UNIVERSIDADE PRESBITERIANA MACKENZIE

FERNANDO CARVALHO DE PAULA CORTES

REDES NEURAIS ARTIFICIAIS VISANDO O RECONHECIMENTO DE COMANDOS DE VOZ

São Paulo
2008

FERNANDO CARVALHO DE PAULA CORTES

REDES NEURAIS ARTIFICIAIS VISANDO O RECONHECIMENTO DE COMANDOS DE VOZ

Trabalho de Graduação Interdisciplinar apresentado ao Curso de Engenharia Elétrica, da Escola de Engenharia da Universidade Presbiteriana Mackenzie, como requisito parcial à obtenção do grau de Bacharel em Engenharia.

Orientador: Prof. Dr. Clodoaldo Aparecido de Morais Lima

São Paulo
2008

“A satisfação está no esforço e não apenas na realização final.”
(Mahatma Gandhi)

“Só falta tempo a quem não sabe aproveitá-lo.”
(Gaspar Melchor)

“Nenhum homem realmente produtivo pensa como se estivesse escrevendo uma dissertação.”
(Albert Einstein)

AGRADECIMENTOS

Ao Professor Doutor Clodoaldo Aparecido de Moraes Lima, minha eterna gratidão, pela atenção, paciência e orientação cujos conselhos e ensinamentos foram essenciais para esse trabalho.

À minha mãe por toda lamentação que teve que escutar, ao meu pai por todos os momentos em que me encorajou, ao meu irmão pela ajuda com o inglês e com o português e a minha irmã pela inspiração, meu eterno amor.

Aos meus amigos pelas discussões, risadas e incentivos.

Aos professores da graduação da Universidade Presbiteriana Mackenzie, que dividiram seu conhecimento com motivação e boa vontade.

RESUMO

Este trabalho apresenta os conceitos básicos e as técnicas envolvidas na implementação de um de um sistema de reconhecimento de voz via Redes Neurais utilizando o programa Matlab®. Inicialmente, é realizado um estudo dos fatores que dificultam o reconhecimento de voz, tais como: formação do aparelho fonador, estilo de falar e sotaque. Posteriormente, são apresentados as formas de pré-processamento e as técnicas de extração de características dos sinais de voz, essenciais no processo de treinamento de um classificador neural. Após, uma descrição detalhada da arquitetura da Rede Neural Perceptron Multicamada adotada como classificador é apresentada. Por fim, os resultados experimentais são descritos. Estes demonstram a viabilidade de uma rede neural como classificador em um sistema de reconhecimento de voz.

Palavras-chave: Reconhecimento de Voz, Rede Neural Perceptron Multicamda, extração de características, classificador, pré-processamento.

ABSTRACT

This work presents the basic concepts and techniques involved in the implementation of a speech recognition via Neural Networks using software Matlab®. Initially, it was accomplished a study of the main factors that hinder the speech recognition system, such as: the constitution of the articulatory apparatus, style of speaking and accent. Later, it was presented preprocessing method and the feature extraction techniques for speech recognition, essential elements in the process of training of a neural classifier. After, a detailed description of the architecture of Multi-Layer Perceptron Neural Network adopted as classifier was presented. Finally, the esperimental results were described. These demonstrate the viability of a Neural classifier in a speech recognition system.

Key-words: Speech Recognition System, Multi-Layer Perceptron, Neural Network, feature extraction techniques, preprocessing method.

SUMÁRIO

1 INTRODUÇÃO 10
1.1 JUSTIFICATIVA 11
1.2 OBJETIVO 12
1.3 METODOLOGIA 12
1.4 ESTRUTURA DO TRABALHO 12
1.5 ESTADO DA ARTE 13
2 O SINAL DE VOZ 16
2.1 A FORMAÇÃO DA VOZ HUMANA 16
2.2 FATORES ADVERSOS AO RECONHECIMENTO 17
2.3 MÉTODOS DE ANÁLISE DO SINAL DE VOZ 19
3 REDES NEURAIS 26
3.1 HISTÓRIA DAS REDES NEURAIS 27
3.2 ANTECEDENTE BIOLÓGICO 29
3.3 REDES NEURAIS DE MULTICAMADA – *MLP* 32
3.4 MÉTODOS DE APRENDIZADO 33
3.5 ALGORITMO DE RETRO-PROPAGAÇÃO (*BACKPROPAGATION*) 36
4 SIMULAÇÕES 40
5. CONCLUSÕES 46
5.1 TRABALHOS FUTUROS 46
REFERÊNCIAS BIBLIOGRÁFICAS 47

ANEXOS

ANEXO A – FUNÇÃO TREINAMENTO 50
ANEXO B – FUNÇÃO PERCEPTRONS 52
ANEXO C – CODIGO DE EXTRAÇÃO DAS CARACTERISTICAS 53
ANEXO D – FUNÇÃO TREINATESTA 54
ANEXO E – CODIGO DE FILTRAGEM 55
ANEXO F – FUNÇÃO GERADADOS 56
ANEXO G – FUNÇÃO CORTASILENCIO 57
ANEXO H – FUNÇÃO FILTROPASSAFAIXA 58
ANEXO I – FUNÇÃO JANELA 59
ANEXO J – FUNÇÃO MAX_PSD 60
ANEXO K – FUNÇÃO MAX_RCEPS 61
ANEXO L – FUNÇÃO MAX_XCORR 62
ANEXO M – FUNÇÃO MAX2 63

LISTA DE ILUSTRAÇÕES

Ilustração 1 – Diagrama dos órgãos usados na produção da fala. 16
Ilustração 2 – Comprimento das cordas vocais. 17
Ilustração 3 – À esquerda laringe masculina e à direita feminina. 18
Ilustração 4 – Modelo de cálculo do *Cepstrum*. 25
Ilustração 5 – Rede recorrente com neurônios ocultos 28
Ilustração 6 – Modelo do neurônio humano. 29
Ilustração 7 – Modelo não linear de um neurônio Fonte: Haykin, 2002 30
Ilustração 8 – Modelos de topologia de redes neurais. 32
Ilustração 9 – Multi–Layer Perceptron 33
Ilustração 10 – Aprendizado supervisionado 34
Ilustração 11 – Superfície de erro para uma rede MLP qualquer 37
Ilustração 12 – Exemplo de mínimos locais. 37

LISTA DE GRÁFICOS

Gráfico 1 – Sinais originais normalizados. 20
Gráfico 2 – Sinais após filtragem dos ruídos. 21
Gráfico 3 – Sinais após a retirada dos períodos de silêncio. 22
Gráfico 4 – Sinal final estudado, tempo normalizado em 25ms 22
Gráfico 5 – Densidade Espectral de Potência, PSD 23
Gráfico 6 – Autocorrelação 24
Gráfico 7 – Cepstrum das vogais ‘a’ e ‘u’ 25
Gráfico 8 – Funções de ativação 31
Gráfico 9 – Representações gráficas da palavra “amarelo” com taxa de amostragem 44,1kHz e 11,025kHz e representação utilizando 16 bits. 41
Gráfico 10 – Representação gráfica da primeira janela da palavra “amarelo” 42
Gráfico 11 - Representação gráfica dos coeficientes de AutoCorrelação, Cesptrum e PSD da janela da palavra “Amarelo”. 42

LISTA DE QUADROS

Quadro 1 – Média de erros encontrados para PSD nos treinamentos 43
Quadro 2 – Média de erros encontrados para PSD nos testes 43
Quadro 3 – Média de erros encontrados para Energia nos treinamentos 43
Quadro 4 – Média de erros encontrados para Energia nos testes 44
Quadro 5 – Média de erros encontrados para Autocorrelação nos treinamentos..... 44
Quadro 6 – Média de erros encontrados para Autocorrelação nos testes..... 44
Quadro 7 – Média de erros encontrados para Cesptrum nos treinamentos 44
Quadro 8 – Média de erros encontrados para Cesptrum nos testes 45

1 INTRODUÇÃO

O objetivo da tecnologia de reconhecimento da voz é "criar máquinas que possam receber informações através de sinais de voz e agir de forma apropriada de acordo com estas informações" (DELLER Jr.; HANSES; PROAKIS, 1993). Os sistemas de reconhecimento de voz (*Automatic Speech Recognition - ASR*) tentam fazer com que um computador reconheça todas as palavras que possam ser entendidas por qualquer pessoa, independente de parâmetros como sotaque e entonação e, preferencialmente, em tempo real. Este tem sido o alvo de intensos estudos na última década, motivados pelas inúmeras aplicações comerciais desta tecnologia. A concepção e a implementação deste tipo de ferramenta está em fase de amadurecimento, mas o problema do reconhecimento ainda está em aberto e sujeito a novas abordagens para a sua solução.

Considerando a riqueza dos aspectos cognitivos da comunicação humana falada, o reconhecimento de voz como mera transcrição das palavras parece restrito, pois ignora o real significado e a intenção do locutor ao se comunicar. A área de pesquisa, então, busca representar as intenções da comunicação de uma forma mais rica do que a simples transcrição do sinal de voz. Obviamente, esta tarefa é extremamente ambiciosa, dado que os processos do pensamento humano e intenções de comunicação são muito pouco compreendidos.

A necessidade da melhoria dos processos e da qualidade de vida incentiva a utilização dos comandos de voz, visto que o usuário pode estar livre para fazer outras tarefas, e que tal sistema pode ajudar a diminuir problemas de adaptação de pessoas com deficiências físicas, movimentando suas cadeiras de rodas e abrindo portas com um simples comando.

Sobre o reconhecimento de voz, pode-se falar de identificação ou verificação. A identificação tem como objetivo mostrar ao sistema quem é o operador, usando a voz como parâmetro biométrico e dispensando, assim, o uso de chaves, cartões magnético, códigos de acesso ou qualquer outro meio. Ao se dizer verificação, o sistema tenta identificar comandos ou alertas do operador e executa uma ação (BITTENCOURT, 2007). A identificação ou Reconhecimento Automático de Locutores (RAL) determina automaticamente o indivíduo emissor de uma determinada locução materializada num sinal de voz mediante a comparação entre

características extraídas da locução atual e locuções anteriores. Já a verificação tem como objetivo a identificação de comandos emitidos pelo locutor. Em um sinal de voz, através da comparação entre características extraídas, define-se qual o comando desejado, independentemente do locutor.

O crescente interesse no paradigma de Redes Neurais Artificiais faz com que seja importante avaliá-lo na solução de problemas de reconhecimento. Este paradigma neural tem sido estudado mais intensamente na última década, e os trabalhos publicados têm enfatizado o desenvolvimento e o aperfeiçoamento de modelos que se aproximam do funcionamento do neurônio biológico.

1.1 JUSTIFICATIVA

Desde as primeiras máquinas que chegaram ao mercado após a segunda metade do século XX com o objetivo de diminuir o trabalho intelectual do ser humano, os computadores vêm sendo aprimorados dia após dia, desafiando até mesmo os mais otimistas com uma crescente rede de aplicações. Hoje, muitas delas são voltadas para a área médica e são capazes de identificar problemas antes impossíveis, com uma diagnose precisa e a antecipação que possibilita a cura.

Na ultima década, intensificaram-se as pesquisas para o reconhecimento de voz, que, além de facilitar o dia a dia de pessoas normais no seu cotidiano e de favorecer sua segurança num mundo cada vez mais caótico, visam, sobretudo, diminuir as dificuldades de pessoas portadoras de deficiências físicas ou vítimas de acidentes com impedimento – ainda que transitório – de exercerem suas atividades mais prementes. Não é demais lembrar que as estatísticas apontam para um número cada vez maior de portadores de deficiências físicas (totais ou parciais), dado o elevado índice de acidentes – principalmente os de trânsito.

Assim, o avanço destas pesquisas representam uma importante conquista de caráter social e que, longe de configuraram-se como uma obra de ficção, poderão garantir a dignidade, o conforto e a independência que todo o ser humano almeja.

1.2 OBJETIVO

Este trabalho tem como objetivo principal estudar a instalação de uma rede neural artificial para o reconhecimento de sinais voz; também, o estudo presente tem como objetivos específicos: a) identificar e catalogar a bibliografia existente e b) estudar os métodos de extração e seleção de características para a classificação de sinais de voz. Deseja-se implementar os métodos de extração e seleção de características da voz, familiarizar-se com redes neurais artificiais do tipo *perceptron* multicamada para a classificação de sinais de voz e, por fim, comparar os métodos de seleção/extração de características.

1.3 METODOLOGIA

Inicialmente, este estudo estará direcionado para a familiarização e extração de características para sinais de voz; posteriormente, será realizado um estudo aprofundado sobre os modelos de classificação de sinais, com foco, principalmente, nos modelos baseados em Redes Neurais Artificiais; por fim, será efetuada uma análise comparativa da influência dos métodos de extração de características no desempenho da classificação desses sinais. Todas as implementações serão realizadas utilizando-se o *software* MATLAB.

1.4 ESTRUTURA DO TRABALHO

Este trabalho será estruturado em cinco capítulos. No capítulo 1, serão feitas uma introdução do tema e uma revisão bibliográfica, expondo alguns conceitos importantes para facilidade de exposição dos resultados dos capítulos seguintes; o capítulo 2 expõe os métodos de extração de características da voz; o capítulo 3 conterá a definição de Redes Neurais e uma listagem das mais importantes e suas

propriedades; o capítulo 4 apresentará as simulações para o reconhecimento de palavras isoladas, com recursos gráficos e algébricos para procurar relações e propriedades no espectro de tais sinais; finalmente, o capítulo 5 tratará das conclusões do trabalho.

1.5 ESTADO DA ARTE

Os recentes trabalhos publicados na área de redes neurais aplicadas à classificação de sinais de voz para a predição de comandos tratam das dificuldades encontradas em lidar com ruídos do ambiente e em estudar e compreender melhor estruturas biológicas da fala, auxiliando na pesquisa de novos métodos mais adequados para a classificação de padrões.

Em 2006, ANDRÉ *et al.* popuseram um sistema de reconhecimento automático de fala a ser usado no interior de um automóvel em pleno trânsito, sujeito a diversos tipos de ruídos inerentes a esta aplicação, tais como ruídos de tráfego, ruídos do motor, dos pneus e ruídos do vento e da chuva. Sobre os resultados, o mais significativo consistiu na confecção de uma Base de Dados de Fala, que permitirá a otimização dos sistemas de reconhecimento de voz em ambientes adversos, mais especificamente no interior de um automóvel. Para realizar esta otimização, foi necessário fazer um treinamento adequado, com locuções gravadas em situações reais; portanto, a base de dados confeccionada serviu perfeitamente a este propósito.

