste Dependente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
K=50-Cepstral10	82,8	87,9	74,5	3,82	**57.607,63**	**2,20**
K=50-Cepstral15	83,6	86,5	79,4	**1,99**	67.333,22	2,71
K=100-Cepstral10	86,3	92,9	79,4	3,48	90.308,87	4,07
K=100-Cepstral15	88,6	**94,3**	81,6	3,33	108.791,83	5,18
K=200-Cepstral10	88,4	92,9	82,3	2,31	154.029,93	8,12
K=200-Cepstral15	**90,1**	**94,3**	**85,1**	2,36	189.384,96	9,87

Observa-se na Tabela 6.8 que a taxa de acerto cresce em função do número de protótipos e de atributos. A maior taxa média de acerto foi obtida com 200 protótipos e 15 atributos Cepstrais (90,1%). Além disso, os desvios-padrão estiveram entre 1,99 e 3,82%.

Com relação ao tempo de treinamento, os valores crescem à medida que se aumenta o número de protótipos e número de atributos, o mesmo ocorre com o tempo de teste. O tempo de processamento no treinamento é da ordem de dezenas a centenas de segundos, ou seja, é inviável

treinar o algoritmo K-Médias, da forma como descrita aqui, em um celular. No entanto, o teste é muito rápido (unidades de milissegundos), o que indica a possibilidade de embarcar a rede já treinada no celular. A Figura 6.4 ilustra os índices I1 do algoritmo K-Médias para o caso dependente do locutor.

Observa-se que, segundo o índice I1, o melhor algoritmo é o K-Médias com $K = 50$ e com 10 coeficientes Cepstrais. Isto aconteceu porque o tempo de teste desse algoritmo (2,20 ms) é cerca de 4 vezes menor que o do algoritmo que utiliza $K = 200$ e 15 coeficientes (9,87 ms). É importante ressaltar que não há garantias de que essa seja a melhor escolha para o sistema, pois se a diferença de tempo entre as duas abordagens não for perceptível para o usuário, a redução computacional com uso de $K = 50$, não justificaria a perda de quase 8% na taxa média de acerto da aplicação.

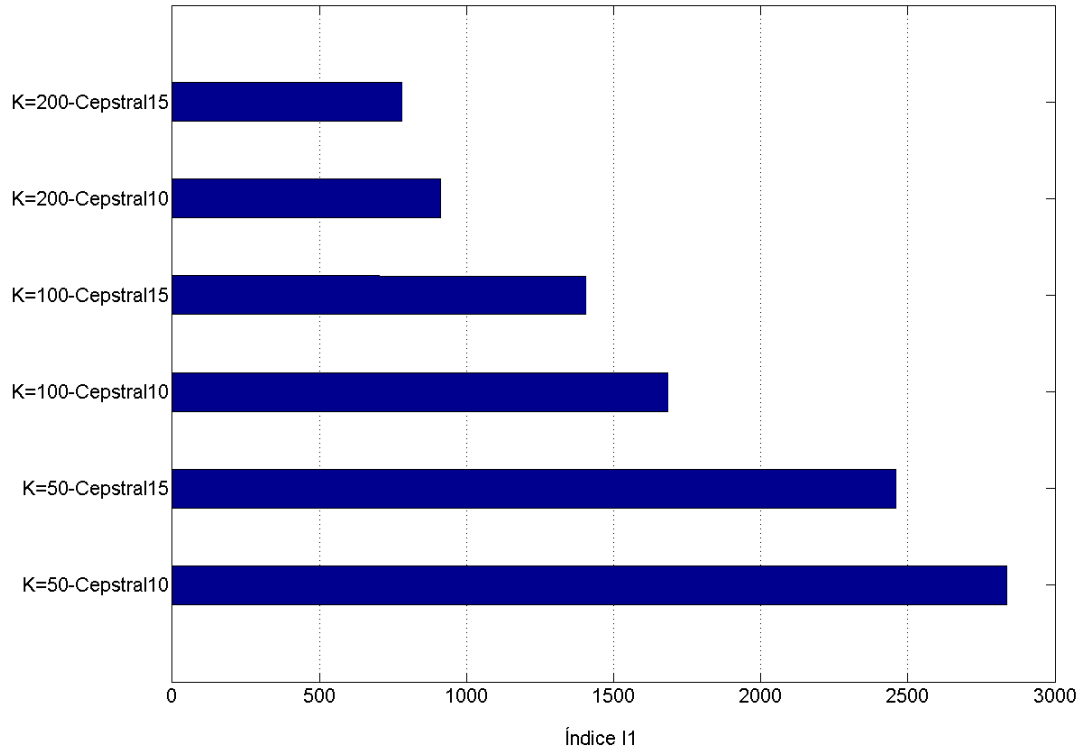


Figura 6.4: Índices I1 do Algoritmo K-Médias - Caso Dependente do Locutor.

A Tabela 6.9 mostra os resultados para os testes no modo independente do locutor.

Tabela 6.9: Resultados para algoritmo K-Médias - Teste Independente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
K=50-Cepstral10	65,2	73,8	54,7	5,02	**56.714,27**	**2,15**
K=50-Cepstral15	66,7	75,8	61,4	4,05	66.943,83	2,76
K=100-Cepstral10	66,9	**77,8**	57,5	4,87	86.653,62	4,39
K=100-Cepstral15	67,1	75,2	55,4	4,77	106.635,91	5,07
K=200-Cepstral10	67,5	75,8	59,4	5,11	151.065,77	8,06
K=200-Cepstral15	**69,5**	76,5	**62,7**	**3,78**	183.211,72	10,54

No caso independente do locutor, as taxas médias de acerto variaram menos entre si do que no caso dependente do locutor. Um indício disso é que a taxa máxima foi obtida em um algoritmo diferente da maior taxa média. O melhor desempenho classificatório (69,5%), foi alcançado com $K = 200$ e com 15 coeficientes Cepstrais. Além disso, os tempos estão coerentes com o caso dependente do locutor. A Figura 6.5 ilustra os índices I1 para o caso independente do locutor.

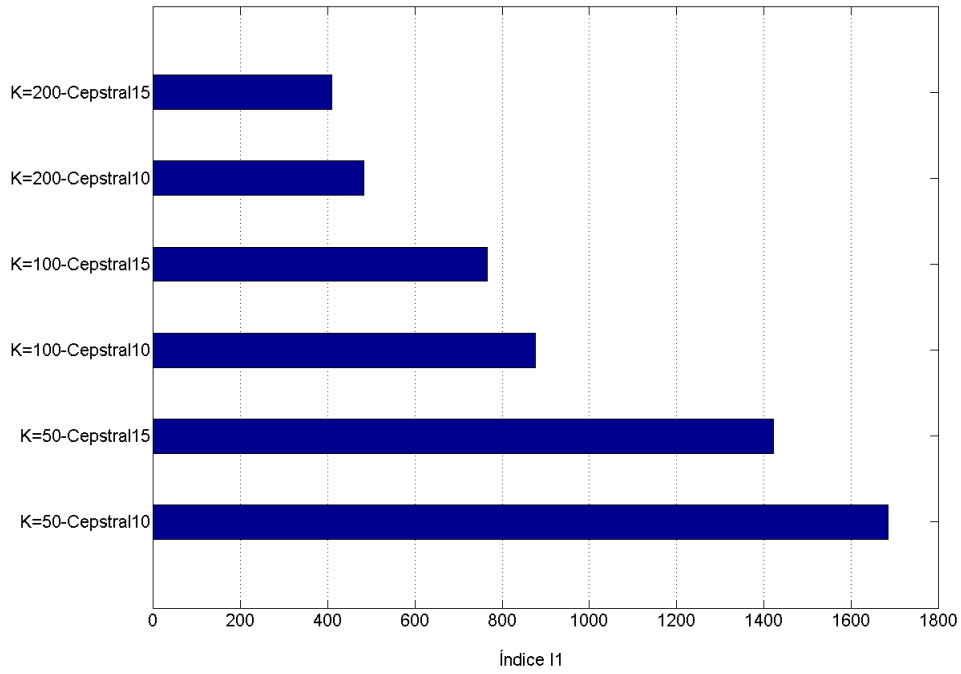


Figura 6.5: Valores do índice I1 para o algoritmo K-Médias (teste independente do locutor).

Observa-se que os índices seguem a mesma tendência daqueles obtidos para o caso dependente do locutor. O algoritmo com maior índice, e também o mais rápido, foi o *K=50-Cepstral10*. Uma vez que a variação da taxa média de acerto é de aproximadamente 4%, e este

valor é bem próximo dos desvios-padrão obtidos, o algoritmo *K=50-Cepstral10* pode realmente ser a melhor escolha para a aplicação embarcada.

Rede SOM

Esta subseção visa a análise do desempenho da rede SOM, e de suas técnicas de aceleração computacional, em uma aplicação real de reconhecimento de voz, uma vez que o Capítulo 5 tratou da avaliação com dados gerados artificialmente. Para simplificar a descrição dos algoritmos, será usada a seguinte notação.

- Gaussiana - Função vizinhança gaussiana com $\vartheta_o = 30$ e $\vartheta_\varepsilon = 0,001$.
- Retangular(200) - Função vizinhança retangular com decaimento linear do raio de ativação da função vizinhança, de $\vartheta_o = 30$ a $\vartheta_{200} = 1$ em 200 épocas de treinamento.
- Truncada(0,001) - Função vizinhança gaussiana truncada, com limite em $0,001$.
- PDS - Busca com Distância Parcial.
- Atalho(200) - Busca com Atalho a partir da iteração 200.
- Exa - Abreviação para Exaustiva. Indicativo que todas as componentes dos vetores protótipos serão usadas na determinação das distâncias e da busca pelo vencedor, em contraste com o PDS.
- Comp - Abreviação para Completa. Significa que todos os neurônios da rede serão considerados na etapa de busca pelo vencedor, em contraste com a Busca com Atalho.

As redes foram treinadas por 400 épocas, todas possuem topologia bidimensional com 256 neurônios (16×16), inicialização com os dados e métrica euclidiana. Todas as redes foram avaliadas utilizando-se 10 coeficientes Cepstrais e 10 coeficientes LPC. Eles foram escolhidos com base em testes realizados preliminarmente, indicando que o número de coeficientes não afetada consideravelmente o desempenho dos algoritmos. Além disso, um número padronizado de coeficientes torna a análise do desempenho das técnicas de aceleração mais simples.

Na Tabela 6.10 estão mostrados os resultados obtidos com a rede SOM, para o caso dependente do locutor e 10 coeficientes Cepstrais.

