INSTITUTO DE ENGENHARIA NUCLEAR

RT-IEN- 03/2007

Reconhecimento de Voz baseado em Análise Cepstral e Redes Neurais

por

Carlos Alexandre Fructuoso Jorge
Antônio Carlos de Abreu Mól

Serviço de Engenharia de Salas de Controle, DICH

Dezembro/2007

INSTITUTO DE ENGENHARIA NUCLEAR

Título: Reconhecimento de Voz baseado em Análise Cepstral e Redes Neurais				
Autor(es): Carlos Alexandre Fructuoso Jorge, Antônio Carlos de Abreu Mól				e-mail: calexandre@ien.gov.br, mol@ien.gov.br
Identificação: RT-IEN- 03/2007	Nº de páginas: 43	Tipo de Divulgação: Irrestrita (×) Restrita ( )	Divulgar para:	Localização: http://intranet.ien

Publicação externa associada (congresso/periódico):
Jorge, C. A. F.; Aghina, M. A. C.; Mól, A. C. A.; Pereira, C. M. N. A.; "Man Machine Interface Based on Speech Recognition", *2007 International Nuclear Atlantic Conference – INAC 2007*, DVD-ROM, 2007.

Palavras chave: Reconhecimento de voz; Processamento de sinais; Redes neurais; Interfaces homem-sistema

Resumo:
Este Relatório Técnico reporta as técnicas adotadas para o desenvolvimento de um sistema de reconhecimento de voz, realizado no Laboratório de Realidade Virtual – LABRV/IEN. O objetivo deste sistema é a implementação de uma interface homem-sistema baseado em reconhecimento de voz – reconhecimento de palavras e de locutores –, de modo a permitir ao usuário executar comandos a voz, sem a necessidade manipulação de interfaces, como teclado e mouse. Com isto, seria possível navegar em ambientes virtuais de uma forma mais natural, em frente à tela de projeção. O reconhecimento de voz é realizado em duas etapas: o pré-processamento, e o reconhecimento propriamente dito. O pré-processamento visa extrair parâmetros da voz para o reconhecimento de padrões, o que é realizado através da técnica de análise cepstral, freqüentemente citada na literatura, tanto para o reconhecimento de palavras isoladas como para reconhecimento de locutores. Para o reconhecimento de voz, é usada uma rede neural direta "backpropagation". Os resultados são apresentados e analisados. Este Relatório Técnico visa servir também como um resumo bibliográfico sobre processamento de sinais de voz, em especial sobre a técnica de análise cepstral.

Abstract:
This Technical Report shows the techniques adopted for the development of a speech recognition system, carried out at Laboratório de Realidade Virtual – LABRV/IEN. The purpose of this system is the implementation of a man-system interface based on speech recognition – word and speaker recognition –, so as the user can execute spoken commands, with no need for interfaces manipulation, such as keyboard and mouse. Then, it would be possible to navigate in virtual environments in a more natural form, in front of the projection screen. Speech recognition is achieved in two steps: the pre-processing, and the recognition itself. The pre-processing has the purpose of extracting relevant parameters for pattern recognition, what is done through the cepstral analysis technique, frequently cited in the literature, both for isolated word recognition and for speaker recognition. For the speech recognition, a feed-forward backpropagation neural network is used. The results are presented and analyzed. This Technical Report has also the purpose of serving as a bibliographical review about speech signal processing, specially about the cepstral analysis technique.