RIBAS *et al.* (2006) apresentaram um sistema de acionamento que usa comandos de voz de baixo custo e fácil implementação. Apesar de o sistema ainda estar em fase de testes, os resultados obtidos foram bastante satisfatórios. A velocidade de resposta aos comandos verbais foi razoável e a quantidade de erros de reconhecimento foi mínima, dependendo, basicamente, do tempo dedicado ao treinamento e à clareza de fala do locutor.

Em 2005, PETRY *et al.* apresentam os resultados obtidos em testes para reconhecer pessoas pela voz utilizando processamento digital de sinais. No

trabalho, também foram mostradas as técnicas utilizadas, do pré-processamento do sinal à extração de características significativas.

Na fase de pré-processamento foram utilizados filtros para eliminarem uma tendência espectral de aproximadamente -6dB/oitava, pois estas não trazem nenhuma informação significativa. Após esse filtro o sinal foi dividido em janelas.

Como parâmetros foram utilizados dois tipos de coeficientes, os *cepstrais* e os *mel-cepstrais*. Estes coeficientes foram escolhidos porque ajudam a reduzir o volume de dados e apresentam perdas mínimas de informação.

Para classificação de padrões foi utilizada uma técnica conhecida como Quantização Vetorial Multisecção. O funcionamento dessa técnica pode ser dividida em três partes: geração do *CodeBook*; quantização de um padrão desconhecido; e a medida de distorção. A etapa de geração de *CodeBook* consiste na geração dos níveis discretos que cada vetor pode assumir, esses são chamados de centróides. A quantização de um padrão é a escolha do centróide que melhor representa esse vetor; isso é feito levando em consideração a menor distância entre o vetor em questão e todos os centróides existentes no *CodeBook*. Como última etapa deste processo de classificação tem-se a comparação entre dois vetores. Esta é realizada através do cálculo de distorção entre eles. A medida de distorção mínima, ou Euclidiana, é a mais conhecida.

Para os testes deste trabalho, foi utilizada uma única palavra pronunciada por cinqüenta locutores distintos. Nos testes foi identificado que, conforme se aumentam os centróides, menor é a taxa de aceitação do locutor e maior a taxa de rejeição. Como melhor resultado, foi encontrada uma rejeição de 96% dos impostores e uma aceitação de 90% para locutores autorizados, mostrando que os coeficientes extraídos e o método de classificação são eficazes para estas tarefas.

Em 2001, Dos Santos examinou as silabas como unidades fonéticas (UF) em Sistemas de Reconhecimento de Voz Contínua (RCV) para o português. Eles mostraram que para o reconhecimento de voz em tempo contínuo, os sistemas utilizam unidades fonéticas menores que a palavra, com o intuito de diminuir a quantidade de processamento e armazenamento necessários. O trabalho apresenta como unidades mais utilizadas os trifones, mas estes são de difícil treinabilidade. Para um sistema na língua inglesa que utilize um extenso vocabulário, é necessário o treinamento de 60.000 trifones, na língua portuguesa seria necessário o dobro de

trifones. Desta forma, para grandes dicionários é mais interessante o inventario um de Santos e Alcaim, que tem uma treinabilidade mais fácil que os trifones na língua portuguesa. Porém, para dicionários pequenos e médios é ainda mais interessante o uso de sílabas. Estas não são apropriadas para a língua inglesa porque a mesma não tem divisão silábica trivial.

No artigo, foram efetuados testes em 113 modelos de sílabas, número suficiente para compor um dicionário utilizado nos sistemas RVC para ligações telefônicas automáticas. Neste o usuário pode ditar um número de telefone sem se preocupar em dizer os números na forma de digito.

Como reconhecedor, foi utilizado o Modelos de Markov Escondidos (HMM). Para o treinamento utilizou-se somente um locutor nas 20 repetições de cada palavra do dicionário. Os coeficientes utilizados foram os *Mel-Ceptrais*, os *Delta Mel-Ceptrais* e o Logaritmo da energia.

Nos resultados obtidos foi possível confirmar a eficiência das sílabas como unidades fonéticas, que produziram resultados vantajosos em sistemas de RVC baseados na língua portuguesa com pequenos e médios dicionários. Tal desempenho se mostrou equivalente ao dos sistemas de RVC que utilizam trifones, onde o treinamento exigiria o uso de aproximadamente 2000 modelos no treinamento, que contrastam aos 113 modelos utilizados para o treinamento com sílabas.

2 O SINAL DE VOZ

Neste capítulo, serão mostrados os princípios básicos da formação da voz humana, bem como os fatores que dificultam o reconhecimento e alguns dos métodos de reconhecimento utilizados.

2.1 A FORMAÇÃO DA VOZ HUMANA

A compreensão do fundamento de sistemas de reconhecimento de voz requer primeiro o entendimento de como o corpo humano produz a voz.

A fala é o resultado do esforço de várias partes do corpo: o sistema nervoso; o sistema respiratório; os músculos e os ossos, que contribuem para a eficiência desta difícil forma de se comunicar. A produção da voz se inicia com uma contração-expansão dos pulmões, criando, assim, uma diferença entre a pressão do ar nos pulmões e do ar na frente da boca. Essa pressão expele o ar, que passa pela laringe e, antes homogêneo, vai se transformando em uma série de pulsos de ar que chegam na boca e na cavidade nasal, formando um som. Ainda na boca, esses pulsos são modulados pela língua, pelos dentes e pelos lábios de forma a produzir a voz.(Ilustração 1)

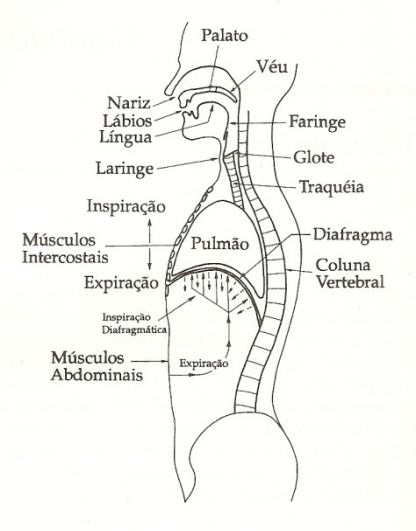

Ilustração 1 – Diagrama dos órgãos usados na produção da fala.

2.2 FATORES ADVERSOS AO RECONHECIMENTO

Na fala, muitos fatores importam, e não só a correta posição dos órgãos fonoarticulatórios. A seguir, são exemplificados alguns fatores.

2.2.1 Sotaques estrangeiros e regionais

Ao investigar as diferenças entre locutores com os métodos da análise estatística, constatou-se que os primeiros dois componentes principais da variação correspondem ao sexo (e relacionado às propriedades fisiológicas) e ao sotaque respectivamente. Certamente, comparado ao reconhecimento do discurso nativo, o desempenho degrada quando se tenta reconhecer o discurso estrangeiro (KUBALA *et al.*, 1994; LAWSON *et al.*, 2003).

2.2.2 Fisiologia do locutor

Ao lado da origem regional, outra propriedade pode ser vista na forma do aparelho fonador, que determina a escala de variação dos parâmetros da voz de um determinado locutor. As figuras abaixo mostram um bom exemplo dessas diferenças, vistas entre o aparelho fonador de homens e mulheres.

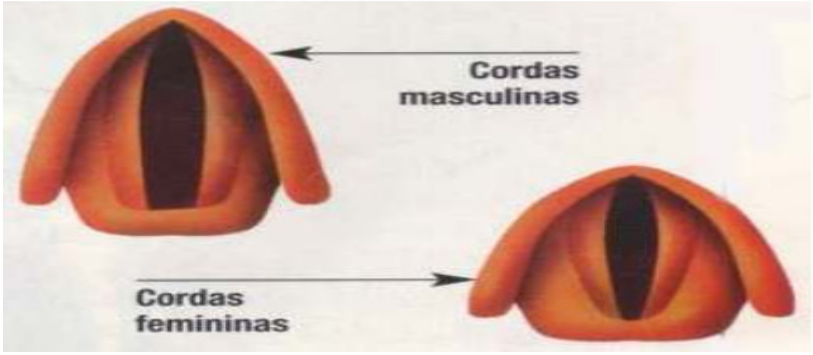

Ilustração 2 – Comprimento das cordas vocais.

Ilustração 3 – À esquerda laringe masculina e à direita feminina.

2.2.3 Estilo de falar e fala espontânea

No discurso espontâneo ou sob a pressão do tempo, a redução de pronunciações de determinados fonemas ou sílabas acontece freqüentemente. Sugeriu-se que estas pronúncias indistintas afetam mais fortemente as seções que trazem menos informações. No contraste, as parcelas do discurso onde a dificuldade é mais alta tendem a ser articuladas com mais cuidado (dado sugestões fonéticas, sintáticas e semânticas) (KUBALA *et al.*, 1994; LAWSON *et al.*, 2003).

2.2.4 Fala de crianças

O reconhecimento de discursos das crianças ainda é um problema difícil para sistemas de reconhecimentos convencionais. O discurso das crianças representa uma área importante e ainda mal compreendida no campo do reconhecimento do discurso por computadores. O primeiro problema está relacionado ao tamanho físico das crianças: elas têm um intervalo vocal mais curto e as dobras vocais mais próximas comparadas aos adultos. Isso resulta em posições mais elevadas da freqüência fundamental. A freqüência fundamental elevada é refletida em uma distância grande entre os harmônicos, tendo por resultado a definição espectral pobre de sons expressos. Um segundo problema é que algumas crianças mais novas podem não ter uma pronunciação correta, pois, às vezes, não aprenderam ainda como articular fonemas específicos corretamente. (KUBALA *et al.*, 1994; LAWSON *et al.*, 2003).

2.3 MÉTODOS DE ANÁLISE DO SINAL DE VOZ

Nesta seção, são apresentados métodos de análise de sinais de voz. Para isso, serão usados como exemplo dois sinais de voz gravados contendo respectivamente a vogal '*a'* e a vogal '*u'*. Estes sinais foram gravados com microfones de baixo custo em um micro computador, em horários diferentes, mas no mesmo local.

Os sinais, na sua forma bruta, normalmente prejudicam o reconhecimento, pois as variações no ambiente (ruídos de alta freqüência, variações nos tempos de silêncio iniciais e finais e variações no distanciamento do microfone) causam uma variação na amplitude. Para anular estas influências e facilitar o trabalho de reconhecimento, foram aplicadas funções do MATLAB. São elas: filtragem de ruídos, cortes de períodos de silêncio e normalização da amplitude.

Para eliminar as variações de amplitude entre sinais, geradas por gravações diferentes, é necessário normalizar o sinal em sua amplitude, tendo, assim, sempre um sinal com amplitudes entre 1 e -1. Para isso, foi desenvolvida uma rotina que divide os valores das amostras pelo maior valor de amplitude encontrado no sinal. (Gráfico 1)

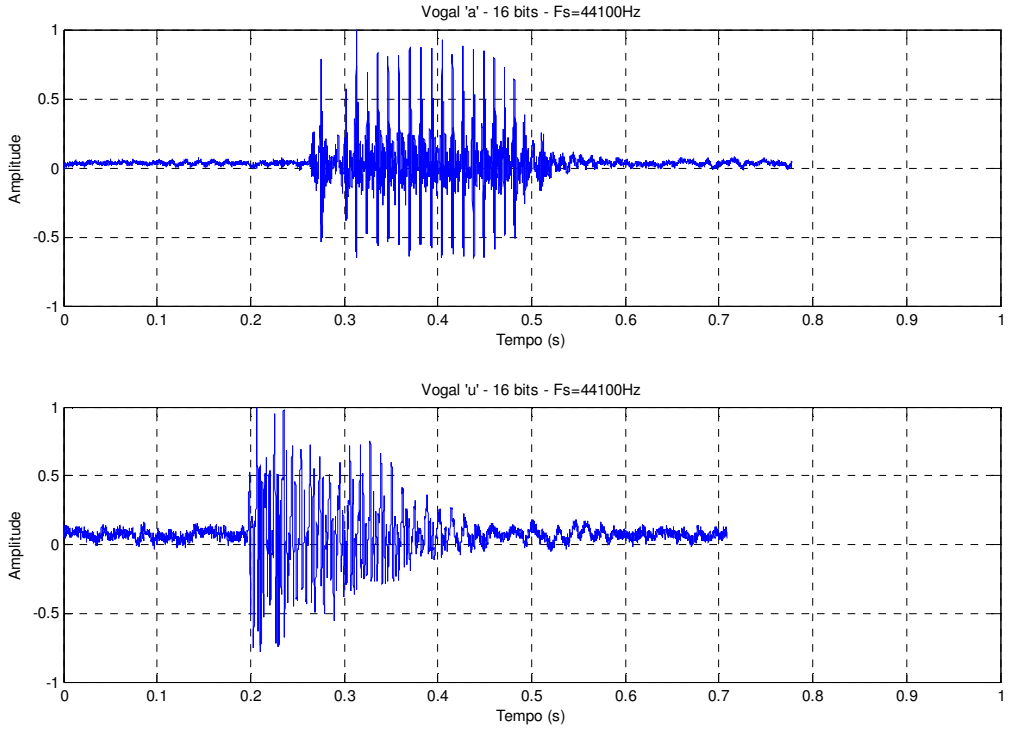

Gráfico 1 – Sinais originais normalizados.

Para a filtragem de ruídos que normalmente são originados no ambiente, usamos um filtro *Butterworth* de quinta ordem, do tipo passa-faixa, com freqüências de corte de 300Hz e 4500Hz, pois a linguagem falada compreende sons nesta faixa de freqüência (FOLMER & JOHNSON, 1968). (Gráfico 2)

A taxa de amostragem, foi reduzida a 11025Hz, para diminuir o esforço computacional envolvido nas simulações. Tal redução é permitida pois ainda atende as necessidades do teorema de Nyquist, que diz que a freqüência de amostragem tem que ser maior ou igual a duas vezes a freqüência amostrada para que possamos recuperar sem perdas. Assim com uma taxa de 11025 amostras por segundo, sabemos que o sinal conservara todas as qualidades necessárias para a simulação.