Tabela 6.10: Resultados para a rede SOM (10 atr. Cepstrais) - Teste Dependente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
Atalho(200)+PDS+Retangular(200)	85,8	92,2	79,4	4,41	2.275.220,35	4,48
Comp+Exa+Gaussiana	87,7	92,9	82,9	3,22	2.911.873,47	8,14
Comp+PDS+Gaussiana	**89,6**	92,9	**85,1**	**2,39**	2.860.639,27	**3,97**
Comp+PDS+Retangular(200)	88,2	92,9	84,4	2,80	**2.275.061,98**	4,07
Comp+PDS+Truncada(0,001)	87,9	**93,6**	83,7	2,69	2.661.322,49	5,80

A maior taxa de acerto média foi obtida com a configuração *Comp+PDS+Gaussiana*. Considerando a taxa máxima, alcançou-se 93,6% de acerto com a configuração *Comp+PDS+Truncada(0,001)*. O pior resultado classificatório foi produzido pela rede que utilizou a Busca com Atalho. Os desvios-padrão para todas as configurações variaram de 2,39 a 4,41%.

As configurações com treinamento mais rápido foram *Atalho(200)+PDS+Retangular(200)* e *Comp+PDS+Retangular(200)*, com tempos de 2.275.220,35 ms e 2.275.061,98 ms, respectivamente. Com a configuração *Comp+PDS+Retangular(200)* conseguiu-se reduzir o tempo de processamento em cerca de 21% com relação ao pior caso (configuração *Comp+Exa+Gaussiana*).

Na Tabela 6.11 está mostrado o desempenho da rede SOM no reconhecimento de voz dependente do locutor com a utilização de coeficientes LPC.

Tabela 6.11: Resultados para a rede SOM (10 LPC) - Teste Dependente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
Atalho(200)+PDS+Retangular(200)	80,9	87,9	75,9	3,99	**2.285.042,97**	5,43
Comp+Exa+Gaussiana	**82,8**	89,4	**76,6**	3,44	2.927.405,97	8,29
Comp+PDS+Gaussiana	80,7	84,4	77,3	**2,31**	2.866.736,11	5,06
Comp+PDS+Retangular(200)	82,7	88,6	77,3	3,49	2.285.316,73	5,11
Comp+PDS+Truncada(0.001)	81,7	**90,1**	77,3	4,38	2.663.267,02	**5,04**

Com a utilização de coeficientes LPC a taxa de acerto média é cerca de 5% menor em comparação com os coeficientes Cepstrais. Foram alcançadas taxas médias de 82,8% e 82,7% com as redes *Comp+Exa+Gaussiana* e *Comp+PDS+Retangular(200)*, respectivamente. Os tempos médios de treinamento e teste foram similares aos encontrados com uso dos coeficientes cepstrais.

Na Figura 6.6 são ilustrados os índices I1 para os algoritmos avaliados no caso dependente do locutor. O tempo usado no índice foi o de treinamento.

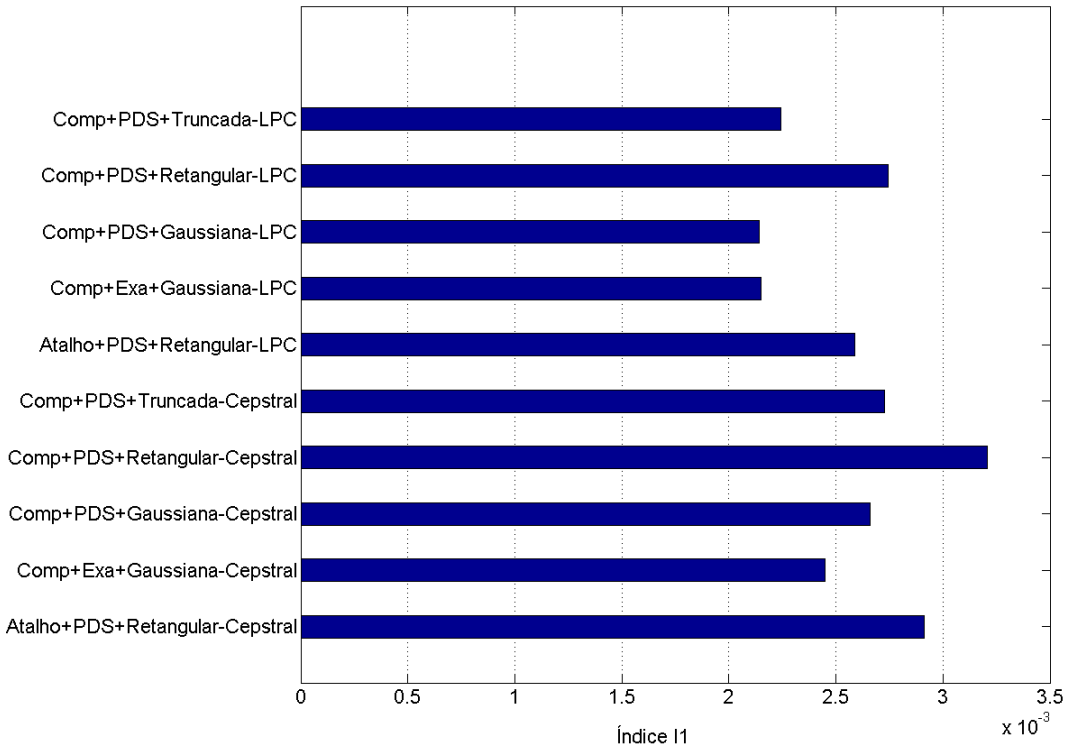


Figura 6.6: Índices I1 para rede SOM dependente do locutor - Tempo de Treinamento.

A configuração que resultou no maior índice foi *Comp+PDS+Retangular*. Além disso, as redes que utilizaram função vizinhança retangular obtiveram bons índices quando comparadas com as demais. É importante ressaltar que embora tenham sido mostrados os índices com relação ao tempo de treinamento, é notório que o treinamento das redes SOM avaliadas nesta seção não é viável de ser realizado no próprio celular. A Figura 6.7 apresenta os índices I1 com uso do tempo de teste.

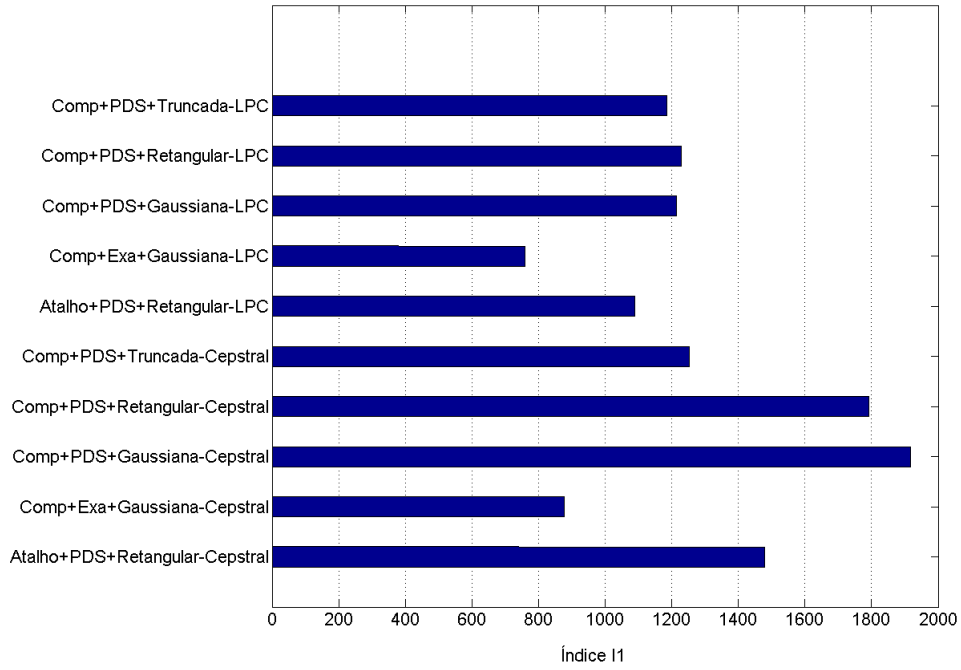


Figura 6.7: Índices I1 para rede SOM Dependente do Locutor - Tempo de teste.

Em resumo, os maiores índices foram alcançados com as configurações *Comp+PDS+Retangular* e *Comp+PDS+Gaussiana*. Observa-se ainda na Figura 6.7 que as abordagens que utilizaram PDS obtiveram os melhores resultados, visto que a função vizinhança não influencia no tempo da etapa de teste.

A Tabela 6.12 apresenta os resultados obtidos com a utilização de 10 coeficientes Cepstrais e teste independente do locutor.

Tabela 6.12: Resultados para a rede SOM (10 atr. Cepstrais) - Teste Independente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
Atalho(200)+PDS+Retangular(200)	67,9	76,5	61,4	4,34	**2.186.967,85**	4,62
Comp+Exa+Gaussiana	**73,0**	**81,7**	**66,7**	4,24	2.782.609,86	8,65
Comp+PDS+Gaussiana	71,1	75,0	64,9	3,20	2.806.355,48	4.06
Comp+PDS+Retangular(200)	71,1	78,4	64,0	4,51	2.262.188,45	**3,83**
Comp+PDS+Truncada(0.001)	70,7	**81,7**	62,2	5,17	2.612.394,09	4,02

A maior taxa de acerto foi obtida com a configuração *Comp+Exa+Gaussiana* e a menor com *Atalho(200)+PDS+Retangular(200)*. Além disso, as configurações *Comp+Exa+Gaussiana*

e *Comp+PDS+Truncada(0.001)* alcançaram taxa de acerto de 81,7%. O pior resultado dentre todas as simulações, 61,4%, ocorreu com a configuração *Atalho(200)+PDS+Retangular(200)*.

Quanto ao tempo de treinamento e teste, a tendência da redução do tempo de treinamento com a utilização da função vizinhança retangular e do tempo de teste com PDS foi mantida. A Tabela 6.13 mostra o desempenho da rede SOM com uso de coeficientes LPC e teste independente do locutor.

Tabela 6.13: Resultados para a rede SOM (10 LPC) - Teste Independente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
Atalho(200)+PDS+Retangular(200)	59,3	**71,2**	**44,7**	7.40	2.279.424,91	6,27
Comp+Exa+Gaussiana	59,5	66,0	49,3	5,99	2.851.765,19	7.96
Comp+PDS+Gaussiana	57,4	69,9	47,9	6,78	2.830.390,29	5,08
Comp+PDS+Retangular(200)	58,4	67,9	47,0	6,60	**2.274.191,92**	**4,97**
Comp+PDS+Truncada(0.001)	**61,6**	69,9	52,7	5,28	2575155,14	5,74

A utilização de coeficientes LPC provocou uma redução de cerca de 10% na taxa de acerto média, em comparação com coeficientes cepstrais. Além disso, o desvio-padrão de todos os algoritmos foi maior do que com uso dos coeficientes cepstrais. A Figura 6.8 ilustra os índices I1 para rede SOM com tempo de teste e modo independente do locutor.