Emissão		Nome	Rubrica	Data
Data: 28/12/2007	Elaboração:	Carlos Alexandre Fructuoso Jorge, Antônio Carlos de Abreu Mól		28/12/2007
Divisão: DICH	Revisão:	Mauro Vítor de Oliveira		28/12/2007
Serviço: SEESC	Aprovação :	Paulo Victor Rodrigues de Carvalho		28/12/2007

Instituto de Engenharia Nuclear:
Via 5 s/n, Cidade Universitária, Ilha do Fundão, CEP 21945-970, CP 68.550, Rio de Janeiro – RJ - Brasil .
Tel.: 00 55 21 2209-8080 Internet: www.ien.gov.br

Índice

1 INTRODUÇÃO

Este Relatório Técnico reporta as técnicas adotadas para o desenvolvimento de um sistema de reconhecimento de voz, realizado no Laboratório de Realidade Virtual – LABRV/IEN, entre 2006 e 2007. Este sistema visa a implementação de uma interface homem-sistema baseada em reconhecimento de voz – reconhecimento de palavras e de locutores –, de modo a permitir ao usuário executar comandos a voz, sem a necessidade manipulação de interfaces como teclado e mouse. Com isto, seria possível navegar em ambientes virtuais de uma forma mais natural, em frente à tela de projeção.

O reconhecimento de voz é realizado em duas etapas: o pré-processamento, e o reconhecimento propriamente dito. O pré-processamento visa extrair parâmetros da voz para o reconhecimento de padrões, o que é realizado através da técnica de análise cepstral, freqüentemente citada na literatura, tanto para o reconhecimento de palavras isoladas como para reconhecimento de locutores. Para o reconhecimento de voz, é usada uma rede neural direta *backpropagation*.

Este primeiro trabalho é dedicado ao reconhecimento de palavras isoladas, onde o objetivo é a identificação de comandos falados. Porém, as mesmas técnicas podem ser usadas para o reconhecimento de locutores, com pequenas modificações.

A Fig. 1 mostra o diagrama de um sistema genérico de reconhecimento de voz, onde os dois primeiros blocos, a partir da esquerda, compreendem o pré-processamento.

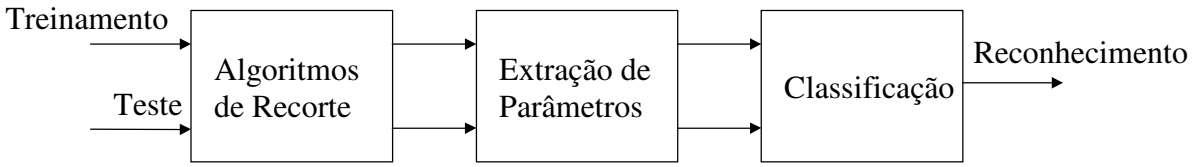


Fig. 1- Sistema de reconhecimento de palavras ou locutores.

O recorte é necessário para isolar, do sinal de voz gravado, o trecho que corresponde somente à palavra falada, eliminando o ruído de fundo presente no início e no fim da locução. Alguns métodos são descritos na literatura, como o algoritmo de Rabiner e Sambur [1], que utiliza informações da Energia e da Taxa de Cruzamento por Zero, de tempo curto, além da média e variância do ruído; há também outros algoritmos derivados deste. Entretanto, para este trabalho, foi adotada a abordagem descrita em [2], que utiliza somente a Energia de tempo curto, dada sua simplicidade de implementação, e os bons resultados reportados nesta referência.

A extração visa obter parâmetros que permitam modelar o processo que gera fala, ou de outra forma modelar o trato vocal, e que possam ser utilizados

para o reconhecimento de palavras ou de locutores. Diversos tipos de parâmetros são reportados na literatura, dentre os quais pode-se citar: os coeficientes do Preditor Linear (*Linear Predictor Coefficients* – LPC) [1], os coeficientes Cepstrais [3] (a partir da Transformada de Fourier), e outros métodos derivados, como os coeficientes LPC-Cepstrais [4], os Mel-Cepstrais [4], e as raízes (pólos) do filtro Preditor [5], dentre outros. Neste trabalho, optou-se pelos coeficientes Cepstrais (a partir da Transformada de Fourier), dada a simplicidade de implementação para a obtenção de tais parâmetros.

Quatro comandos foram considerados: "abaixo", "acima", "direita" e "esquerda". Todas as etapas de pré-processamento e geração do conjunto de entrada para a rede neural foram realizados com o uso da ferramenta matemática Scilab[1] [6].