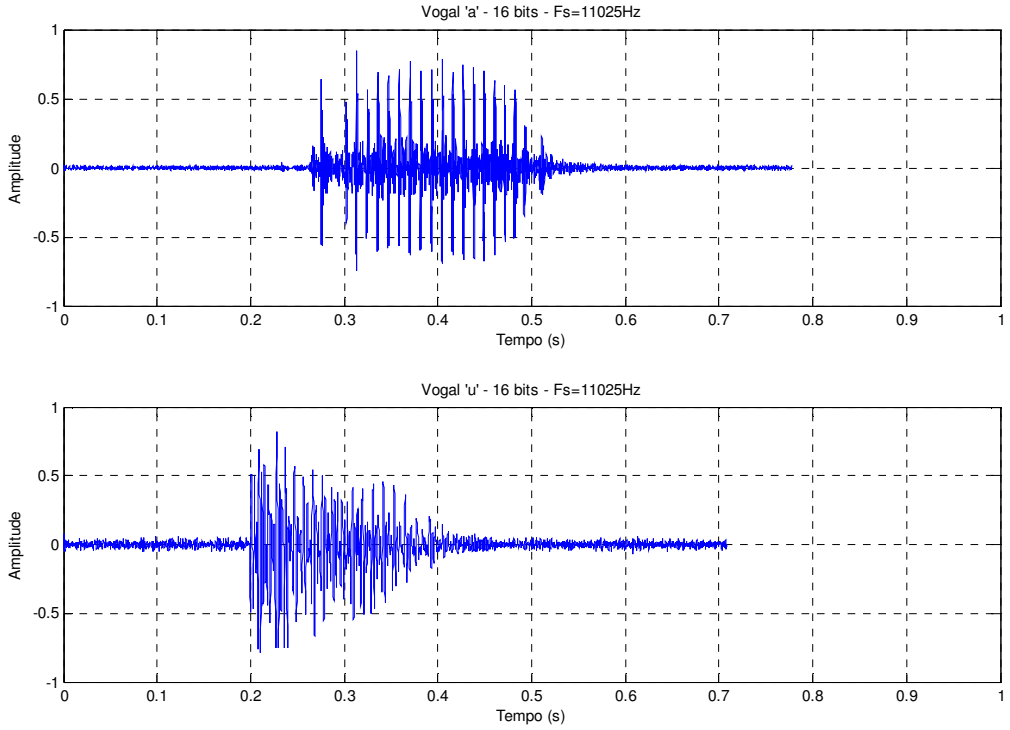

Gráfico 2 – Sinais após filtragem dos ruídos.

Períodos de silêncio antes e depois do sinal não possuem nenhuma informação para o reconhecimento, mas podem conter ruídos e sinais indesejados. A duração destes períodos é muito variada; portanto, é preciso retirar estes períodos para analisar o sinal. Para esta função, construiu-se uma rotina que elimina os intervalos de silêncio do sinal de acordo com parâmetros ajustáveis. Nesta função, há duas entradas: o sinal (x) e um valor de limiar (xlim). Com esses dados, é possível calcular a amplitude média do sinal (xm) e o número de amostras (n). Com essas informações, a rotina procura a primeira amostra que tenha uma amplitude maior que o limiar indicado (xlim); assim que a rotina encontra esta amostra, ela começa a calcular a média das amostras subseqüentes até que essa média ultrapasse a média da amplitude do sinal todo (xm). Quando isso acontece, a rotina descarta toda informação anterior àquele ponto. Este algoritmo foi construído dessa forma para otimizar a eliminação dos períodos sem informação, visto que, se apenas o valor limítrofe (xlim) fosse utilizado, correr-se-ia o risco de obter-se períodos de silêncio devido à extensão de ruídos como o início do sinal. O mesmo

método é utilizado para retirar o silêncio do final do sinal, simplesmente analisando o sinal de trás para frente (LEMOS, D. R., *et al.*, 2004). (Gráfico 3)

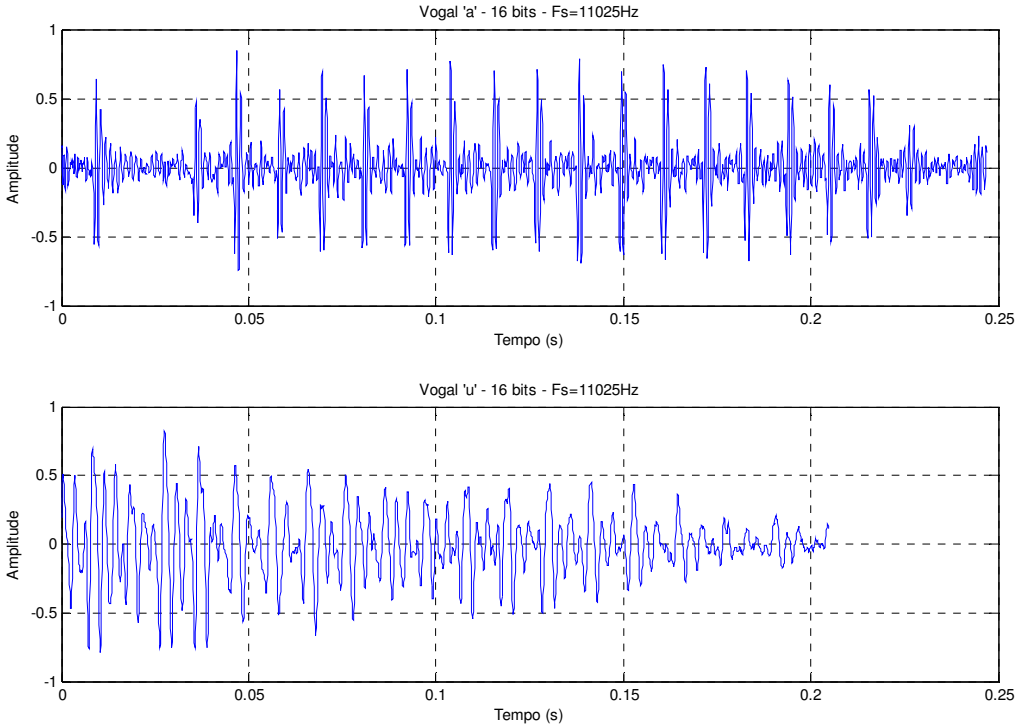

Gráfico 3 – Sinais após a retirada dos períodos de silêncio.

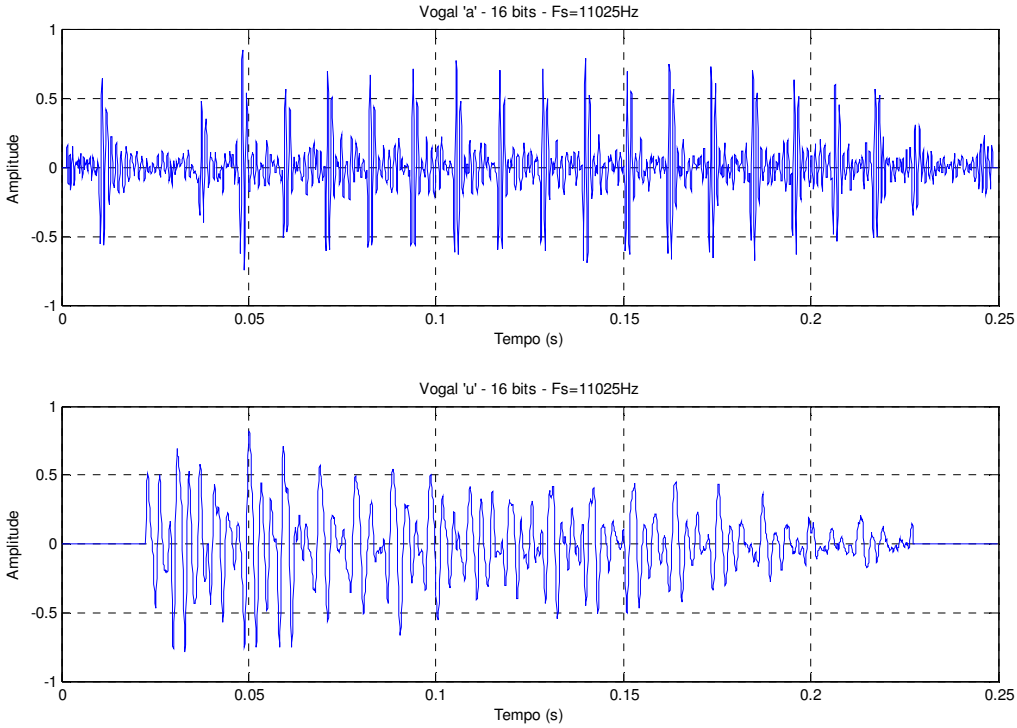

Gráfico 4 – Sinal final estudado, tempo normalizado em 25ms.

Para que um computador possa reconhecer as palavras, utilizaram-se coeficientes que possuem semelhança entre locuções diferentes. Estes coeficientes, ainda, devem ser de fácil extração, não devem variar no tempo, da saúde ou do estado emocional do locutor, e devem ser robustos ao ruído ambiente (FORSYTH, 1995).

A análise da energia é um método que provê bons meios para a obtenção de coeficientes importantes acerca da identidade fonética dos sons. Sons fricativos (como o *s*) têm menos energia que sons como os da vogal *a*. Contudo, para que a energia possa ser usada como coeficiente de forma confiável, é necessário que o som esteja devidamente normalizado (RABINER & JUANG, 1993).

A Análise Espectral tem como princípio básico aplicar um filtro passa-banda, de faixa estreita ao sinal estudado (Gráfico 4), percorrendo toda a faixa de freqüências de interesse.

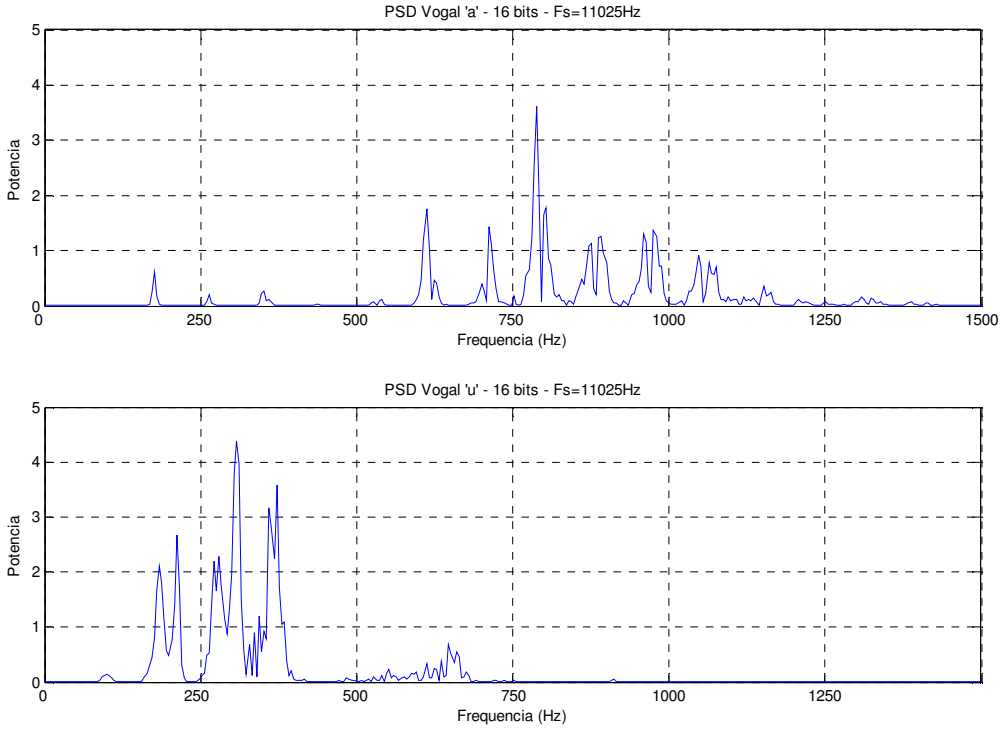

Gráfico 5 – Densidade Espectral de Potência, PSD.

A representação espectral obtida (Gráfico 5), pode ser usada diretamente como uma representação do sinal de voz.

A autocorrelação (Gráfico 6) é uma medida estatística que informa o quanto uma variável aleatória pode influenciar seus vizinhos. Assim supondo, uma variável aleatória X_t, discreta, estacionária e dependente do tempo médio μ, tem sua autocorrelação $R(k)$ definida como:

$$R(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2} \quad (2.1)$$

onde o E[.] é o valor médio, *k* é o deslocamento no tempo e o *σ* é a variância da variável X_t.

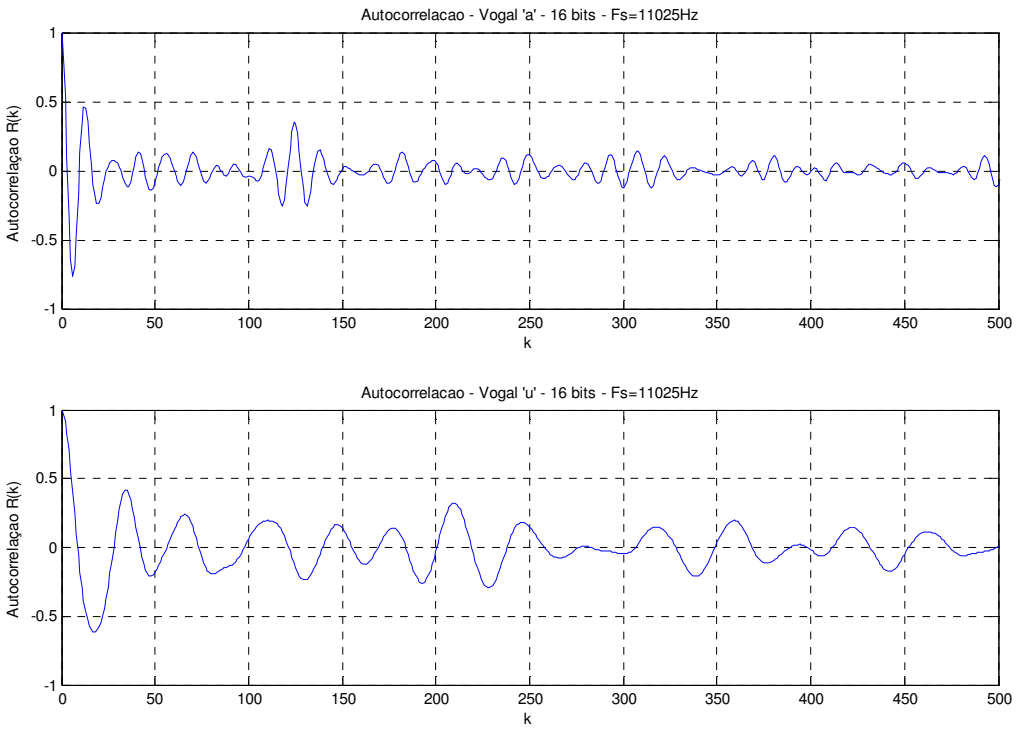

Gráfico 6 – Autocorrelação

O Coeficiente *Cepstral* (*Cepstrum*) é a Transformada de Fourier inversa do logaritmo neperiano da transformada do sinal de entrada, como mostrado na ilustração 4. A facilidade do *Cepstrum* em capturar a estrutura do sinal de voz segmentada torna-o no coeficiente mais comum para aplicações em tecnologias aplicadas à fala (FORSYTH, 1995). É possível calcular os coeficientes *cepstrais* (Gráfico 7) por meio de:

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} X(n) e^{-\frac{jnk2\pi}{N}} \quad (2.2)$$

$$P_m \triangleq \lim_{N\to\infty} \frac{\sum_{-N}^{N} |x(t)|^2}{2N+1} \quad (2.3)$$

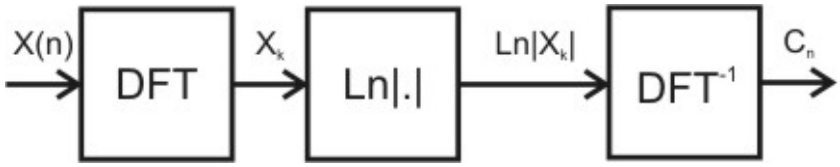

Ilustração 4 – Modelo de cálculo do *Cepstrum*.