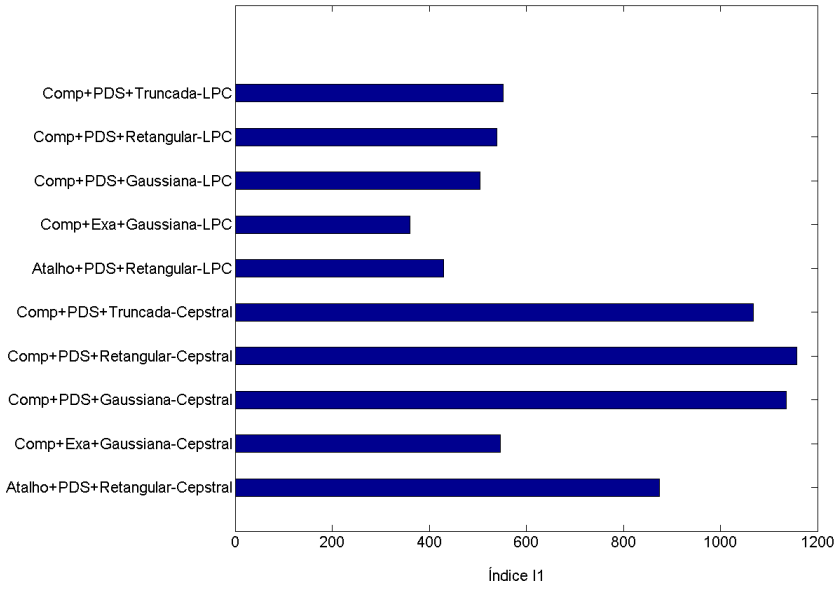


Figura 6.8: Índices I1 para rede SOM com Tempo de Teste - Caso Independente do Locutor.

Diferente do que aconteceu para o uso da rede SOM no modo dependente do locutor, a configuração com maior índice foi *Comp+PDS+Gaussiana-Cepstral*. Novamente, as redes que utilizaram coeficientes cepstrais apresentaram melhores desempenhos, pois forneceram taxas de acerto maiores sem perda significativa de desempenho.

Observa-se ainda que, em geral, a utilização da Busca com Atalho não promoveu uma redução no tempo de processamento. Isto aconteceu porque o número de dimensões dos vetores de entrada não era grande o suficiente para compensar o tempo gasto para implementar a busca, diferentemente do que o ocorreu no Capítulo 5, em que foram utilizados vetores de entrada de dimensão 50.

Rede TS-SOM

Nas simulações que se seguem, as redes TS-SOM foram implementadas como quantizadores vetoriais. Os parâmetros de treinamento desta arquitetura foram os seguintes: 9 camadas, 2 filhos, taxa de aprendizagem igual a 0,8 e 4000 épocas de treinamento. Foram comparadas arquiteturas com uso de 10, 15 e 20 coeficientes Cepstrais e LPC. A Tabela 6.14 mostra os resultados da rede TS-SOM para o caso dependente do locutor.

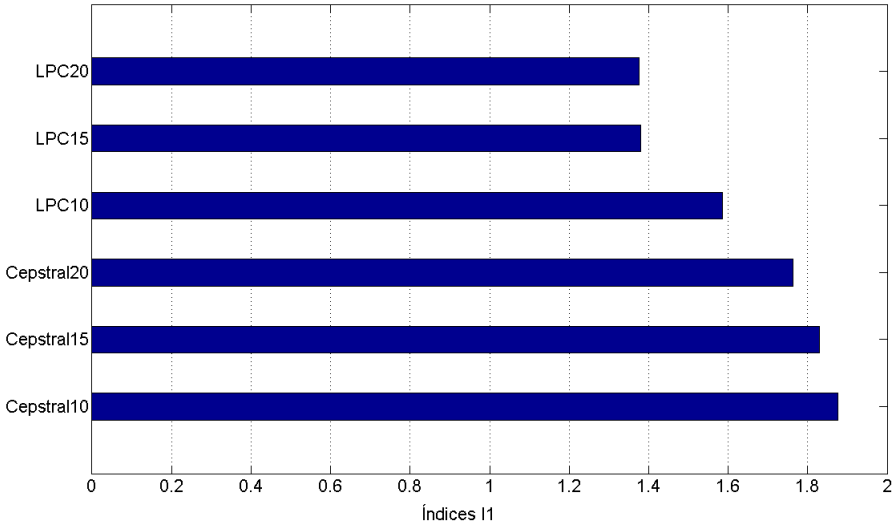
Tabela 6.14: Resultados para a rede TS-SOM - Teste Dependente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
Cepstral10	82,5	**88,6**	75,9	3,69	**3.310,01**	**7,71**
Cepstral15	82,1	85,1	**79,4**	2,16	3.492,40	8,09
Cepstral20	**84,1**	**88,6**	76,6	3,66	3.668,25	8,51
LPC10	76,2	80,1	70,9	3,33	3.349,94	7,72
LPC15	73,9	80,8	65,2	4,04	3.534,34	8,20
LPC20	74,2	76,6	68,1	2,48	3.737,43	8,60

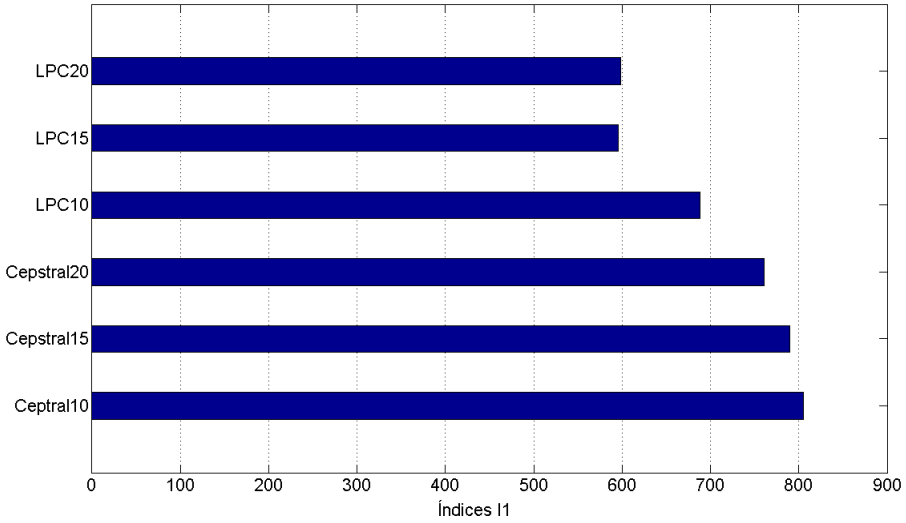
O melhor resultado em termos da taxa de acerto média foi obtido com uso da rede com 20 coeficientes Cepstrais (*Cepstral20*). A maior taxa alcançada também ocorreu com *Cepstral20*. Observa-se um crescimento de 100 a 200 ms no tempo de treinamento com o aumento de 5 coeficientes. Como era esperado, as abordagens com menor número de coeficientes foram as mais rápidas, tanto no treinamento quanto no teste. Os desvios-padrão variaram entre 2,16 e 4,04%.

A Figura 6.9 mostra os resultados dos índices I1 para a rede TS-SOM aplicadas no teste dependente do locutor. Considerando tanto o tempo de teste (Figura 6.9(a)), quanto o de treina-

mento, Figura 6.9(b), a rede *Cepstral10* obteve o maior índice I1. Esta arquitetura atingiu uma taxa máxima de acerto de 88,6%, além da classificação de uma elocução em 7,71 ms.



(a) Dependente do Locutor - Tempo de Treianamento



(b) Dependente do Locutor - Tempo de Teste.

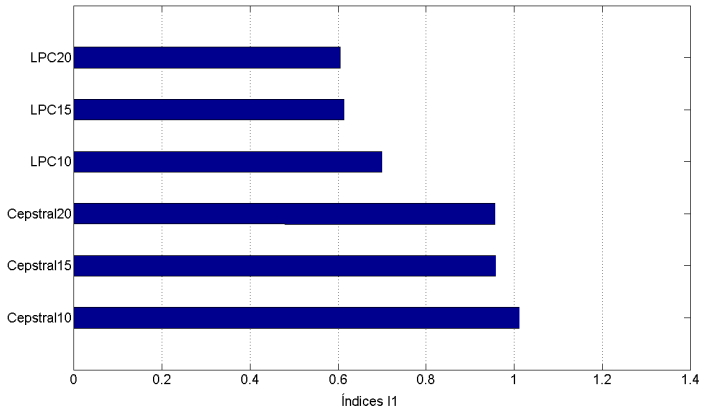
Figura 6.9: Índices I1 para a rede TS-SOM - Dependente do Locutor.

Na Tabela 6.15 são mostrados os resultados para o caso independente do locutor.

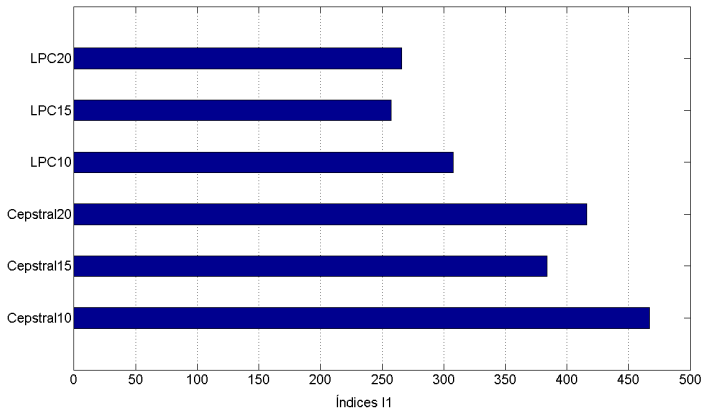
Tabela 6.15: Resultados para a rede TS-SOM - Teste Independente do Locutor.

Algoritmos	Média (%)	Máx. (%)	Mín. (%)	Desvio	Treino (ms)	Teste (ms)
Cepstral10	62,2	68,6	56,2	**4,38**	**3.307,77**	**7,16**
Cepstral15	63,1	**73,2**	**56,9**	5,36	3.482,86	8,69
Cepstral20	**64,5**	71,6	56,0	5,28	3.669,70	8,43
LPC10	54,3	64,0	43,9	5,80	3.361,834	7,65
LPC15	52,7	62,7	42,5	6,05	3.545,36	8,45
LPC20	53,4	64,9	43,8	5.96	3.724,13	8.46

O melhor resultado foi obtido com a rede que utilizou 20 coeficientes cepstrais (*Cepstral20*). Observa-se também, na mesma tabela, que a utilização de coeficientes LPC ou Cepstral não produz diferença significativa nos tempos de treinamento e teste, como era esperado, uma vez que são realizadas um número semelhante de operações.