2 MODELAGEM DA VOZ

A Fig. 2 apresenta um modelo simplificado da geração de voz. Um modelo mais detalhado pode ser encontrado em [1], o qual inclui outras funções de transferência, como o Pulso glotal G(z) e a Radiação R(z), e ganhos separados para os sons sonoros e surdos.

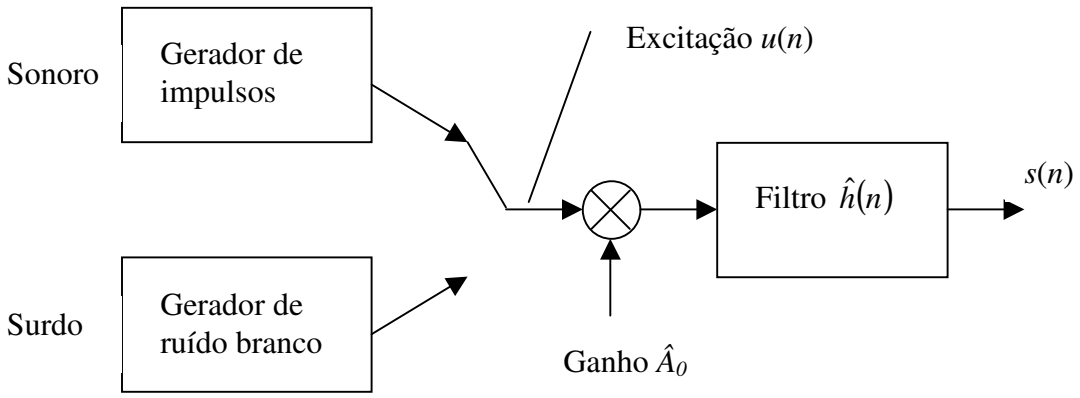


Fig. 2- Modelagem da voz.

Os sinais de voz podem ser classificados, basicamente, em dois tipos:

[1] O Scilab é um software livre, aberto (*open source*), para computação numérica. Desenvolvido pelo *l'Institut National de Recherche en Informatique et en Automatique – INRIA*, é mantido atualmente pelo *Scilab Consortium*. Assemelha-se ao Matlab, podendo inclusive importar seus arquivos de programas (".m").

1- Sonoros ou vozeados: constituem as vogais, sendo gerados pela vibração das cordas vocais tensionadas, durante a passagem de ar. Podem ser modelados como sinais quase periódicos, com freqüência fundamental correspondendo ao tom de voz (*pitch*). São caracterizados por elevada energia e baixa taxa de cruzamento por zeros, de tempo curto [1].

2- Surdos ou não vozeados: nesta classe estão agrupados, por simplicidade, os sons fricativos ou bilabiais, e os sons plosivos. Em ambos os casos, estes sons podem ser modelados como sinais ruidosos de banda larga. São caracterizados por baixa energia e alta taxa de cruzamento por zero, de tempo curto [1].

O sinal $u(n)$ é a excitação, podendo ser constituído por sinais sonoros ou surdos. O filtro $\hat{h}(n)$, variável no tempo, modela a resposta impulsiva do trato vocal. O sinal de voz é gerado por um processo lentamente variável no tempo, que pode entretanto ser considerado estacionário em intervalos de curta duração, aproximadamente entre 10 e 30 ms [1]. Portanto, para ser processado, o sinal deve ser segmentado em intervalos aproximadamente nesta faixa.

Para a segmentação, podem ser utilizadas janelas de diferentes tipos, como: retangular, de Hamming, de Hanning, de Kaiser, entre outras [1], [3], sendo a de Hamming a mais utilizada para processamento de sinais de voz [1]. A partir do processamento por janelas, podem ser extraídos um ou mais parâmetros por segmento, como a energia de tempo curto, por exemplo [1].

Se o sinal for simplesmente dividido em segmentos adjacentes, implicitamente está sendo aplicada a janela retangular sem superposição, que mantém as características temporais do sinal, sem atenuação das amostras. Entretanto, os segmentos obtidos com a janela retangular apresentam variações abruptas em seus extremos, o que gera altas freqüências espúrias no cálculo do espectro do sinal.