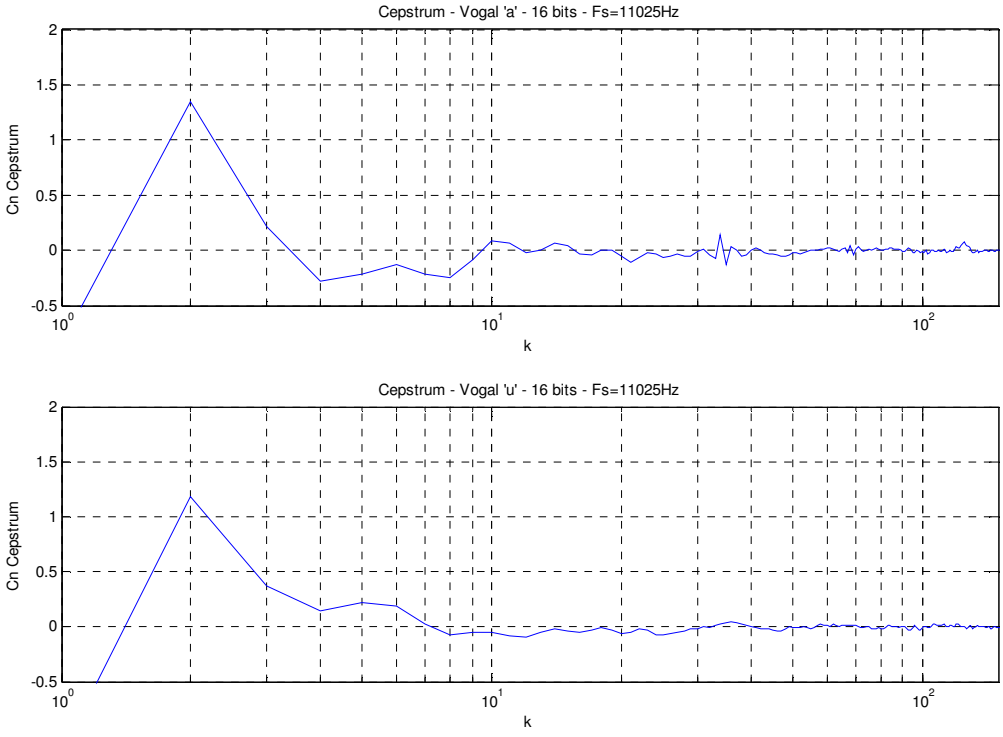

Gráfico 7 – *Cepstrum* das vogais '*a*' e '*u*'

3 REDES NEURAIS

Uma rede neural artificial (RNA) é um sistema de processamento de informação que possui algumas características de desempenho em comum com as redes neurais biológicas. Os modelos neurais artificiais têm como principal fonte de inspiração as redes neurais biológicas. A natureza das RNAs faz com que seu estudo seja multidisciplinar, envolvendo pesquisadores de diversas áreas, como a neurofisiologia, a psicologia, a física, a computação e a engenharia.

Neurofisiologistas e psicólogos são particularmente interessados em compreender o funcionamento do sistema neural humano: as características de resposta a estímulos apresentada por neurônios individuais – bem como as redes de neurônios – são alvo de estudo dos neurofisiologistas, enquanto os psicólogos estudam funções do cérebro no nível cognitivo e estão interessados na utilização de técnicas baseadas em redes neurais para criar modelos detalhados do comportamento humano.

Inspirados na habilidade apresentada por seres humanos e por outros animais no desempenho de funções (como processamento de informações sensoriais e capacidade de interação com ambientes pouco definidos), os engenheiros estão preocupados em desenvolver sistemas artificiais capazes de desempenhar tarefas semelhantes. Habilidades como capacidade de processamento de informações incompletas ou imprecisas e generalização são propriedades desejadas em tais sistemas. Segundo Haykin (1999, p.2):

> Uma rede neural é um processador distribuído altamente paralelo, composto de unidade de processamento simples, que são naturalmente propensas a armazenar conhecimento experimental e a torná-lo disponível para o uso. Ela se assemelha ao cérebro em dois aspectos:
> 1. O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizado.
> 2. As forças das conexões entre neurônios, conhecidas como pesos sinápticos, são usadas para armazenar o conhecimento adquirido.

Com o surgimento dos computadores, houve um aumento significativo na busca do desenvolvimento de máquinas inteligentes. A meta é obter sistemas capazes de desempenhar as mesmas tarefas que os seres humanos de forma

eficiente, tais como o reconhecimento de padrões, controle de sistemas, etc. As frustradas tentativas de reproduzir esta capacidade têm como principais fatores adversos a ausência da capacidade de associação, o conhecimento distribuído, a baixa tolerância a falhas e o processamento paralelo distribuído (HAYKIN, 1999).

3.1 HISTÓRIA DAS REDES NEURAIS

A busca pelo desenvolvimento das redes neurais envolve todas as áreas da ciência. Um exemplo é o fato de que os primeiros estudos quanto à modelagem do processamento dos neurônios foram descritos por neurofisiolosgistas. Com o passar do tempo, engenheiros e cientistas computacionais passaram a utilizar estes resultados para o desenvolvimento de máquinas que podem superar a capacidade dos computadores atuais. Devido ao notável potencial destes modelos, engenheiros acompanharam de perto as investigações do funcionamento do cérebro pelos neurofisiologistas e começaram a projetar um fundamento matemático destas investigações. Com o aperfeiçoamento destes modelos e o surgimento de mecanismos de treinamento, um grande interesse científico evidenciou-se, levando a uma notável expansão no conhecimento destes modelos, que ficaram cada vez mais complexos.

No ano de 1943, os biólogos McCullogh & Pitts apresentaram um modelo de neurônio como unidade de processamento binário e provaram que estas unidades são capazes de executar muitas das operações que podem ser descritas em termos lógicos. Estes modelos, apesar de simplistas, inspiraram a criação de computadores digitais tais como o de Von Neumann. Em 1958, Rosenblatt introduziu uma nova abordagem com o desenvolvimento do *perceptron*. Sua principal contribuição foi o teorema de convergência do *perceptron*. Os desenvolvimentos de Rosenblatt criaram grande expectativa das potencialidades desta linha de investigação, visto que a maior parte das instigações da época era de natureza heurística.

Nos ano de 1969, publicações mostrando a limitação dos sistemas atuais propiciaram um êxodo generalizado, deixando apenas um pequeno número

de investigadores trabalhando com redes neurais nos anos 70. Em 1982, o físico Hopfield publicou um estudo sobre materiais paramagnéticos (ou vidros de spin), onde, usando conceitos conexionistas, obteve importantes resultados que fazem ressurgir o interesse pelas redes neurais. Hopfield considerou um conjunto de neurônios dispostos de forma que suas saídas fossem realimentadas para as entradas. A rede neural de Hopfield pode ser considerada como um sistema dinâmico com um número finito de estados de equilíbrio, de forma que o sistema evolui para um desses estados. Pode-se dizer, ainda, que é natural que a localização destes estados de equilíbrio possa ser controlada pela intensidade das conexões da rede. Uma característica importante deste modelo é a recorrência: As saídas da rede são ligadas as entradas com um atraso de tempo (Ilustração 5), assim a resposta sempre depende do estado anterior.

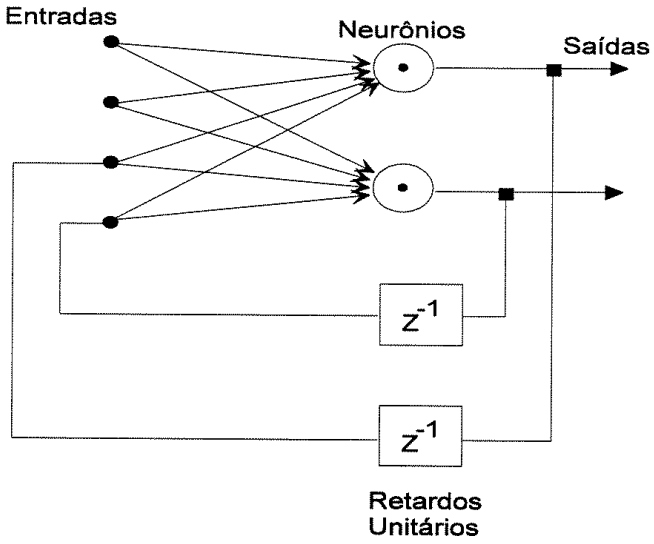

Ilustração 5 – Rede recorrente com neurônios ocultos

A área de redes neurais se colocou como uma das prioridades na obtenção de recursos quando um método para ajustar os parâmetros dos *perceptrons* de multicamadas com alimentação direta (*feed forward multilayered perceptrons*) foi desenvolvido. Tal método foi baseado em um algoritmo denominado *backpropagation*; este se tornou largamente conhecido com a publicação, em 1986, do livro "*Parallel Distributed Processing*" por McClelland e Rumelhart, mostrando

para investigadores de diferentes áreas as interessantes aplicações para redes neurais artificiais.

3.2 ANTECEDENTE BIOLÓGICO

Conhecedores das limitações, os investigadores se voltaram à tarefa de criar modelos simplificados do funcionamento do cérebro humano, começando pela parte mais elementar: o neurônio (Ilustração 6).

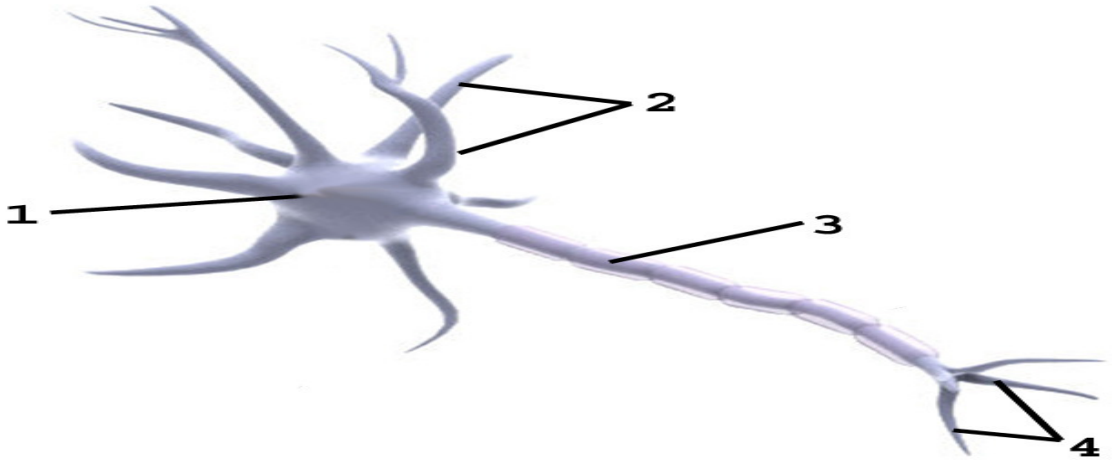

Ilustração 6 – Modelo do neurônio humano.

Nesta figura, pode-se ver um modelo simplificado do neurônio humano. Nele, encontram-se: os dendritos (2), que se comunicam com os outros neurônios através da sinapse; o corpo celular (1), onde ligam-se os dendritos; o axônio(3) e suas terminações(4), que são a prolongação maior do neurônio e é responsável pelo transporte de impulsos de saída para o próximo neurônio.

O modelo básico (Ilustração 7) é caracterizado como a unidade de processamento de informação fundamental para o sistema neural.