(a) Independente do Locutor - Tempo de Treianamento.



(b) Independente do Locutor - Tempo de Teste.

Figura 6.10: Índices I1 para a rede TS-SOM - Independente do Locutor.

A Figura 6.10 apresenta os índices I1 para os testes realizados no modo independente do locutor. Observa-se que os algoritmos $Cepstral20$ e $Cepstral15$ alcançaram índices equivalentes considerando o tempo de treinamento. Em ambos os testes realizados, o uso de coeficientes cepstrais resultou em um maior índice I1.

Uma outra medida de avaliação de classificadores de padrões muito útil é a matriz de confusão. Nela é possível identificar a maior ocorrência dos erros de classificação. As matrizes de confusão de alguns dos algoritmos avaliados nesta seção são mostradas no Apêndice B.

A Tabela 6.16 resume as maiores taxas de acerto alcançadas na identificação das elocuções no modo dependente do locutor.

Tabela 6.16: Melhores Resultados - Teste Dependente do Locutor.

Algoritmos	Média (%)	Desvio	Treino (ms)	Teste (ms)
DTW-Cepstral20-Casual	91,1	2,86	0,0012	4.312,58
K-Médias (K=200-Cepstral15)	90,1	2,36	189.384,96	9,87
SOM-2D (Comp+PDS+Gaussiana+Cesptral10)	89,6	2,39	2.860.639,27	3,97
TS-SOM (Cepstral20)	84,1	3,66	3.668,25	8,51

O melhor resultado, dentre todos os algoritmos testados no modo dependente do locutor, foi obtido com o algoritmo DTW. No entanto, o tempo de teste deste algoritmo é lento, demora em média 4,31 segundos. A rede SOM-2D alcançou taxa média de 89,6%, e embora tenha o treinamento mais lento de todos, classifica uma padrão em cerca de 3,97 milissegundos. Este tempo decorre principalmente do uso do algoritmo PDS. Em uma análise inicial, a rede TS-SOM é a única com possibilidades de ser treinada e utilizada no sistema embarcado. A Tabela 6.17 mostra os melhores resultados obtidos neste trabalho para a identificação das elocuções no modo independente do locutor.

Tabela 6.17: Melhores Resultados - Teste Independente do Locutor.

Algoritmos	Média (%)	Desvio	Treino (ms)	Teste (ms)
DTW-Cepstral20-Casual	72,3	4,92	0,0014	3190,16
K-Médias (K=200-Cepstral15)	69,5	3,78	183211,72	10,54
SOM-2D (Comp+Exa+Gaussiana+Cesptral10)	73,0	4,24	2.782.609,86	8,65
TS-SOM (Cepstral20)	64,5	5,28	3.669,70	8,43

A maior taxa de acerto foi obtida com a rede SOM-2D. Observe, no entanto, que o teste

desta rede não foi muito rápido, em comparação com o uso do algoritmo PDS, pois a rede utilizada a busca Exaustiva. Observa-se também que as taxas de acerto variam mais no caso independente do locutor do que no dependente. De fato, as elocuções pronunciadas por certas pessoas (a identificação exata não foi realizada) podem prejudicar o desempenho do algoritmo quando elas são selecionadas para o teste, e nem tanto para o treinamento.

Devido aos resultados alcançados neste capítulo, será escolhida a rede SOM e K-Médias para embarque da aplicação, no que se refere ao teste. E a rede TS-SOM será avaliada quanto ao seu tempo de treinamento no próprio sistema embarcado. O capítulo a seguir reporta os resultados dos testes realizados no dispositivo celular.

Capítulo 7

Testes no Sistema Embarcado

Nesta seção são discutidos os aspectos computacionas e a simulação de uma calculadora acionada por voz no dispositivo celular Nokia N95 8GB. A seguir, são dicutidas as métricas de avaliação, o projeto da aplicação e os resultados obtidos.

7.1 Metodologia de Simulação

As simulações iniciais foram realizadas sobre emulador da SUN *Microsystems*. Uma descrição dele é apresentada no Apêndice C.

Foram escolhidos os algoritmos de quantização vetorial para embarque na aplicação, visto que com eles foram obtidos resultados aceitáveis quanto à taxa de acerto e tempo de processamento. Serão utilizadas duas abordagens. A primeira consiste em treinar o algoritmo fora do celular e então inserir os parâmetros do algoritmo, já treinado, diretamente no código. Nesta abordagem, serão avaliadas a rede SOM e o algoritmo K-Médias. No entanto, estes tornam-se equivalentes, pois a fase de determinação do vencedor é igual em ambos. Dessa forma, a primeira abordagem será avaliada por um único algoritmo competitivo com 256 e 50 protótipos e que utiliza PDS. Vale ressaltar que o que diferencia a rede SOM do algoritmo K-Médias, nesse caso, são os parâmetros (protótipos) que serão embarcados. No entanto, isto não será levado em conta aqui, uma vez que se está interessado somente no tempo de processamento. Assume-se que o desempenho da aplicação no celular seguirá aqueles obtidos nos testes *offline*, afinal essa foi a principal motivação para eles terem sido realizados.

A segunda abordagem de simulação que será avaliada envolve o treinamento no próprio celular. Assim, a aplicação deve possibilitar o armazenamento de elocuções para que elas possam ser utilizadas no treinamento. Para essa abordagem será utilizada uma rede TS-SOM com 9

camadas, 2 filhos, taxa de aprendizagem de 0,08 e 1000 épocas.

Na análise *online*, o interesse se concentra nas medições de tempo de processamento da aplicação, uma vez que uma análise *offline* do desempenho dos algoritmos já foi realizada no capítulo anterior. Com isso, são utilizados duas métricas de tempo: tempo de processamento e tempo de resposta. O primeiro representa o tempo gasto para realizar a classificação de uma elocução, que será medido pela própria aplicação. O tempo de resposta denota o tempo percebido pelo usuário desde o fim da elocução pronunciada até a identificação no dispositivo celular.

Todos os testes efetuados utilizarão 10 coeficientes cepstrais na extração de características, 8KHz de taxa de amostragem e 16 bits para quantização dos sinais de voz. Quanto ao algoritmo de recorte, foi utilizado a abordagem de Rabiner, a mesma usada nos testes *offline*. Simulações realizadas nos emuladores indicaram que os limiares encontrados nos testes *offline* funcionavam também para o celular. No entanto, o sinal no dispositivo móvel é bem mais ruidoso, o que levou à escolha de limiares diferentes. Os limiares que possibilitaram a captura do sinal recortado de voz foram $A = 2$, $B = 1$ e $C = 1$. Esta escolha equivale a utilizar somente o limiar superior de energia. Isto possibilita um processamento ainda mais rápido no dispositivo móvel, e uma abordagem similar é encontrada em Wang et al. (2008).

7.2 Descrição da Aplicação

A aplicação foi embarcada no *smartphone* N95 8GB da Nokia, pertencente à série S60. Ele possui processador ARM11, com clock de 293 MHz. A aplicação foi desenvolvida na linguagem de programação Java para dispositivos móveis (JME, *Java Micro Edition*). JME é uma plataforma que implementa a linguagem Java e é utilizado em dispositivos móveis, como celulares, *smartphones*, *Palm Tops*, *Pocket PCs*, algumas TVs de nova geração, dentre outros.

A plataforma JME é dividida em configurações (*configurations*), perfis (*profiles*) e APIs (A*pplication Programming Interface*). As configurações são responsáveis por definir um denominador comum que é suportado por uma determinada categoria de dispositivos, atualmente existem dois tipos de configuração, CDC (*Connected Device Configuration*) e CLDC (*Connected Limited Device Configuration*).

Os perfis são responsáveis pela API que é definida para uma determinada família de dispositivos. Eles são implementados sob uma determinada configuração. As simulações *online* foram realizadas utilizando CLDC-1.1 e MIDP-2.0, ambas suportadas pelo dispositivo N95 8GB.

A plataforma JME dispõe de um mecanismo de APIs que podem ser utilizadas para fins

específicos. Elas são chamadas de JSRs (*Java Specification Requests*). A principal API usada nas simulações desta seção foi a *Mobile Media API* (JSR 135). Esta permite captura do sinal de voz especificando-se o número de bits de quantização e a taxa de amostragem.

A tela inicial da aplicação é ilustrada na Figura 7.1. Nela, o usuário pode escolher entre utilizar a calculadora, cadastrar uma nova palavra, realizar o treinamento da rede TS-SOM ou obter informações sobre a aplicação através dos respectivos botões. Além disso, existem as opções de seleção nos menus inferiores da tela. A primeira opção é sair da aplicação. A segunda permite a seleção da rede neural que será utilizada no reconhecimento de voz.

(a) Menu Inicial.

(b) Opção de escolha do método de reconhecimento de voz.

Figura 7.1: Tela inicial da aplicação desenvolvida no emulador da SUN Microsystems.

A tela de cadastro de novas elocuções está ilustrada na Figura 7.2. Ela possui um campo de texto e uma listagem das palavras já cadastradas. O campo de texto possibilita ao usuário rotular a elocução que será gravada, enquanto a listagem permite conhecer as classes de elocuções que já estão cadastradas.

Figura 7.2: Tela de cadastro desenvolvida no emulador da SUN Microsystems.

O processo de reconhecimento em si ocorre na tela ilustrada na Figura 7.3. Pode-se observar que existe um campo de texto, quatro botões para os operadores matemáticos básicos (soma, subtração, divisão e multiplicação) e um outro para a igualdade. Esses botões existem porque a calculadora pode ser utilizada da maneira convencional, através do teclado alfanumérico.

Figura 7.3: Tela da calculadora desenvolvida no emulador da SUN Microsystems.

Observa-se na Figura 7.3 a presença de uma barra de progresso. Essa barra é preenchida continuamente, e ao final de cada uma delas é verificado se o usuário falou alguma palavra. Dessa forma, a barra de progresso serve como guia para o processo de pronunciação das elocuções. Assim, a calculadora fica disponível para capturar o sinal de voz a qualquer mo-

mento.

A barra de progresso demora aproximadamente 3 segundos para completar um ciclo. Para que fosse possível a captura automática da voz ao mesmo tempo em que a barra é preenchida, foram utilizadas técnicas de programação multitarefa (*multithread*).

A identificação automática da pronunciação das palavras é proporcionada pela utilização de um limiar de energia encontrado experimentalmente. O valor do limiar de energia é de 10^{10}. Assim, a cada ciclo da barra de tarefa, caso a energia do sinal obtido nesse intervalo seja maior que 10^{10}, assume-se que algo foi falado. Então efetua-se um recorte na elocução e computam-se os coeficientes cepstrais. Em seguida, estes coeficientes são quantizados pelas redes SOM ou TS-SOM, de acordo com a escolha do usuário. Por fim, o rótulo associado à rede que melhor quantizar uma elocução de teste é apresentado no campo de texto. Se o usuário pronunciar a palavra resultado, as operações existentes no campo de texto são efetuadas e o resultado é mostrado nele.