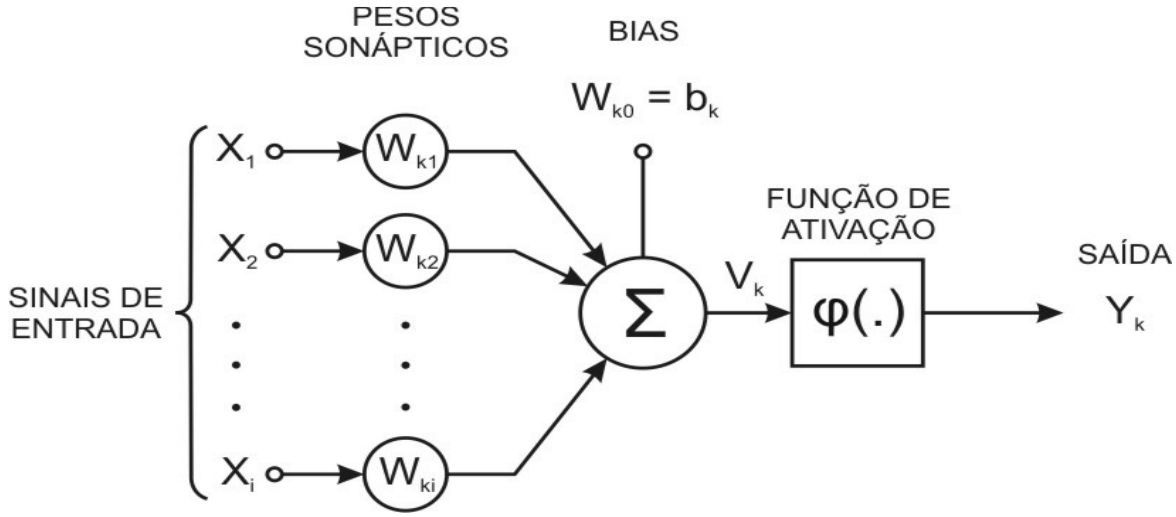

Ilustração 7 – Modelo não linear de um neurônio Fonte: Haykin, 2002

No modelo da ilustração 7, também se identificam partes elementares, que podem ser divididas em três estágios. A primeira é um conjunto de pesos sinápticos que caracterizam a força de conexão. A entrada é multiplicada por este peso, de forma a ser atenuada ou reforçada. A segunda pode ser vista como sendo um somador ou combinador linear, que soma as entradas ponderadas pelos pesos sinápticos. A terceira seria uma função de ativação com o intuito de limitar os valores de saída do neurônio.

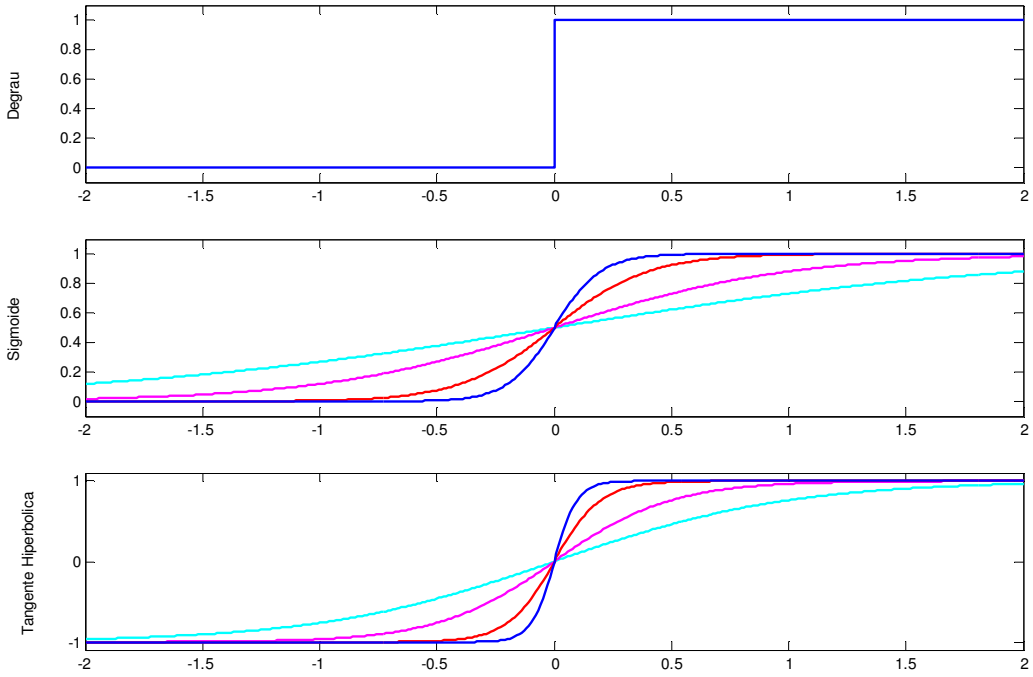

Gráfico 8 – Funções de ativação

É possível identificar três tipos de topologia de conexão dos neurônios para formar uma rede neural artificial: redes totalmente interconectadas, redes não recorrentes entre camadas e redes recorrentes entre camadas. Na primeira, tem-se uma topologia onde cada unidade se conecta consigo e com todas as outras unidades da rede. A segunda englobada as ligações entre as saídas das unidades de um nível inferior em direção às entradas de níveis superiores, não existindo conexões entre elementos de um mesmo nível. O terceiro tipo, por fim, tem a mesma estrutura do segundo, mas com a possibilidade de saídas de unidades de um nível superior estabelecerem ligações com entradas de todas as unidades de um nível inferior, representando uma realimentação interna de informação, conforme é exemplificado na ilustração 8.

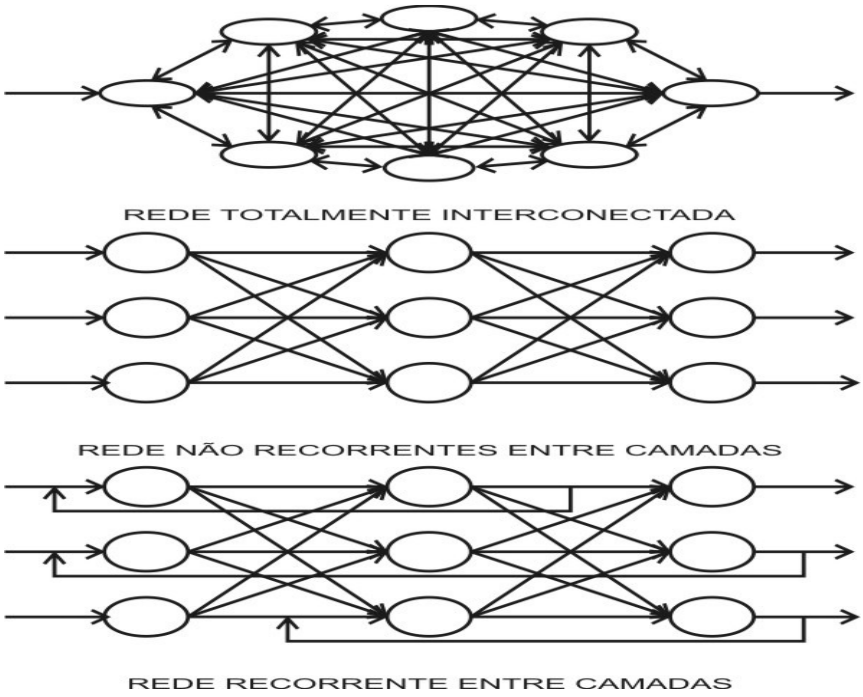

Ilustração 8 – Modelos de topologia de redes neurais.

3.3 REDES NEURAIS DE MULTICAMADA (*MULTILAYER PERCEPTRON – MLP*)

Redes neurais de arquitetura *Multilayer Perceptron*, treinadas pelo algoritmo *backpropagation* de aprendizado supervisionado possuem uso generalizado. Derivadas dos *perceptrons*, essas redes possuem uma ou mais camadas intermediárias compostas de unidades de processamento não lineares que lhes conferem grande poder, como classificadores de padrões.

Em geral, a quantidade de entradas utilizadas é proporcional à dimensão do padrão de características, sendo estas do tipo freqüência e/ou do tipo temporal. A quantidade de saídas – quando utiliza-se essa rede em reconhecimento de padrões – é proporcional à quantidade de classes. A quantidade de neurônios intermediários não está relacionada por nenhuma limitação externa; porém, seu valor é determinístico para que a rede possa separar eficientemente as amostras: se o valor é pequeno, a rede não terá suficientes graus de liberdade para separar o espaço de estados numa quantidade de classes desejada e, caso contrário, se for muito grande, a rede não conseguirá uma boa generalização da base de dados.

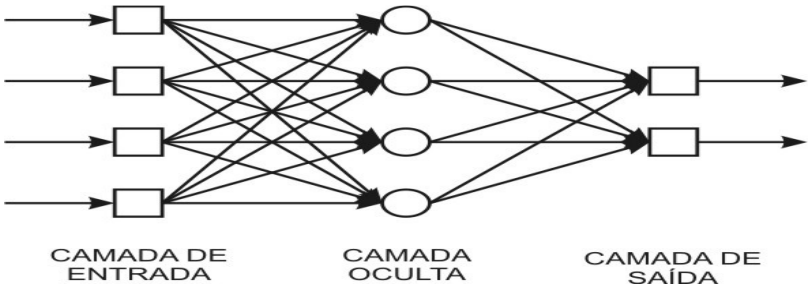

Ilustração 9 – Multi–Layer Perceptron

3.4 MÉTODOS DE APRENDIZADO

As Redes Neurais Artificiais (RNAs) têm a capacidade de aprender por exemplos e fazer interpolações e extrapolações do que aprenderam. O aprendizado conexionista busca determinar a intensidade entre as conexões dos neurônios. Um algoritmo de aprendizado é um conjunto bem definido de procedimentos para adaptar parâmetros de uma RNA para que, assim, ela possa aprender uma determinada função. Existem vários algoritmos de aprendizado, cada qual com sua vantagem e desvantagem, e suas diferenças estão basicamente na maneira com que é feito o ajuste de peso.

> Aprendizagem é o processo pelo qual os parâmetros de uma rede neural são ajustados através de uma forma continuada de estimulo pelo ambiente no qual a rede está operando, sendo o tipo especifico de aprendizagem realizada definida pela maneira particular como ocorrem os ajustes nos parâmetros (BRAGA; LUDEMIR; CARVALHO, 2000).

Diversos métodos foram desenvolvidos para treinar as redes neurais. Esses métodos foram divididos em dois paradigmas principais:

Supervisionado: necessita de um “professor” durante a fase de aprendizagem, que antecede a utilização (execução) da rede.

Não-supervisionado: direcionado por correlações existentes nos dados de entrada e, portanto, não necessita de um “professor”.

3.4.1 Aprendizado supervisionado

Este método de aprendizado é comumente utilizado no treinamento de RNAs e é chamado de supervisionado porque as entradas e saídas desejadas são fornecidas por um supervisor (professor) externo. O objetivo é calcular os parâmetros da RNA para encontrar uma ligação entre os pares de entrada e saída fornecidos. Para isso, o supervisor coloca uma entrada e indica o resultado esperado: com a saída corrente calculada pela RNA e a resposta esperada, o supervisor pode calcular o erro e enviar a rede (Ilustração 10). A cada interação, pequenos ajustes são feitos nos parâmetros da rede de forma a minimizar os erros. A soma dos erros quadráticos é normalmente utilizada como medida de desempenho e também como função de custo a ser minimizado (BRAGA; LUDEMIR; CARVALHO, 2000).

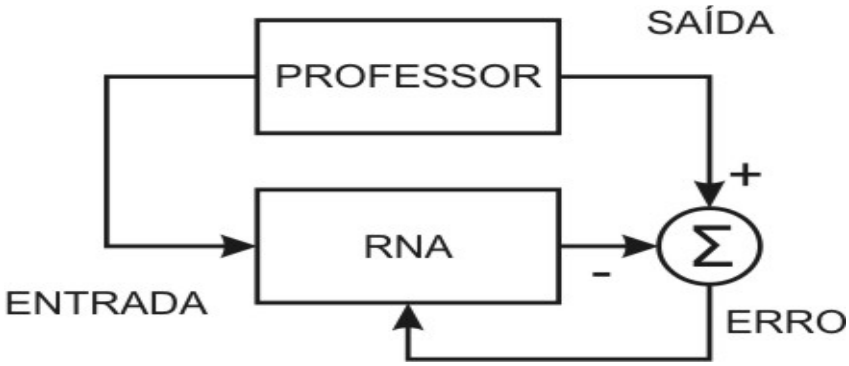

Ilustração 10 – Aprendizado supervisionado

Este tipo de treinamento tem como desvantagem o baixo rendimento quando surgir uma situação não treinada e o supervisor não estiver presente. Exemplos comuns deste tipo de treinamento são os algoritmos da regra delta e sua generalização para redes de múltiplas camadas, o algoritmo *backpropagation*.

A diferença entre a soma das entradas ponderadas pelos pesos (saída calculada) e a saída desejada é denominada como o erro da RNA. O termo e(t) do erro pode ser escrito como e(t) = d – y(t), onde d é a saída desejada e y(t) a saída

calculada no instante t. De forma genérica, a correção dos pesos é apresentada como:

$$\mathrm{W_i(t+1) = W_i(t) + A * x(t)} \tag{3.1}$$

onde A é a taxa de aprendizado e $x(t)$é a entrada para o neurônio i no tempo t.

A dedução das equações dos algoritmos de treinamento envolve a minimização da soma dos erros quadráticos das saídas:

$$\mathrm{E(W) = \frac{\sum_{i=1}^{n} W_i{}^2}{2}} \tag{3.2}$$

tendo essa somatória contemplado todos os nós da rede.

A equação 3.2 define uma superfície de erro dependente do modelo de construção da rede. Em qualquer situação, o objetivo é partir de um ponto arbitrário dessa superfície e move-lo até o mínimo global dela.

3.4.2 Aprendizado não supervisionado

Neste método de aprendizado, inexistem professores ou supervisores para acompanhar o aprendizado. O aprendizado supervisionado tem muitas semelhanças com o aprendizado do ser humano, mas o aprendizado de muitos dos nossos sistemas biológicos ocorrem semelhantes a aprendizados não supervisionados, como, por exemplo, a visão e a audição nos seus estágios iniciais.

Neste método, somente o padrão de entrada está disponível à rede. A rede estabelece uma regularidade estatística da entrada e desenvolve-se para formar representações características da entrada e, assim, criar novas classes ou grupos de saída automaticamente. Para isso, é preciso que haja uma redundância nos dados de entrada, necessários para encontrar padrões ou características nos dados.

Como este método não é parte deste trabalho, não será detalhado.

3.4.3 Aprendizado por reforço

Este método é visto como um caso particular do aprendizado supervisionado. A diferença entre o aprendizado supervisionado e o aprendizado por reforço está na medida usada em cada um dos sistemas (BRAGA; LUDEMIR; CARVALHO, 2000).

Neste método, a única informação de realimentação dada à rede é se a saída está correta ou não. Este método é uma forma de aprendizado on-line obtida por um mapeamento de entrada e saída através de um processo de triagem de erros desenvolvido para maximizar o índice de desempenho escalar chamado sinal de reforço (BRAGA; LUDEMIR; CARVALHO, 2000).

Como este método também não integra este trabalho, não será detalhado.

3.5 ALGORITMO DE RETRO-PROPAGAÇÃO (*BACKPROPAGATION*)

O algoritmo *backpropagation* é um algoritmo supervisionado que utiliza pares de entrada e saída para ajustar os pesos da sua rede por meio de mecanismos de correção de erros. O treinamento ocorre em duas fases: na primeira fase, chamada de *forward*, o algoritmo passa pela rede para definir a saída da rede; a segunda, chamada de *backward*, utiliza a saída desejada e a calculada para atualizar os pesos das conexões.

Os ajustes dos pesos são realizados utilizando-se uma variação do método de Widrow, Hoff ou, ainda, da regra Delta**,** que utiliza o método do gradiente decrescente para a minimização do erro. Assim, esse algoritmo também recebe o nome de regra delta generalizada.

Considerando que a altura de um ponto é diretamente proporcional ao erro associado a este ponto, a solução está nos pontos mais baixos da superfície de erro (Ilustração 11). Para ajustar os pesos da rede de forma que eles atinjam o ponto mais baixo da superfície de erro, utiliza-se o método de gradiente descendente.

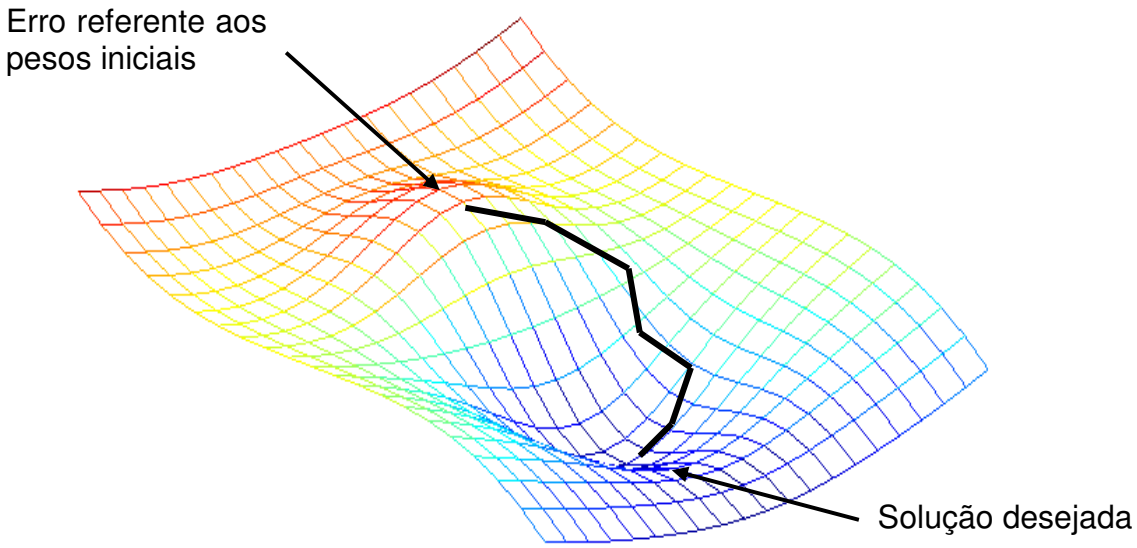

Ilustração 11 – Superfície de erro para uma rede MLP qualquer.

A gradiente de uma função é definida como a direção e o sentido em que a função tem a máxima variação. Para superfícies simples, este método encontra facilmente a solução adequada com erro mínimo; contudo, para superfícies complexas, não é possível garantir que o erro seja mínimo, pois o algoritmo pode convergir para superfícies de mínimos locais (Ilustração 10).

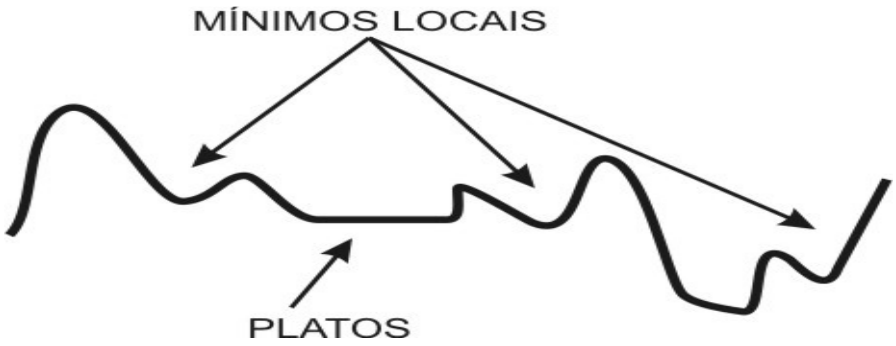

Ilustração 12 – Exemplo de mínimos locais.

Neste algoritmo, a função a ser minimizada é uma função de erro – ou energia – definida pela soma dos erros quadráticos mostrada na equação abaixo:

$$E = \frac{1}{2}\sum_p \sum_{i=1}^{k} \left(d_i^p - y_i^p\right)^2 \quad (3.3)$$

onde *E* é a medida de erro total, *p* é o numero de padrões, *k* é o numero de unidades de saída, d_i é a i-ésima saída desejada e y_i é a i-ésima saída gerada pela rede. Esta equação define o erro total cometido pela rede ou a quantidade de saídas que são diferentes das desejadas.

A regra delta generalizada necessita que as funções de ativação sejam semi-lineares, isto é, que sejam contínuas, diferenciáveis e que não sejam decrescentes da entrada total recebida. Para efeito de demonstração, é posto que a minimização do erro de cada padrão individual levará à minimização do erro total. Assim, a forma de cálculo do erro é simplificada à equação:

$$E = \frac{1}{2}\sum_{j=1}^{k} \left(d_j - y_j\right)^2 \quad (3.4)$$

O algoritmo sugere que a variação dos pesos está de acordo com a gradiente decrescente do erro com relação ao peso, ou seja:

$$\Delta w_{ji} \propto -\frac{\partial E}{\partial w_{ji}} \quad (3.5)$$

Para definir como cada um dos pesos deve ser ajustado de forma a reduzir o erro total da rede, utilizamos a regra da cadeia e temos que:

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} \quad (3.6)$$

onde $net_j = \sum_{i=1}^{n} x_i w_{ji}$. Por tanto:

$$\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial \sum_{i=1}^{n} x_i w_{ji}}{\partial w_{ji}} = x_i \quad (3.7)$$

A primeira parte da equação 3.6, $\frac{\partial E}{\partial net_j}$ se refere ao erro do neurônio *j* e geralmente é abreviada para δ_j:

$$\delta_j = \frac{\partial E}{\partial net_j} \quad (3.8)$$

Nesta equação, também é usada a regra da cadeia para definir a sua derivada:

$$\delta_j = \frac{\partial E}{\partial net_j} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial net_j} \quad (3.9)$$

A primeira derivada da função acima depende de onde está localizado o neurônio *j*; se ele estiver na última camada, tem-se:

$$\frac{\partial E}{\partial y_j} = \frac{\partial\left(\frac{1}{2}\sum_{i=1}^{k}(d_i - y_i)\right)}{\partial y_j} = (d_i - y_i) \quad (3.10)$$

Caso o neurônio j não esteja na saída, é usada a regra da cadeia para escrever a equação:

$$\frac{\partial E}{\partial y_j} = \sum_{l=1}^{M} \frac{\partial E}{\partial net_l} \frac{\partial net_j}{\partial y_j} = \sum_{l=1}^{M} \frac{\partial E}{\partial net_l} \frac{\partial \sum_{i=1}^{k} w_{il} y_i}{\partial y_j} = \sum_{l=1}^{M} \frac{\partial E}{\partial net_l} w_{jl} = \sum_{l=1}^{M} \delta_l w_{jl} \quad (3.11)$$

Ainda pode-se definir segunda derivada da equação 3.9 como sendo:

$$\frac{\partial y_i}{\partial net_j} = \frac{\partial f(net_i)}{\partial net_j} = f'(net_i) \quad (3.12)$$

Pode-se, assim, generalizar a fórmula de ajuste de pesos proposta pela equação 3.5 como:

$$w_{ji}(t+1) = w_{ji}(t) + \eta \delta_j(t) x_i(t) \quad (3.13)$$

sendo que a função $\delta_j(t)$ é definida pela posição em que o neurônio se encontra na rede. No caso de este estar na saída, temos $\delta_j = (d_i - y_i)\, f'(net_i)$; caso contrário, tem-se

$$\delta_j = \sum_{l=1}^{M} \delta_l w_{jl} \quad f'(net_i). \quad (3.14)$$

4 SIMULAÇÕES

Neste capítulo são discutidos detalhes dos métodos utilizados para o reconhecimento dos sinais de voz. Para tal, foram realizadas simulações utilizando algumas rotinas implementadas no programa Matlab (vide anexos). Serão simuladas, simultaneamente, conjuntos de palavras isoladas e comandos compostos por duas palavras. Estas simulações usaram modelos baseados na teoria de Redes Neurais.

As palavras utilizadas como comando foram gravadas, anteriormente às simulações, em arquivos digitais de som do tipo WAV (ex azul.wav). As palavras foram gravadas com um microfone Edifier CD-6631MV acoplado ao microcomputador, durante os vários períodos do dia.

A escolha dos locutores foi feita ao acaso, selecionando pessoas com os mais variados timbres de voz, sotaques, de ambos os sexos e com idade entre dezesseis e quarenta anos. Essas variações nas gravações têm como objetivo demonstrar que um reconhecedor de fala é capaz de identificar comandos independente do estilo de falar do locutor. Para as simulações, foram escolhidas 8 palavras/comandos distintos, são eles: amarelo, azul, verde, vermelho, para baixo, para cima, virar esquerda, virar direita. Cada palavra/comando foi gerada oito vezes por onze locutores distintos. Sendo assim temos 704 palavras gravadas, todas com uma resolução de 16bits e um único canal (mono) e com uma taxa de amostragem de 44100Hz. Posteriormente as locuções tiveram sua taxa de amostragem reduzida para 11025Hz, para diminuir o esforço computacional e assim diminuir o tempo as simulações. A freqüência de amostragem, segundo o Teorema de Nyquist, poderia ser ainda menor (próximo de 9000Hz), mas optamos por 11025 pois a maioria dos softwares de aquisição de voz trabalham com esta freqüência como padrão.

De todos os locutores, oito foram utilizados para o treinamento da rede neural e outros três foram guardados para formar um conjunto de teste.

No gráfico 8, são mostradas as representações gráficas da palavra amarelo, em sua forma original, forma que foi gravada, e em sua forma filtrada, com os modelos detalhados no capítulo 2.

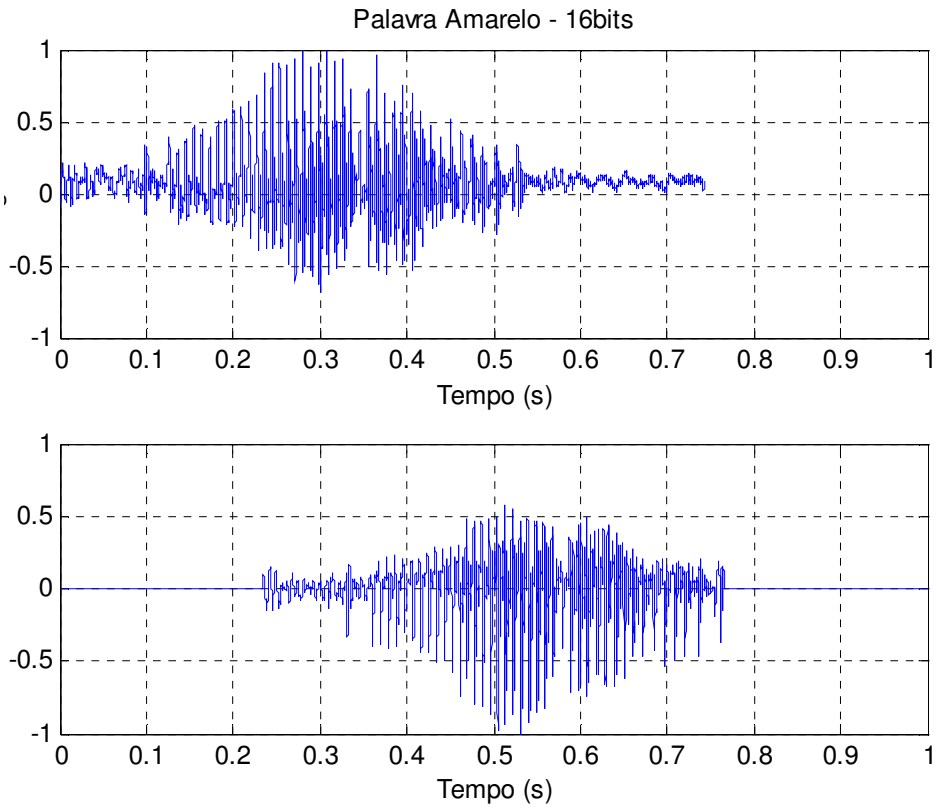

Gráfico 9 – Representações gráficas da palavra “amarelo” com taxa de amostragem 44,1kHz e 11,025kHz e representação utilizando 16 bits.

Para este trabalho, escolhe-se entre os diversos coeficientes usados como parâmetros para reconhecimento: a autocorrelação, o cesptrum, a densidade espectral de potência e a energia contida no sinal (Gráfico 10). Para que fosse possível extrair o maior numero de informações de um sinal, dividiu-se em janelas retangulares de 20ms, pois os fonemas falados normalmente tem esta duração, estas se sobrepunham em 4ms para diminuir o efeito de borda gerado na aquisição dos coeficientes (Gráfico 10).