A tela Sobre, apresentada na Figura 7.4, possui um texto explicativo com informações referentes ao uso da calculadora e aos créditos da aplicação.

Figura 7.4: Tela da informações sobre a aplicação desenvolvida no emulador SUN Microsystems.

7.3 Resultados

Esta seção exibe os resultados dos testes *online*. Foram avaliadas as redes SOM/K-Médias com 256 e 50 protótipos. Além disso, na busca pelo neurônio vencedor, foi usado o algo-

ritmo PDS e métrica euclidiana. Nessa abordagem, as redes são treinadas e têm seus pesos ou protótipos embarcados no próprio código fonte da aplicação. Por fim, a rede TS-SOM foi avaliada quanto à viabilidade do seu treinamento no próprio celular.

Para avaliar as redes foram utilizadas duas métricas: tempo de processamento e tempo de resposta. O primeiro é o tempo que a aplicação demora para classificar uma elocução após a barra de progresso ter concluído um ciclo. Esse tempo é medido no próprio aparelho, através da função *System.currentTime();*. O tempo de resposta é o tempo decorrido desde o fim da elocução até a identificação da mesma, e esse tempo é medido em um cronômetro fora da aplicação. Observa-se que o tempo de resposta depende, também, do momento em que o usuário pronuncia a elocução em relação ao início do seu processamento (final da barra de progresso), além da duração da própria palavra.

Na avaliação, foram escolhidas quatro palavras, duas com tempo considerado grande (com vários fonemas), e outras duas mais curtas. As palavras são /dividido/, /resultado/, /um/ e /dois/. Elas foram pronunciadas, cada uma 10 vezes, para cada rede avaliada (K=50 e K=256), em períodos (com relação à barra de tarefa) arbitrários, não controlados. Assim, os tempos de processamento e de resposta são valores médios.

A Figura 7.5 ilustra os tempos de processamento e de resposta médios para a rede competitiva com 256 protótipos.

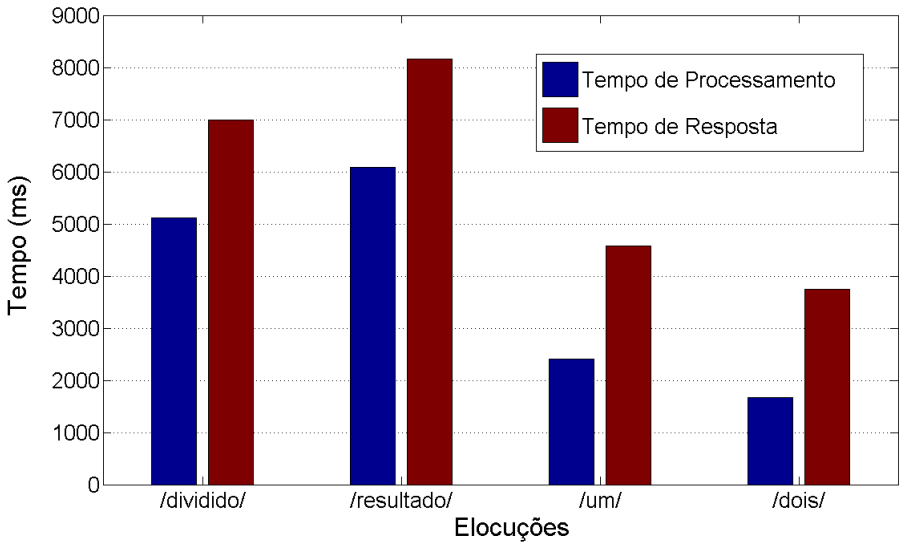


Figura 7.5: Tempo de processamento e de resposta para K=256.

O tempo médio que o usuário espera para a palavra mais duradoura, /resultado/, é de aproximadamente 8 segundos. Para a palavra /dois/, o tempo médio de espera pelo usuário é de

cerca de 4 segundos. Observa-se, ainda, que a diferença entre o tempo de processamento e de resposta varia entre 1,5 e 2,5 segundos. Essa diferença é justamente devido ao tempo que o usuário termina a pronúncia e a barra de progresso chega ao seu fim, somado ao tempo de percepção humana para medição do tempo, no cronômetro. A Figura 7.6 ilustra o desempenho do algoritmo competitivo com $K = 50$.

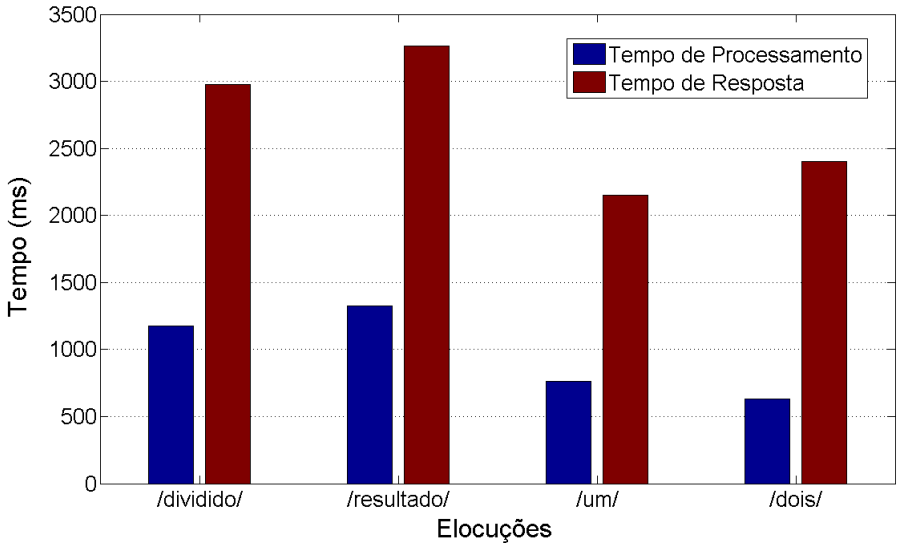


Figura 7.6: Tempo de processamento e de resposta para K=50.

Os tempos de processamento para $K = 50$ são cerca de mais de quatro vezes menores do que para $K = 256$. Com relação ao tempo de resposta, o usuário espera, em média, 3,25 segundos para a identificação da palavra mais longa.

A rede TS-SOM foi avaliada quanto ao treinamento no próprio dispositivo celular móvel. Foram realizados testes com 34 elocuções cadastradas, 2 pessoas falando cada palavra uma única vez. Assim, foram geradas 17 redes TS-SOM, uma para cada classe de palavras. O tempo médio de treinamento da rede TS-SOM foi de 14.406,0 milissegundos.

7.4 Conclusões

Por fim, este capítulo apresentou os resultados dos testes realizados no dispositivo N95 8GB da Nokia. Foi desenvolvida uma aplicação de calculadora acionada pela voz. As principais telas da aplicação são: menu inicial, tela de cadastro de novas elocuções, tela de treinamento da rede TS-SOM, tela de informações sobre a aplicação e a calculadora. Observou-se que o sinal capturado no dispositivo celular é muito ruidoso em comparação com aqueles sinais obtidos

nos testes *offline*. Alguns parâmetros dos algoritmos tiveram que ser adaptados e técnicas de programação multitarefa foram utilizadas no desenvolvimento da aplicação.

A avaliação do tempo de processamento das redes testadas comprovaram que a rede SOM pode ser utilizada no reconhecimento de voz em sistemas embarcados. A média do tempo de resposta para a elocução mais duradoura foi de 8 segundos, através do uso redes competitivas com 256 protótipos. Com a utilização de menos protótipos é possível atingir tempos de resposta de 2 segundos.

A rede TS-SOM se mostrou uma solução muito rápida nos testes *offline*. No entanto, se a rede TS-SOM realizar treinamento com um conjunto de dados grande, esse treinamento pode não ser viável para ser realizado no próprio dispositivo móvel.

Capítulo 8

Conclusões

Este trabalho apresentou um estudo e avaliação de alternativas para a utilização da rede de Kohonen em sistemas embarcados, mais especificamente na tarefa de reconhecimento de voz.

Inicialmente, foram estudados os aspectos relacionados aos sistemas de reconhecimento de voz. Conforme visto, várias técnicas foram desenvolvidas para as etapas de detecção de extremos, extração de características e classificação de padrões. Dentre elas, foram descritas e implementadas as técnicas de Rabiner & Sambur (1975) para detecção de extremos, coeficientes LPC e Cepstrais como atributos representativos para cada elocução. Além disso, foram discutidas algumas abordagens existentes para o projeto do classificador de padrões, tais como a rede MLP, a técnica clássica DTW, e abordagens baseadas em quantização vetorial, dentre as quais a rede auto-organizável de Kohonen.

Como uma das possíveis abordagens para aplicação da rede SOM no reconhecimento de voz, cada classe de elocuções é associada a uma rede SOM. Esta deve ser treinada exclusivamente com os atributos extraídos das elocuções que ela representa. Na identificação de uma nova elocução, as redes competem entre si, sendo a palavra identificada aquela associada à rede que melhor quantiza a elocução.

Em seguida, a rede SOM foi explorada quanto às métricas para cálculo de distâncias em espaços vetoriais, medidas de avaliação e custo computacional. A análise inicial indicou a atualização dos neurônios como etapa computacionalmente dominante no treinamento da rede. Isto ocorreu devido à utilização de uma função vizinhança Gaussiana sobre todos os neurônios. Ou seja, como a gaussiana é uma função assintótica, todos os neurônios são atualizados, mesmo aqueles mais distantes do vencedor, nos quais a atualização não possui grande influência. A avaliação computacional da rede SOM consiste em uma importante contribuição desse trabalho, pois essa análise não é encontrada nos livros-texto, nem na literatura científica especializada.

No Capítulo 4, as principais abordagens para acelerar o treinamento e teste da rede SOM foram descritas, além de dicas computacionais que podem ser aplicadas à rede de Kohonen. Dentre as diversas técnicas existentes, foram avaliadas a rede TS-SOM, a Busca com Distância Parcial, Busca com Atalho, além de algumas dicas para reduzir o custo computacional do treinamento/teste da rede.

No Capítulo 5, foi apresentado um estudo acerca do desempenho dos métodos para aceleração da rede SOM sobre um conjunto de dados artificial. O algoritmo TS-SOM mostrou ser uma alternativa extremamente rápida para mapas com muitos neurônios, embora obtenha erros de quantização menores, em comparação com a rede SOM convencional. Além disso, os métodos PDS e Busca com Atalho apresentaram uma redução computacional significativa, sem perda de desempenho na formação topológica da rede, nem na quantização vetorial dos dados.

As funções vizinhança alternativas, como a Retangular e Gaussiana Truncada, permitiram uma diminuição no tempo de treinamento, sem comprometer a quantização vetorial e a formação topológica da rede neural. Todas as técnicas foram estudadas visando a aplicação da rede SOM em sistemas embarcados.

O Capítulo 6 mostra os resultados obtidos com as redes SOM em comparação com outras abordagens clássicas de reconhecimento de voz, tais como a rede MLP, o algoritmo K-Médias Sequencial e o algoritmo DTW, na tarefa de reconhecimento de dígitos e operações, relativas ao desenvolvimento de uma calculadora acionada pela voz. Foram realizados testes com coeficientes Cepstrais e LPC, e testes dependente e independente do locutor. Além disso, o interesse era escolher as abordagens para implementar a calculadora no *smartphone* N95 8GB da Nokia.

Inicialmente, foram realizados testes *offline*, fora do sistema embarcado, com o intuito de avaliar as taxas de acerto e tempos de processamento dos algortimos. Na análise dos classificadores, foi definido o Índice I1, que ajudou na visualização e avaliação conjunta das taxas de acerto e tempo de processamento obtidas com os algoritmos avaliados.

Dentre as abordagens avaliadas, os melhores desempenhos foram obtidos com aquelas que utilizaram quantização vetorial. No entanto, mesmo com a utilização das técnicas de aceleração computacional da rede SOM não foi possível que fosse realizado o treinamento da rede no sistema embarcado. Uma outra contribuição dos testes *offline* foi a visualização dos desempenhos, classificatório e computacional, das técnicas de aceleração da rede SOM em uma aplicação que utiliza dados reais.

Para o caso dependente do locutor, os melhores resultados médios de classificação foram de 91,1% para DTW, 90,1% para K-Médias, 89,6% para rede SOM, 84,1% para rede TS-SOM e

81,1% para rede MLP. Já para os testes independentes do locutor, foram obtidas taxas médias de acerto de 72,3% para DTW, 69,5% para K-Médias, 73,0% para rede SOM e 70,6% para MLP. Observou-se que as maiores taxas médias de acerto foram alcançadas com uso de coeficientes Cepstrais.

As redes SOM/K-Médias e TS-SOM foram implementadas no *smartphone* N95 8GB, das quais somente a rede TS-SOM pode ser treinada no próprio celular. Foi desenvolvida uma calculadora acionada pela voz que possui basicamente quatro telas, permitindo que o usuário cadastre novas palavras, treine a rede TS-SOM, obtenha informações sobre a aplicação e utilize a calculadora acionada pela voz. Para avaliar as redes utilizadas nos testes *online*, foram definidas as métricas de tempo de processamento e tempo de resposta. No caso das palavras mais longas, a duração média de espera do usuário foi de 7,5s para a rede com $K = 256$ e de 3,25s para $K = 50$. Para as palavras com menor duração, o tempo médio é de 4s para $K = 256$ e de 2s para $K = 50$. Além disso, a rede TS-SOM com 256 protótipos leva em torno de 14 segundos para treinar um conjunto de dados com 34 elocuções, gravadas no próprio dispositivo móvel. Isso indica que para dados de treinamento com muitas amostras, o treinamento da rede TS-SOM pode não ser viável de ser realizada no dispositivo celular móvel. No entanto, ela pode ser útil em sistemas que utilizam adaptação ao locutor.

8.1 Proposta de Trabalhos Futuros

Dentre os trabalhos futuros está a avaliação de novas técnicas de detecção de extremos, principalmente do método LTSD (*Long-Term Spectral Divergence*) devido sua utilização recente em dispositivos celulares (ALHONEN et al., 2007), (OLSEN et al., 2008).

Com relação às técnicas de aceleração, pretende-se implementar os algoritmos SOTT (*Self-Organizing Topological Tree*) e o algoritmo proposto por Kusumoto & Takefuji (2006). Além disso, as métricas para cálculo de distâncias podem ser avaliadas quanto à sua influência na visualização de *clustering*, através da Matriz de distância Unificada (Matriz U) (ULTSCH; SIEMON,).

Um outro ponto importante é verificar se abordagens alternativas para utilização da rede SOM, como a utilização de LVQ (*Learning Vector Quantization*) (KOHONEN, 1997) em conjunto com DTW, propiciam melhor desempenho no reconhecimento de voz. Além disso, para tornar a aplicação comercial, é necessária uma análise do consumo de bateria da mesma, bem como a determinação de um limiar de reconhecimento, de modo que elocuções desconhecidas, pelo classificador, sejam identificadas.

Ainda como desdobramentos deste trabalho, pretende-se realizar estudos sobre reconhecimento da fala contínua. Isto inclui simulações com Modelos Ocultos de Markov, Máquinas de Vetores de Suporte (SVM, *Support Vectors Machine*) e análise de novos métodos de extração de características, tais como o uso de critérios baseados na teoria da informação (ITL, *Information Theoretic Learning*).

Referências Bibliográficas

ALHONEN, J. et al. Mandarin short message dictation on symbian series 60 mobile phones. In: *Proceedings of the 4th international conference on mobile technology, applications, and systems (Mobility '07)*. [S.l.: s.n.], 2007. p. 431–438.

ANDREAO, R. V. *Implementação em tempo real de um sistema de reconhecimento de dígitos conectados*. Dissertação (Mestrado) — Universidade Estadual de Campinas - UNICAMP, 2001.

BAKER, J. The DRAGON system - an overview. *IEEE Transactions on Acoustics, Speech and Signal Processing*, v. 23, n. 1, p. 24–29, Feb 1975.

BEI, C.-D.; GRAY, R. An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Transactions on Communications*, v. 33, n. 10, p. 1132–1133, 1985.

BELLMAN, R. *Dynamic Programming*. Princenton: Princeton University Press, 1957.

BERCHTOLD, S.; KEIM, D. A.; KRIEGEL, H.-P. The x-tree: an index structure for high-dimensional data. *Readings in multimedia computing and networking*, p. 451–462, 2001.

BUZO, A. et al. Speech coding based upon vector quantization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, v. 28, n. 5, p. 562–574, 1980.

CAMPBELL JR, J. P. Speaker recognition: A tutorial. In: *Proceedings of the IEEE*. [S.l.: s.n.], 1997. v. 85, n. 9, p. 1947–1462.

CHENG, D.-Y.; GERSHO, A. A fast codebook search algorithm for nearest-neighbor pattern matching. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '86)*. [S.l.: s.n.], 1986. v. 11, p. 265–268.

CHEUNG, E.; CONSTANTINIDES, A. Fast nearest neighbour search algorithms for self-organising map and vector quantisation. *Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers,*, p. 946–950 vol.2, 1993.

COSTA, J. A. F. *Classificação Automática e Análise de Dados por Redes Neurais Auto-Organizáveis*. Tese (Doutorado) — Universidade Estadual de Campinas, Dezembro 1999.

CRUZ, M. A. da. *Avaliação de Redes Neurais Competitivas em tarefas de Quantização Vetorial: Um Estudo Comparativo*. Dissertação (Mestrado) — Universidade Federal do Ceará, 2007.

DELLER, J. R.; HANSEN, J. H. L.; PROAKIS, J. G. *Discrete-Time Processing of Speech Signals*. [S.l.]: John Wiley & Sons, 2000.

FRIEDMAN, J. H.; BENTLEY, J. L.; FINKEL, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, v. 3, n. 3, p. 209–226, 1977.

FURUI, S. *Digital Speech Processing, Synthesis and Recognition*. 2. ed. New York: Marcel Dekker, 2001.

GRASSI, S. et al. Efficient algorithm to compute lsp parameters from 10th-order lpc coefficients. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)*. Washington, DC, USA: IEEE Computer Society, 1997. p. 1707.

HAIGH, J.; MASON, J. Robust voice activity detection using cepstral features. In: *Proceedings of the IEEE Conference on Computer, Communication, Control and Power Engineering (TENCON'93)*. [S.l.: s.n.], 1993. v. 3, p. 321–324.

HAYKIN, S. *Adaptive filter theory*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 1ª. ed. [S.l.]: Prentice Hall, 1994.

HENRIQUE, L. L. *Acústica Musical*. [S.l.]: Fundaçao Caloust Gulbenkian, 2002.

HUANG, X.; ACERO, A.; HON, H.-W. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

INDYK, P.; MOTWANI, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC '98)*. [S.l.: s.n.], 1998. p. 604–613.

ITAKURA, F. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, v. 23, n. 1, p. 67–72, 1975.

JELINEK, F. A real-time, isolated-word, speech recognition system for dictation transcription. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '85)*. [S.l.: s.n.], 1985. v. 10, p. 858–861.

JOLLIFFE, I. T. *Principal Component Analysis*. 2. ed. [S.l.]: Springer, 2002.

JUNG, C. R. et al. Implementação de veículos autônomos inteligentes. In: *XXV Congresso da Sociedade Brasileira de Computação*. [S.l.: s.n.], 2005.

KARPOV, E. *Real-Time Speaker Identification*. Dissertação (Mestrado) — University of Joensuu, Department of Computer Science, 2003.

KASKI, S. *Data Exploration Using Self-Organizing Maps*. Tese (Doutorado) — Helsinki University of Technology, 1997.

KASKI, S. Fast winner search for som-based monitoring and retrieval of high-dimensional data. In: *In Proceedings of ICANN'99, Ninth International Conference on Artificial Neural Networks*. [S.l.: s.n.], 1999. p. 940–945.

KASKI, S.; LAGUS, K. Comparing self-organizing maps. In: MALSBURG, C. von der et al. (Ed.). *Proceedings of International Conference on Artificial Neural Networks (ICANN'96),*. Berlin: Springer, 1996, (Lecture Notes in Computer Science, vol. 1112). p. 809–814.

KATAYAMA, N.; SATOH, S. The sr-tree: an index structure for high-dimensional nearest neighbor queries. In: *Proceedings of the 1997 ACM SIGMOD international conference on Management of data (SIGMOD '97)*. New York, NY, USA: [s.n.], 1997. p. 369–380.

KIVILUOTO, K. Topology preservation in self-organizing maps. In: *IEEE International Conference on Neural Networks*. [S.l.: s.n.], 1996. v. 1, p. 294–299.

KLEINBERG, J. M. Two algorithms for nearest-neighbor search in high dimensions. In: *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (STOC '97)*. [S.l.: s.n.], 1997. p. 599–608.

KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, v. 43, p. 59–69, 1982.

KOHONEN, T. K. *Self-Organizing Maps*. 2nd extended. ed. Berlin, Heidelberg: Springer-Verlag, 1997.

KOIKKALAINEN, P. Progress with the tree-structured self-organizing map. In: *European Conference on Artificial Intelligence (ECAI'94)*. [S.l.: s.n.], 1994. p. 211–215.