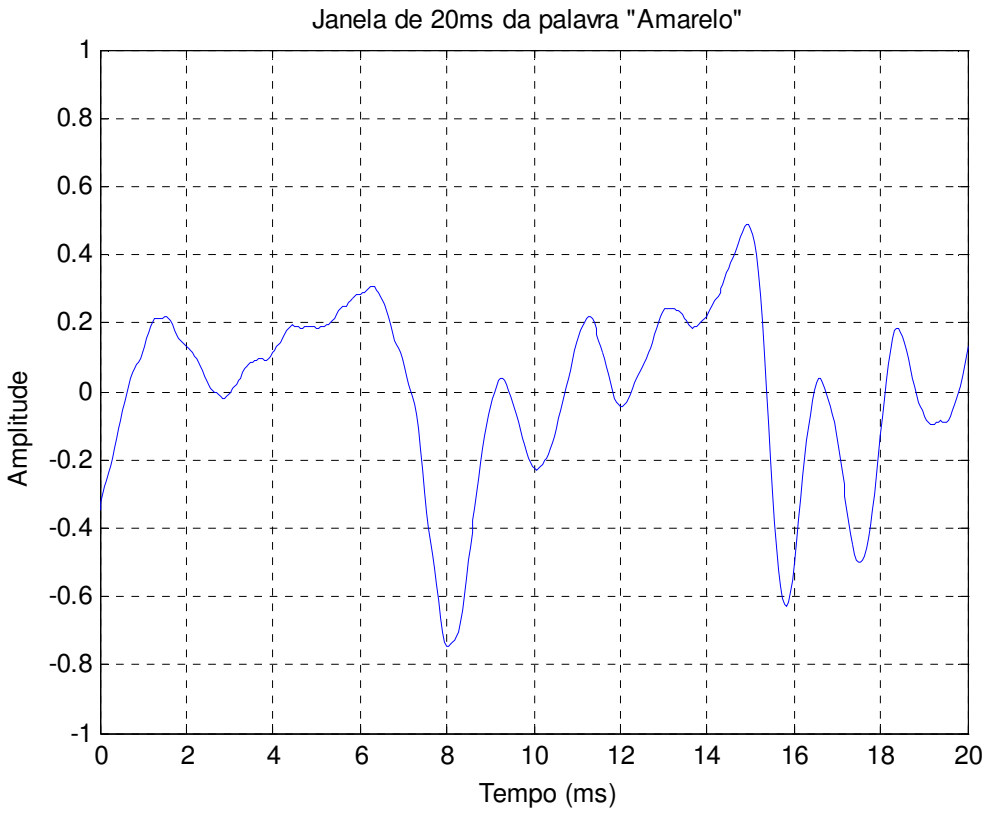

Gráfico 10 – Representação gráfica da primeira janela da palavra “amarelo”.

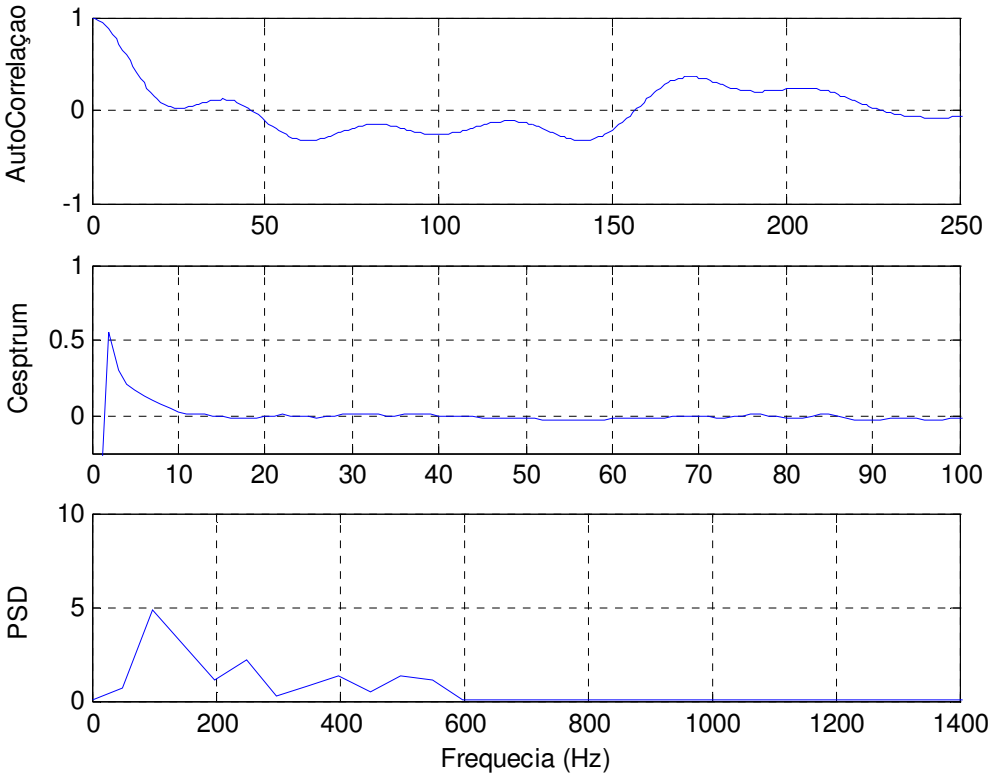

Gráfico 11 - Representação gráfica dos coeficientes de AutoCorrelação, Cesptrum e PSD da janela da palavra “Amarelo”.

Para alimentar a Rede Neural foi construída uma matriz composta pelos valores máximos dos coeficientes do *Cesptrum* e PSD e o valor do segundo máximo da Autocorrelação foram usados os valores de pico em cada janela e para a energia foi usado a energia contida no sinal da janela.

Após construídas, essas quatro matrizes foram treinadas em vinte Redes Neurais MLP (*Multi Layer Perceptron*) diferentes para cada tipo de coeficiente, usando cinco, dez, quinze, vinte e vinte e cinco camadas de neurônios e quinhentas, mil e quinhentas, duas mil e quinhentas e três mil e quinhentas épocas de treinamento. Este treinamento foi realizado dez vezes para cada coeficiente. Os resultados percentuais médios e seus respectivos desvios estão apresentados nos quadros de 1 à 8.

Épocas / Neurônios	500	1500	2500	3500
5	11,36±0,72	11,02±0,72	10,39±1,36	9,81±1,21
10	7,79±1,79	5,80±0,88	5,72±1,22	4,77±0,68
15	6,05±0,95	4,22±0,51	3,20±0,43	2,97±0,59
20	4,74±0,72	2,79±0,53	2,33±0,26	1,86±0,36
25	4,29±0,43	2,30±0,66	1,60±0,23	1,30±0,25

Quadro 1 – Média de erros encontrados para PSD nos treinamentos

Épocas / Neurônios	500	1500	2500	3500
5	14,27±0,72	14,25±0,72	13,75±1,36	14,03±1,21
10	12,06±1,79	12,06±0,88	11,75±1,22	12,48±0,68
15	11,56±0,95	10,78±0,51	10,80±0,43	10,75±0,59
20	11,08±0,72	10,61±0,53	10,38±0,26	10,34±0,36
25	10,81±0,43	10,17±0,66	9,910±0,23	10,25±0,25

Quadro 2 – Média de erros encontrados para PSD nos testes

Épocas / Neurônios	500	1500	2500	3500
5	18,80±1,74	19,46±1,09	18,92±0,18	17,73±1,42
10	17,06±2,87	13,65±2,66	15,42±2,63	12,59±3,13
15	12,53±2,47	9,75±2,85	9,78±2,07	9,48±1,65
20	9,27±1,57	7,59±2,27	7,07±2,06	5,90±1,23
25	6,97±1,05	5,81±1,03	5,51±1,72	4,39±0,92

Quadro 3 – Média de erros encontrados para Energia nos treinamentos

Épocas / Neurônios	500	1500	2500	3500
5	19,25±1,74	19,82±1,09	19,37±0,18	18,72±1,42
10	18,51±2,87	16,29±2,66	17,00±2,63	15,80±3,13
15	15,56±2,47	13,40±2,85	13,98±2,07	12,85±1,65
20	12,93±1,57	12,39±2,27	11,44±2,06	11,32±1,23
25	10,90±1,05	10,74±1,03	11,01±1,72	10,64±0,92

Quadro 4 – Média de erros encontrados para Energia nos testes

Épocas / Neurônios	500	1500	2500	3500
5	21,86±0,02	21,90±0,13	21,84±0,08	21,71±0,29
10	21,43±0,91	21,72±0,46	21,12±1,02	21,11±0,95
15	20,98±1,17	19,61±0,82	20,51±1,19	20,33±1,21
20	19,69±1,53	19,95±1,19	20,36±1,36	19,38±1,14
25	19,84±1,59	19,43±1,24	18,17±1,60	18,71±0,69

Quadro 5 – Média de erros encontrados para Autocorrelação nos treinamentos

Épocas / Neurônios	500	1500	2500	3500
5	21,89±0,02	21,90±0,13	21,94±0,08	21,78±0,29
10	21,57±0,91	21,83±0,46	21,36±1,02	21,27±0,95
15	21,10±1,17	19,85±0,82	20,87±1,19	20,49±1,21
20	19,82±1,53	20,32±1,19	20,46±1,36	19,83±1,14
25	20,19±1,59	19,65±1,24	18,55±1,60	18,95±0,69

Quadro 6 – Média de erros encontrados para Autocorrelação nos testes

Épocas / Neurônios	500	1500	2500	3500
5	12,28±1,07	11,35±2,31	11,13±1,97	10,59±1,73
10	9,86±0,71	7,15±0,52	6,77±0,68	6,29±0,48
15	9,40±0,65	7,20±0,26	6,22±0,55	5,24±0,61
20	9,64±0,50	7,27±0,38	5,63±0,48	4,97±0,37
25	9,70±0,48	7,15±0,31	5,83±0,59	4,83±0,60

Quadro 7 – Média de erros encontrados para Cesptrum nos treinamentos

Épocas / Neurônios	500	1500	2500	3500
5	15,14±1,07	15,35±2,31	15,02±1,98	14,85±1,73
10	14,25±0,71	14,16±0,52	14,04±0,68	14,37±0,48
15	14,16±0,65	14,16±0,26	13,82±0,55	14,25±0,61
20	14,10±0,50	13,95±0,38	13,67±0,48	13,51±0,37
25	14,30±0,48	14,25±0,31	13,72±0,59	13,72±0,60

Quadro 8 – Média de erros encontrados para Cesptrum nos testes

Analisando somente as medias dos erros das tabelas acima pode-se observar que o número de coeficientes classificados errados no treinamento diminui com o aumento da quantidade de neurônios; porém, nos testes essa conclusão não é verdadeira, pois com o aumento do numero de neurônios da camada escondida diminui-se o erro até um limiar; que varia com o coeficiente analisado, depois este erro volta a aumentar. A justificativa para o aumento do erro esta na complexidade que a rede neural apresenta, e acima deste limiar, a rede começa a gerar sinais de sobre-ajuste que aumentam o erro.

5 CONCLUSÕES

Neste trabalho foi estudado aspectos teóricos referentes à formação da fala humana e sobre Redes Neurais. O foco deste estudo foi o de entender os conceitos básicos dos principais métodos e modelos existentes para o reconhecimento de voz.

No reconhecimento de sinais de voz, geralmente envolvem problemas associados a parametrização, como o pré processamento ou filtragem e estas foram sem duvida as maiores dificuldades deste trabalho.

Nas simulações quatro coeficientes foram testados: A Energia, a Densidade Espectral de Potência, a Autocorrelação e o Cesptrum. Antes de começar a simular as redes neurais as amostras de voz foram normalizadas. Essas amostras foram extraídas de dez locutores distintos em diferentes horários. As redes Neurais foram treinadas com diferentes quantidades de neurônios e com quantidades diferentes de épocas de treinamento.

Constatou-se, olhando somente os valores médios dos erros de treinamentos que, o coeficiente que apresentou o melhor resultado foi a Densidade Espectral de Potencia (PSD). Os resultados não são totalmente satisfatórios, pois espera-se que uma rede neural acerte cem por cento dos casos avaliados.

5.1 TRABALHOS FUTUROS

Para que se possa melhorar os resultados, em trabalhos futuros poderia ser estudado métodos de seleção das partes mais relevantes das características já estudadas.

Outras características poderiam ser estudadas como os coeficientes mel-ceptrais.

Outros tipos de redes neurais ou treinamentos também poderiam melhorar os resultados.

Ou ainda construir de um grande banco de dados de locuções para melhorar o treinamento e teste da rede.

REFERÊNCIAS BIBLIOGRÁFICAS

BALDI, P.; BRUNAK, S. **Bioinformatics**: The Machine Learning Approach. Adaptive Computation and Machine Learning. Cambridge, MA: MIT, 2001.

BECHETTI, C.; RICOTTI, L. P. **Speech Recognition**: Theory and C++ implementation. West Sussex, England: John Wiley & Sons, 1999.

BISHOP, C. M. **Neural Networks for pattern Recognition**. Oxford: Oxford University Press, 1995.

BITTENCOURT, G. **Inteligência artificial**: ferramentas e teorias. 2. ed. Florianópolis: Editora da UFSC, 2001.

BRAGA, A. P. **Redes neurais artificiais**. São Paulo: LTC, 2007.

CHIOVATO, A. G.; FRAGA, F. J.; ATTIA, F. L.; AMARAL. G. F. **Sistema de reconhecimento de fala com ruído para automóveis**. Belo Horizonte, 2006

CIHOCKI, A.; UNBEHAUEN, R. **Neural Networks for Optimization and Signal Processing**. [s.l.]: John Wiley, 1996.

DAOUD, M.; KREMER, S.C. **Neural and Statistical Classification to Families of Bio-sequences, International Joint Conference on Neural Networks**. [s.l.]: [s.n.], 2006.

DELLER JR., J. R. & HANSEN, J. H. L. & PROAKIS, J. G., **Discrete-Time Processing of Speech Signals**, 2.a edição, Wiley-IEEE Press, 1999

FARRELL, K.; MAMMONE, R. J.; ASSALEH, K. T. **Speaker recognition using neural networks and conventional classifiers. Acoustic, Speech and Signal Processing**. [s.l.]: IEEE Transactions, 1994.

FOLMER-JOHNSON, Tore N. O. .**Oscilações, ondas, acústica**. São Paulo: Nobel, 1968.

FORSYTH, D. A., 1993, **Recognizing Algebraic Surfaces from Their Outlines**, In International Conference on Computer Vision, 476-480.

HAYKIN, S. **Neural Networks**: A Comprehensive Foundation. New York, US: Prentice Hall, 1999.

HAWKINS, J.; BODÉN, M. **The Applicability of Recurrent Neural Networks for Biological Sequence Analysis**. [s.l.]: [s.n.], 2005.

LAWSON, A.D., HARRIS, D.M., GRIECO, J.J., 2003. **Effect of foreign accent on speech recognition in the NATO N-4 Corpus**. In: Proc. Eurospeech'03, Eur. Conf. on Speech Communication and Technology, Geneva, Switzerland, September 1-4, pp. 1505-1508.