KOIKKALAINEN, P.; OJA, E. Self-organizing hierarchical feature maps. In: *International Joint Conference on Neural Networks (IJCNN'90)*. [S.l.: s.n.], 1990. p. 279–284 vol.2.

KUSHILEVITZ, E.; OSTROVSKY, R.; RABANI, Y. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, Society for Industrial and Applied Mathematics, v. 30, n. 2, p. 457–474, 2000.

KUSUMOTO, H.; TAKEFUJI, Y. O(logM) self-organizing map algorithm without learning of neighborhood vectors. *IEEE Transactions on Neural Networks*, v. 17, n. 6, p. 1656–1661, 2006.

LAMEL, L. et al. An improved endpoint detector for isolated word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, v. 29, n. 4, p. 777–785, 1981.

LEE, K.-F. et al. The sphinx speech recognition system. In: *International Conference on Acoustics, Speech, and Signal Processing, ICASSP-89*. [S.l.: s.n.], 1989. v. 1, p. 445–448.

LIMA, A. A. de. *Análises comparativas em sistemas de reconhecimento de voz*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, COPPE, 2000.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: Le Cam, L. M.; NEYMAN, J. (Ed.). *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, Califonia. University of California Press: [s.n.], 1967. v. 1, p. 281–297.

MAFRA, A. T. *Reconhecimento automático de locutor em modo independente de texto por Self-Organizing Maps*. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 2002.

MAKHOUL, J. Linear prediction: A tutorial review. In: *Proceedings of the IEEE*. [S.l.: s.n.], 1975. v. 63, n. 4, p. 561–580.

MAREJKA, R. Java me technology: Everything a developer needs for the mobile market. Acessado em: 20/11/2008. July 2008. Disponível em: <http://java.sun.com/developer/techinicalArticles/javame/mobilemarket/>.

MARQUES, J. S. *Reconhecimento de Padrões: Métodos estatísticos e neuronais*. 2. ed. Lisboa: IST Press, 2005.

MARTIN, A.; CHARLET, D.; MAUUARY, L. Robust speech/non-speech detection using lda applied to mfcc. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'01)*. [S.l.: s.n.], 2001. v. 1, p. 237–240.

MARTINS, J. A. *Avaliação de diferentes técnicas para reconhecimento da fala*. Tese (Doutorado) — Universidade Estadual de Campinas - UNICAMP, 1997.

MINAMIMOTO, K.; IKEDA, K.; NAKAYAMA, K. Topology analysis of data space using self-organizing feature map. In: *Proceedings of the IEEE International Conference on Neural Networks*. [S.l.: s.n.], 1995. v. 2, p. 789–794.

NISKANEN, M.; KAUPPINEN, H.; SILVAN, O. Real-time aspects of som-based visual surface inspection. In: *Proceedings SPIE Machine Vision Applications in Industrial Inspection*. [S.l.: s.n.], 2002.

OLSEN, J. et al. A decoder for large vocabulary continuous short message dictation on embedded devices. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*. [S.l.: s.n.], 2008. p. 4337–4340.

OPPENHEIM, A. V.; WILLSKY, A. S.; NAWAB, S. H. *Signals & systems (2nd ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

POLZLBAUER, G. Survey and comparison of quality measures for self-organizing maps. In: *Proceedings of the Fifth Workshop on Data Analysis (WDA'04)*. [S.l.: s.n.], 2004. p. 67–82.

PRINCIPE, J.; EULIANO, N.; LEFEBVRE, W. *Neural and Adaptative Systems: Fundamentals through simulation*. [S.l.]: John Wiley and Sons, INC, 2000.

RA, S.; KIM, J. Fast weight-ordered search algorithm for image vector quantisation. *Electronics Letters*, v. 27, n. 22, p. 2081–2083, 1991.

RABINER, L.; JUANG, B.-H. *Fundamentals of speech recognition*. [S.l.]: Prentice-Hall International, 1993.

RABINER, L.; JUANG, J. H. Historical perspective of the field of ASR/NLU. In: BENESTY, J.; SONDHI, M. M.; HUANG, Y. (Ed.). *Springer Handbook of Speech Processing*. [S.l.]: Springer, 2008. p. 301–325.

RABINER, L.; SAMBUR, M. R. An algorithm for determining the endpoints of isolated utterances. *Bell System Technical Jornal*, v. 54, p. 297–315, 1975.

RABINER, R. W. S. L. R. *Digital Processing of Speech Signals*. New Jersey: Prenctice-Hall, 1978.

RAMíREZ, J. et al. An effective subband OSF-based VAD with noise reduction for robust speech recognition. *IEEE Transactions on Speech and Audio Processing*, v. 13, n. 6, p. 1119–1129, 2005.

RAMíREZ, J. et al. Efficient voice activity detection algorithms using long-term speech information. *Speech Comunication*, v. 42, p. 271–287, 2004.

RITTER, H.; SCHULTEN, K. Kohonen's self-organizing maps: exploring their computational capabilities. *IEEE International Conference on Neural Networks*, p. 109–116, Jul 1988.

SAGHEER, A. et al. Fast competition approach using self organizing map for lip-reading applications. In: . [S.l.: s.n.], 2006. p. 3775–3782.

SAKOE, H.; CHIBA, S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, v. 26, n. 1, p. 43–49, 1978.

SARASAMMA, S.; ZHU, Q. Min-max hyperellipsoidal clustering for anomaly detection in network security. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, v. 36, n. 4, p. 887–901, 2006.

SHAFER, R. W. Homomorphic systems and cepstrum analysis of speech. In: BENESTY, J.; SONDHI, M. M.; HUANG, Y. (Ed.). *Springer Handbook of Speech Processing*. [S.l.]: Springer, 2008. p. 161–180.

SOONG, F.; JUANG, B. Line spectrum pair (lsp) and speech data compression. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'84)*. [S.l.: s.n.], 1984. v. 9, p. 37–40.

SU, M.-C.; CHANG, H.-T. Fast self-organizing feature map algorithm. *IEEE Transactions on Neural Networks*, v. 11, n. 3, p. 721–733, 2000.

TANYER, S.; OZER, H. Voice activity detection in nonstationary noise. *IEEE Transactions on Speech and Audio Processing*, v. 8, n. 4, p. 478–482, 2000.

TOKUDA, K.; KOBAYASHI, T.; IMAI, S. *Recursive calculation of mel cepstrum from LP Coefficients*. [S.l.], 1994.

TUONONEN, M.; HAUTAMäKI, R. G.; FRäNTI, P. Automatic voice activity detection in different speech applications. In: *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia*. [S.l.: s.n.], 2008. p. 1–6.

ULTSCH, A.; SIEMON, H. P. Kohonen's self organizing feature maps for exploratory data analysis. In: *Proceedings Intern. Neural Networks*. [S.l.]: Kluwer Academic Press. p. 305–308.

VENNA, J.; KASKI, S. Neighborhood preservation in nonlinear projection methods: An experimental study. In: *Proceedings of the International Conference on Artificial Neural Networks (ICANN'01)*. London, UK: Springer-Verlag, 2001. p. 485–491.

VILLMANN, T. et al. Topology preservation in self-organizing feature maps: exact definition and measurement. *IEEE Transactions on Neural Networks*, v. 8, n. 2, p. 256–266, 1997.

WANG, J.-F. et al. The design of a speech interactivity embedded module and its applications for mobile consumer devices. *IEEE Transactions on Consumer Electronics*, v. 54, n. 2, p. 870–875, 2008.

WEBER, R.; SCHEK, H.-J.; BLOTT, S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: *Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB '98)*. [S.l.: s.n.], 1998. p. 194–205.

WILPON, J.; RABINER, L. R.; BERGH, A. Speaker-independentisolated word recognition using a 129-word airline vocabulary. *The Journal of the Acoustical Society of America*, v. 72, p. 390–396, 1982.

XU, P.; CHANG, C.-H. Self-organizing topological tree. In: *International Symposium on Computer Architecture (ISCAS (5))*. [S.l.: s.n.], 2004. p. 732–735.

XU, P.; CHANG, C.-H.; PAPLINSKI, A. Self-organizing topological tree for online vector quantization and data clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, v. 35, n. 3, p. 515–526, 2005.

YOUNIS, K.; ROGERS, S.; DESIMIO, M. Vector quantization based on dynamic adjustment of mahalanobis distance. In: *Proceedings of the IEEE on National Aerospace and Electronics Conference (NAECON '96)*. [S.l.: s.n.], 1996. v. 2, p. 858–862.

ZAYKOVSKIY, D.; SCHMITT, A. Java (J2ME) front-end for distributed speech recognition. In: *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW '07)*. [S.l.: s.n.], 2007. v. 2, p. 353–357.

Apêndice A

Códigos de Normalização da Rede MLP

O Código 2 mostra a implementação, em Java, utilizada para a *Normalização1*. Nesta, não há perda da informação original. Isso acontece porque o tamanho da janela (variável *janela* no Código 2) é definida completamente somente com o número de *frames* (variável *nFrames*) e espaçamento (variável *espacamento*). Observe que não há nenhuma operação de divisão (que poderia acarretar em perda de amostras, devido ao arredondamento) na linha 4 do Código 2.

```
1  int tamanhoSinal = y.length;
2  int nFrames = 60;
3  int espacamento = 60;
4  int janela = tamanhoSinal - (nFrames - 1)*espacamento;
5  double [][] sinalJanelado = new double[nFrames][janela];
6  for(int i = 0; i < nFrames; i++){
7          int contador = espacamento*i;
8          double [] frame = new double[janela];
9          for(int z = 0; z < janela; z++, contador++){
10                 frame[z] = y[contador];
11         }
12         sinalJanelado[i] = hamming(frame);
13 }
14 return sinalJanelado;
```

Código 2: Implementação da normalização do tipo 1 - Código Java.

A implementação da *Normalização2* é ilustrada no Código 3. Nesse caso há perda de uma parte do sinal, ela pode ser observada na divisão matemática da linha 4 e é tão grande quanto maior for o resto da divisão.

```
int tamanhoSinal = y.length;
int nFrames = 60; //Constant
int superposicao = 15;
int janela = (Math.abs(tamanhoSinal - (superposicao*(nFrames - 1))))/nFrames;
double [][] signalJanelado = new double[nFrames][janela];
int espacamento;
if(janela < superposicao){
        throw new NegativeArraySizeException();
}
else{
        espacamento = janela - superposicao;
}

for(int i = 0; i < nFrames; i++){
        int contador = espacamento*i;
        double [] frame = new double[janela];
        for(int z = 0; z < janela; z++, contador++){
                frame[z] = y[contador];
        }
        sinalJanelado[i] = hamming(frame);
}
```

Código 3: Implementação da normalização do tipo 2 - Código Java.