LEMOS, D. R.; RODRIGUES, G. J.; HERNANDEZ, E. D. M. **Reconhecedor de voz via redes neurais.** São Paulo: [s.e.], 2004

KUBALA, F., ANASTASAKOS, A., MAKHOUL, J., NGUYEN, L., SCHWARTZ, R., ZAVALIAGKOS, E., 1994. **Comparative experiments on large vocabulary speech recognition**. In: Proc. ICASSP'94, IEEE Internat. Conf. on Acoustics, Speech, and Signal Processing, Adelaide, Australia, April 19-22, pp. 561-564.

MICHELI-TZANAKOU, E. **Supervised and Unsupervised Pattern Recognition**. [s.l]: John Wiley, 2000.

MITRA, S.; HAYASHI, Y. **Bioinformatics With Soft Computing**. [s.l.]: [s.n], 2006.

RABINER, L. R. & JUANG, B. H., 1986. **An introduction to hidden Markov models**. In IEEE ASSP Magazine, January, 4-15.

RABINER, L. R. & SCHAFER, R. W. **Digital processing of speech signals**. New Jersey: Prentice-Hall, 1978.

RIBAS, J. C.; CUNHA, F. L.; CLIQUET Jr., **A. Sistema de controle por voz aplicado à reabilitação humana**. São Paulo: [s.e.], 2005

TEVAH, R. T. **Implementação de um sistema de reconhecimento de fala contínua com amplo vocabulário para o português brasileiro**. 2006. Tese (Mestrado em Engenharia Elétrica) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006.

WANG, L. P.; FU, X. **Data Mining with Computational Intelligence**. Berlin: Springer, 2005.

XIANG, Z. **A Neural Network model for Chinese speech synthesis**. Beijing: [s.e.], 2007

ANEXO A – FUNÇÃO TREINAMENTO

```
%***************************************************************
% Este programa realiza o treinamento de uma rede perceptrons
% com uma camada intermediaria
% Metodo do gradiente
% Treinamento em batelada
% Data: 22/06/99
% Autor: Clodoaldo Aparecido de Moraes Lima
% Alterado em 20/05/2008
%***************************************************************

function [A,B,veterro_tr]=treinamento(X,S,A,B,h,nepocasmax)

% Calcula o numero de entradas e Ss
[N,ne]=size(X);
[N,ns]=size(S);

if isempty(A)&isempty(B)
   %disp('Inicializando a rede neural')
   A=rands(h,ne+1)/50;
   B=rands(ns,h+1)/50;
end

% Define a taxa de aprendizado
alfa=1;

%Define o erro minimo
erromin=1e-5;

% Define o numer de epocas maximo
%nepocasmax=250;

% Calcula a S da rede
Sr=perceptrons(X,A,B);

% Calcula o vetor com os erros
erro=Sr-S;

% Calcula o Erro quadratico medio
EQM=(1/N)*sum(sum(erro.*erro));
veterro_tr=[];
veterro_tr=[veterro_tr,EQM];
nepocas=0;
while  EQM>erromin & nepocas <nepocasmax

   % Incrementa o numero de epocas
   nepocas=nepocas+1;

   % Calcula o Gradiente
   [dJdA,dJdB]=grad(X,A,B,S,h);

   % Transforma para vetor
   vetdJdA=reshape(dJdA,1,h*(ne+1))';
   vetdJdB=reshape(dJdB,1,ns*(h+1))';

   % Monta o vetor gradiente
   g=[vetdJdA;vetdJdB];
```

```
    % Normaliza o vetor gradiente
    g=g/norm(g);

    % Transforma em vetor
    vetdJdA=g(1:h*(ne+1));
    vetdJdB=g(h*(ne+1)+1:end);

    % Transforma em matriz
    dJdA=reshape(vetdJdA',h,ne+1);
    dJdB=reshape(vetdJdB',ns,h+1);

    % Atualiza a matriz A e B
    Aatual=A-alfa*dJdA;
    Batual=B-alfa*dJdB;

    % Calcula a S da rede
    Sr=perceptrons(X,Aatual,Batual);

    % Calcula o vetor com os erros
    erro=Sr-S;

    % Calcula o Erro quadratico medio
    EQMatual=(1/N)*sum(sum(erro.*erro));

    while EQMatual>EQM
       alfa=alfa/2;

       % Atualiza a matriz A e B
       Aatual=A-alfa*dJdA;
       Batual=B-alfa*dJdB;

       % Calcula a S da rede
       Sr=perceptrons(X,Aatual,Batual);

       % Calcula o vetor com os erros
       erro=Sr-S;

       % Calcula o Erro quadratico medio
       EQMatual=(1/N)*sum(sum(erro.*erro));

    end

    % Incrementa a taxa de aprendizagem
    alfa=alfa*2;

    % Atualiza as matrizes de entrada e S
    A=Aatual;
    B=Batual;
    EQM=EQMatual;
    veterro_tr=[veterro_tr,EQM];
end
```

ANEXO B – FUNÇÃO PERCEPTRONS

```
function Y=perceptrons(X,A,B)

N=size(X,1);
Z=[X,ones(N,1)]*A';
V=tanh(Z);
Y=[V,ones(N,1)]*B';
```

ANEXO C – CODIGO DE EXTRAÇÃO DAS CARACTERISTICAS

```
clc
clear
warning off
td=20e-3;   %Tempo da janela
ts=4e-3;    %Tempo de sobreposiçao entre janelas

Caminho='E:\TGI\Palavras\filtrados\';


%
%Calcula coeficientres das cores (palavras isoladas)
%
MATRIX=0;
Aux=1;      %Variavel Auxiliar
for Loc=1:10
   for Cor=1:8
      for Palavra=1:7
         Arquivo=[Caminho 'Locutor' NUM2STR(Loc) '\cor' NUM2STR(Cor) ' (' NUM2STR(Palavra)
').wav']; %Cria nome da cor + numero
         [y,fs]=readwav(Arquivo, 's');   %Abre aquivo e normaliza

         yp=janela(y,fs,td,ts);         %Cria Janelas

         for k=0:size(yp,2)-1           %Coleta Valores dos Coeficientes
            X1(Aux,k+1)=max_rceps(yp(:,k+1));                          %MAX Cesptrum
            [X2(Aux,((k*2)+1)) X2(Aux,((k*2)+2))]=max_xcoor(yp(:,k+1));      %MAX AutoCorrelacao e
posicao
            X3(Aux,k+1)=max_PDS(yp(:,k+1),fs);                          %MAX PDS
            X4(Aux,k+1)=max_energia(yp(:,k+1));                        %Somatoria da Energia
         end
         MATRIX(Aux,Cor)=1;             %Cria indice
         Aux=Aux+1;
      end
   end
end
MATRIX=MATRIX*2;
Y=MATRIX-1;

save COEF X1 X2 X3 X4 Y
clc
clear
```

ANEXO D – FUNÇÃO TREINATESTA

```
function erro=treinatesta(X,Y,numarq)
AUX = 1;                              %Inicia a variavel auxiliar para montar o vetor erro
for n=5:5:25
   for N=500:1000:3500
      for i=1:10
         [xtr,ytr,xts,yts]=gera_dados(X,Y);

         [A,B,VetErro]=treinamento(xtr,ytr,[],[],n,N);       %Treina a rede
         yRtr=perceptrons(xtr,A,B);                          %Testa a rede
         [xx,ytrc]=max(ytr,[],2);
         [xx,yRtrc]=max(yRtr,[],2);
         ErroTreina(i)=sum(yRtrc~=ytrc)/size(yRtrc,1);    %Marca o minimo erro do teste
         yRts=perceptrons(xts,A,B);                          %Testa a rede
         [xx,ytsc]=max(yts,[],2);
         [xx,yRtsc]=max(yRts,[],2);
         ErroTesta(i)=sum(yRtsc~=ytsc)/size(yRtsc,1);    %Marca o minimo erro do teste
      end
         mtre = sum (ErroTreina)/10;                 %Define media do treinamento
         dtre = sqrt(sum((ErroTreina-mtre).^2)/(9)); %Define desvio do treinamento
         mtes = sum (ErroTesta)/10;                  %Define media do teste
         dtes = sqrt(sum((ErroTesta-mtes).^2)/(9));  %Define desvio do teste
         erro(AUX,1) = n;         %Salva na matriz de erros
         erro(AUX,2) = N;         %Salva na matriz de erros
         erro(AUX,3) = mtre;      %Salva na matriz de erros
         erro(AUX,4) = dtre;      %Salva na matriz de erros
         erro(AUX,5) = mtes;      %Salva na matriz de erros
         erro(AUX,6) = dtes;      %Salva na matriz de erros
         AUX = AUX+1;
   end
end
erro=[erro(:,1:2) erro(:,3:6)];
save (strcat('Arq',num2str(numarq)),'erro')
```

ANEXO E – CODIGO DE FILTRAGEM

```
Quanto=0.1;     %Percentual do silencio (Comum aos 2)
Treinamento1='E:\TGI\Palavras\Originais\';   %Origem
Treinamento2='E:\TGI\Palavras\filtrados\';   %Destino

for Loc=1:10
   for Cor=1:8
      for Palavra=1:7
         table1=[Treinamento1 'Locutor' 48+Loc '\cor' 48+Cor ' (' 48+Palavra ').wav']; %Cria nome da
cor + numero
         table2=[Treinamento2 'Locutor' 48+Loc '\cor' 48+Cor ' (' 48+Palavra ').wav']; %Cria nome da
cor + numero
         [y,fs]=readwav(table1, 's'); %abre aquivo e normaliza
         if fs==44100
            y=y(1:4:end);
            fs=11025;
         end
         y=filtropassafaixa(y,fs); %Corta frequencias que nao sao voz
         y=cortasilencio(y, Quanto); %Tira silencio do inicio e do fim do sinal
         y=[zeros(round((tempo-length(y))/2),1);y;zeros(ceil((tempo-length(y))/2),1)]; %Coloca todos
com o mesmo tempo
         fidx=writewav(y,fs,table2,'s'); % salva e normaliza
      end
   end
end
```

ANEXO F – FUNÇÃO GERADADOS

```
function [Xtr,Ytr,Xts,Yts]=gera_dados(X,Y)

[Ntr,nclass]=size(Y);
perc=0.7;
Itr=[];
Its=[];
for i=1:nclass
    [ii,jj]=find(Y(:,i)==1);
    Nc=length(ii);
    I=randperm(Nc);
    ii=ii(I);
    jj=jj(I);
    Itr=[Itr;ii(1:floor(0.7*Nc))];
    Its=[Its;ii(floor(0.7*Nc)+1:end)];
end

Xtr=X(Itr,:);
Ytr=Y(Itr,:);
Xts=X(Its,:);
Yts=Y(Its,:);
```

ANEXO G – FUNÇÃO CORTASILENCIO

```
function y=cortasilencio(x, xlim)
n=length(x);
xm=0;
Somax=0;
%define media do sinal
xm=mean(x);
%retira componente DC
x=x-xm;
%define media do sinal sem a componente DC
xm=mean(x);
%corta silencios
ncortei=0;
ncortef=0;
for i=1:n
   if x(i)>=xlim
      Somax=0;
      ni=0;
      xmi=0;
      for j=i:n
         Somax=Somax+x(j);
         ni=ni+1;
         xmi=Somax/ni;
         if xmi >= abs(xm*2)
            if ncortei==0
               ncortei=j;
               break
            end
         end
      end
   end
end
for i=n:-1:ncortei
   if x(i)>=xlim
      Somax=0;
      ni=0;
      xmi=0;
      for j=i:n
         Somax=Somax+x(j);
         ni=ni+1;
         xmi=Somax/ni;
         if xmi >= abs(xm*2)
            if ncortef==0
               ncortef=j;
               break
            end
         end
      end
   end
end
y=x(ncortei:ncortef);
```

ANEXO H – FUNÇÃO FILTROPASSAFAIXA

```
function x=filtropassafaixa(x,fs)
[B,A]=butter(4,[300/fs 4500/fs],'bandpass');
x=filter(B,A,x);
```

ANEXO I – FUNÇÃO JANELA

```
% voz - sinal de voz
% fs - frequencia de amostragem
% td - tempo de duraçao
% ts - tempo de sobreposiçao
% x - matriz com cada frame

function x=janela(voz,fs,td,ts)

%voz
Nd=round(fs*td);
Ns=round(fs*ts);
N=length(voz);
Nframe=ceil((N-Ns)/(Nd-Ns));

for i=1:Nframe,

   if (Nd*i-Ns*(i-1))<=N
      x(:,i)=voz(1+(Nd-Ns)*(i-1):Nd*i-Ns*(i-1));
   else
       x(:,i)=[voz(1+(Nd-Ns)*(i-1):end);zeros(Nd*i-Ns*(i-1)-N,1)];
   end
end
```

ANEXO J – FUNÇÃO MAX_PSD

```
function maxi1=max_PDS(x,fs)

n=length(x);
[P f]=psd(x,n,fs);
for i=1:length(P)
    if isnan(P(i))
        P(i)=0;
    end
end
maxi1=max(P);
```

ANEXO K – FUNÇÃO MAX_RCEPS

```
function maxi1=max_rceps(x)

z=rceps(x);
for i=1:length(z)
   if isnan(z(i))
      z(i)=0;
   end
end
maxi1=max(z);
```

ANEXO L – FUNÇÃO MAX_XCORR

```
function maxi1=max_xcoor(x)

N=length(x);
rx=xcorr(x,N,'coeff');
for i=1:length(rx)
    if isnan(rx(i))
        rx(i)=0;
    end
end
maxi1=max2(rx);
```

ANEXO M – FUNÇÃO MAX2

```
function [maxi2,posi2,nx]=max2(x)
maxi2=0;
posi2=0;
nx=0;

if sum(x)==0
   maxi2=0;
   posi2=1;
   nx=(x);
else
   a=x;
   x1=1;
   x2=2;
   n=length(x);
   [maxi1,posi1]=max(x);
   limit=0.1*maxi1;

   for i=posi1:-1:1
      if x(i)<limit
         x1=i;
         break
      end
   end
   for i=posi1:(n-1)
      if x(i)<limit
         x2=i;
         break
      end
   end
   nx=[x(1:x1(1)) x(x2(1):end)];
   maxi2=max(nx);


   if maxi2<maxi1*1.001 & maxi2>maxi1*0.999
      [maxi2,posi2,nx]=max2(nx);
   end
   maxi2=maxi2(1);
   for i=1:(n)
      if a(i)==maxi2
         posi2=i;
         break
      end
   end
   posi2=posi2(1);
end
```