A Normalização3 tem sua implementação mostrada no Código 4. Observe que o valor da variável superposição pode ter valores negativos. Neste caso então não existe superposição e se o valor da variável for muito alto poderá haver perda signicativa do sinal de voz.

```
int tamanhoSinal = y.length;
int nFrames = 60; //Constant
int janela = 120;
double [][] sinalJanelado = new double[nFrames][janela];
int superposicao = -(tamanhoSinal - janela*(nFrames+1))/(nFrames);
int espacamento = janela - superposicao;
for(int i = 0; i < nFrames; i++){
        int contador = espacamento*(i);
        double [] frame = new double[janela];
        for(int z = 0; z < janela; z++, contador++){
                frame[z] = y[counter];
        }
        sinalJanelado[i] = hamming(frame);
}
return sinalJanelado;
```

Código 4: Implementação da normalização do tipo 3 - Código Java.

Apêndice B

Matrizes de Confusão

Em muitos casos, o estudo do desempenho de um classificador é dado por matrizes que relacionam a classificação dada ou predita com a palavra real pronunciada. Uma matriz que permite este tipo de observação é denominada matriz de confusão. Como o objetivo é reconhecer 17 palavras da língua portuguesa, podem-se ter 17 classes, gerando uma matriz de confusão com 17 linhas e 17 colunas. Assume-se que, na disposição horizontal para a matriz, apresentam-se os dígitos reais pronunciados, e na disposição vertical a predição realizada pelo classificador. Desta forma, acertos são representados na diagonal principal da matriz.

Buscando-se gerar uma matriz de confusão representativa para um conjunto de simulações independentes, é apresentada a Matriz de confusão Média por Classe. Nesta matriz, cada célula i, j indica a quantidade de dados da classe i que foram classificados como da classe j dividido pelo número total de padrões da classe i. Isto fornece taxas quem devem ser igual a 100% ao longo de qualquer linha.

Com o intuito de simplificar a visualização das matrizes de confusão foi utilizada a seguinte notação: /r/ denota a palavra /resultado/; /v/ corresponde a /voltar/; /l/ denota /limpar/. As elocuções relativas aos algarismos são representadas pelo valor numérico. Além disso, as operações são reportadas como símbolos, i.e. /+/, /-/, /// e /*/.

Na Tabela B.1 é mostrada a matriz de confusão média por classe para o algoritmo que obteve a maior taxa acerto no reconhecimento de voz no modo dependente do locutor, *DTW (Casual20)*. Os valores iguais a 0 não são mostrados na tabela.

Tabela B.1: Matriz de Confusão Média por Classe: DTW (*Casual20*) - Teste dependente do locutor.

	/0/	**/1/**	**/2/**	**/3/**	**/4/**	**/5/**	**/6/**	**/7/**	**/8/**	**/9/**	**/+/**	**/-/**	**/*/**	**///**	**/r/**	**/l/**	**/v/**
/0/	**99,3**					0,7											
/1/		**96 ,5**	0 ,6		1 ,2	0 ,6				1 ,2							
/2/		0 ,6	**86 ,1**	0 ,6			2 ,9			1 ,2			8 ,7				
/3/			1 ,4	**83 ,9**			7 ,7						7 ,0				
/4/					**88 ,1**				1 ,7	9 ,0	0 ,6					0 ,6	
/5/						**96,2**	0,5						2,2			1,1	
/6/	1,8		1,2	13,9			**80,6**					1,2	1,2				
/7/	2 ,8		0 ,7		2 ,1	2 ,8	1 ,4	**87,9**		2 ,1							
/8/	1,2	6,1	3,0		0,6		1,8		**85,4**				0,6		0,6		0,6
/9/		3,5			0,6					**95,3**						0,6	
/+/							2,9			1,7	**94,9**		0,6				
/-/	0,6	2,5			0,6	3,2	0,6				0,6	**88,6**	3,2				
/*/			5,9	4,9			0,5					5,9	**82,7**				
///	2,7					6,7								**89,9**			0,7
/r/									0,7						**98,7**		0,7
/l/		0,6													4,3	**93,8**	1,2
/v/																0,5	**99,5**

Os dígitos mais fáceis de identificar foram 0 e voltar, com taxas de acerto de 99,3% e 99,5%, respectivamente. A principal confusão causada pelo classificador DTW está entre as palavras /seis/ e /três/. De todos os padrões pertencentes ao dígito /seis/, 13,9% foram classificados como /três/. Da mesma forma acontece com o dígito /três/, em que 7,7% foram classificados como da classe /seis/.

A Tabela B.2 mostra a matriz de confusão para a rede com melhor desempenho nos testes independentes do locutor. Observa-se que para algumas classes de palavras foram obtidas taxas altas de acerto, como /limpar/ e /resultado/, e para outras as taxas ficaram muito baixas, /um/, /vezes/, /três/ e /seis/. Esses resultados abrem a possibilidade de obtermos desempenhos altos para o caso indpendente ao locutor por meio do uso de rótulos diferentes, como /multiplicação/ ao invés de /vezes/.

Tabela B.2: Matriz de Confusão Média por Classe: SOM 2D (Comp+Exa+Gaussiana) - Teste independente do locutor.

	/0/	**/1/**	**/2/**	**/3/**	**/4/**	**/5/**	**/6/**	**/7/**	**/8/**	**/9/**	**/+/**	**/-/**	**/*/**	**///**	**/r/**	**/l/**	**/v/**
/0/	**84,4**			1,1		1,1	1,1	4,4	1,1				2,2		4,4		
/1/	3,3	**34,4**			3,3	13,3			16,7	5,6	1,1	6,7	7,8	2,2	2,2	2,2	1,1
/2/			**66,7**	2,2			1,1	5,6	4,4		3,3		4,4	1,1	11,1		
/3/			7,8	**44,4**			23,3	2,2	1,1			1,1	13,3	4,4	1,1	1,1	
/4/					**83,3**					4,4					1,1		11,1
/5/						**92,2**	1,1		3,3	1,1				1,1		1,1	
/6/			12,2	27,8		6,7	**43,3**	2,2			1,1		4,4	1,1	1,1		
/7/	1,1			3,4		2,3	10,2	**79,5**			2,3				1,1		
/8/			1,1		1,1	2,3			**93,2**					2,3			
/9/					8,2	2,4			2,4	**68,2**				1,2	1,2	3,5	12,9
/+/			1,1			2,3		6,9			**82,8**				6,9		
/-/	1,1		1,1			5,6	2,2	3,3	1,1		2,2	**76,7**	2,2	2,2	1,1	1,1	
/*/	1,1		2,2	16,9			18,0		4,5		1,1	2,2	**28,1**	12,4	13,5		
///				2,2			2,2							**95,5**			
/r/					2,2				1,1		2,2	1,1			**91,0**	2,2	
/l/										3,3					2,2	**94,4**	
/v/					14,4					1,1							**84,4**

Apêndice C

Framework Java para simulação

Segundo Marejka (2008), mais de 1 bilhão de dispositivos móveis possuem compatibilidade com aplicações Java. O *framework* Java criado para o suporte à aplicações embarcadas é o JME (*Java Micro Edition*).

JME é constituído de configurações, *profiles* e pacotes opcionais. As configurações definem um padrão mínimo de funcionalidade, mais precisamente um conjunto de bibliotecas que podem ser utilizadas com um determinado *hardware*. Este segmento pode ser visto como uma API (*Application Programming Interface*) de baixo nível, provendo as funcionalidades básicas para um determinado conjunto de dispositivos. Atualmente, existem duas destas configurações: a CLDC (*Connected Limited Device Configuration*) mais voltadas para os celulares, e a CDC (*Connected Device Configuration*) para dispositivos como maior capacidade de processamento, geralmente estacionários, como *set-top-box* (STB).

Os *profiles* melhoram o serviço de configuração através da adição de bibliotecas para certos tipos de dispositivos. Estes *profiles* são APIs de alto nível que definem o modelo do ciclo de vida da aplicação, a interface com usuário, a persistência dos dados e o acesso às propriedades dos dispositivos.

Os pacotes opcionais representam um conjunto de classes Java que constituem uma certa API, em geral criadas para suporte a novas tecnologias. Um exemplo é a *Mobile Media API* (JSR-135) que permite a captura de voz a partir do microfone. Assim, para executar esta tarefa com sucesso, o dispositivo deve dar suporte a esta API. No entanto, existem diferenças na implementação desses padrões. Nem todos os dispositivos suportam a captura da voz no formato .wav, como os dispositivos Nokia da série S40, o que tem dificultado o desenvolvimento das aplicações para celulares.

Os testes foram realizados utilizando a *Mobile Media API*. Esta permitiu a captura do sinal de voz na codificação PCM (*Pulse Code Modulation*), sem compressão. A Figura C.1 ilustra a ferramenta WTK (*Sun Wireless ToolKit*) utilizada nos testes.

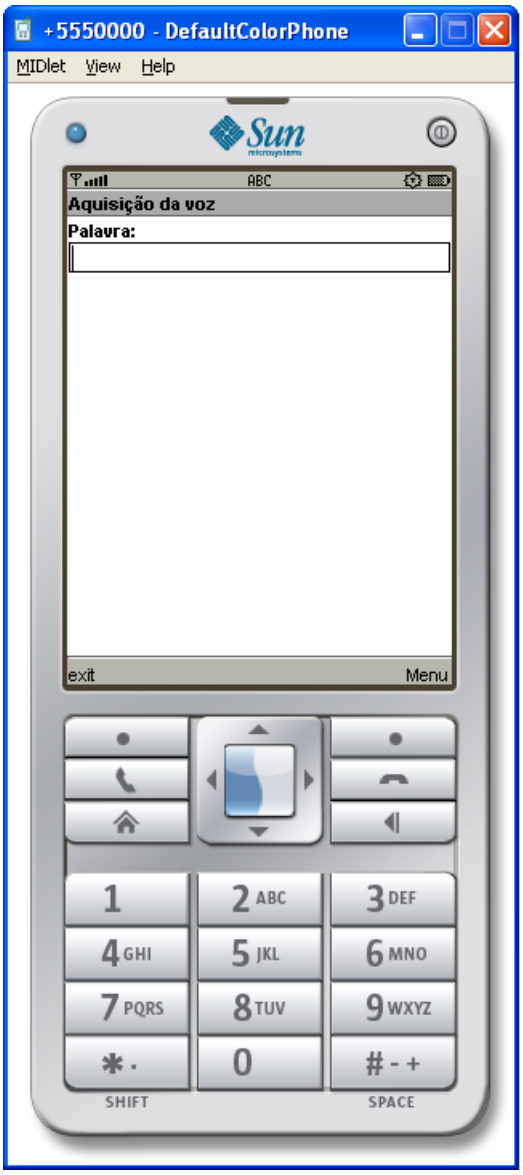


Figura C.1: Interface Gráfica do simulador WTK