Por outro lado, a janela de Hamming, assim como também as outras citadas anteriormente (Hanning, Kaiser), atenua o sinal nos extremos do segmentos, reduzindo as variações abruptas do sinal. Neste caso, porém, faz-se necessário utilizar superposição de janelas, de modo a compensar estas atenuações. O número de janelas resulta maior, porém esta redundância compensa a perda de informação devido às atenuações causadas pelo janelamento.

3 RECORTE

Neste trabalho, para o recorte do sinal, utiliza-se janela retangular sem superposição, uma vez que não está sendo calculado o espectro nesta etapa.

O recorte é efetuado baseado apenas no cálculo da Energia de tempo curto [1], dada pela Eq. (1).

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

onde:

- *s*: sinal de voz,
- *E*: energia calculada para cada segmento *m*,
- *N*: número de amostras por segmento.

São eliminados os segmentos com energia menor que 1% da energia máxima na locução, conforme [2].

4 EXTRAÇÃO DE PARÂMETROS

4.1 Janelamento

Para a extração de parâmetros, foi utilizada a janela de Hamming, dada pela Eq. (2), com superposição de 75 %, valor próximo ao adotado em [7], [8], afim de minimizar as perdas indesejadas, conforme explicado no item 2. A Fig. 3 mostra um exemplo de janela de Hamming de tamanho 221.

$$hm(n) = \begin{cases} 0{,}54 - 0{,}46\cos(2\pi n/(N-1)), & 0 \le n \le N-1 \\ 0, & \text{caso contrário} \end{cases} \tag{2}$$

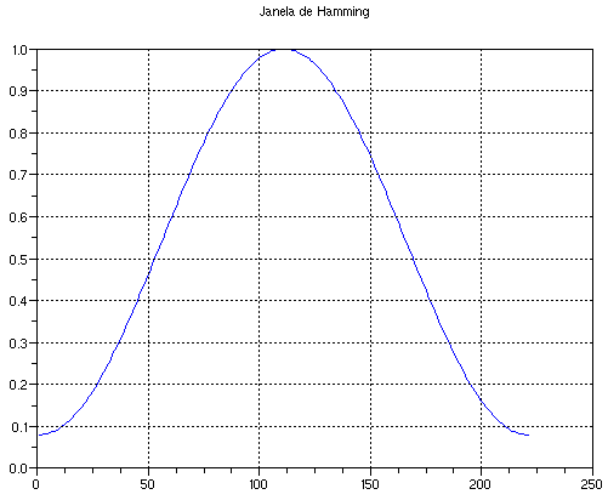


Fig. 3 - Janela de Hamming.

Um aspecto importante a considerar, é que a etapa de reconhecimento de padrões (classificação) é realizada por um classificador neural, o que requer que o número de segmentos utilizados na etapa de extração de parâmetros seja sempre constante, independentemente do tamanho da palavra recortada, pois o número de entradas da rede neural adotada (direta, *backpropagation*) não varia.

Há algumas possíveis soluções para este requisito [7]: ajustamento temporal; superposição variável; ou janelas adaptativas. No primeiro caso, efetua-se interpolação ou decimação do sinal de voz recortado, de modo a resultar em um número fixo de amostras, o que tem a desvantagem de alterar a taxa de amostragem do sinal original [7]. No segundo, a taxa de superposição é variável, o que pode resultar em perda de informação nas fronteiras entre segmentos, quando a taxa de superposição for reduzida.

A solução adotada utiliza janelas adaptativas [7], onde tanto o número de janelas quanto a taxa de superposição são fixadas, porém o tamanho das janelas é variável. Esta abordagem proporciona uma vantagem adicional: uma mesma palavra pode ser pronunciada de forma mais lenta ou mais rápida, em locuções diferentes; considerando que o alongamento temporal ocorra de modo uniforme, as janelas adaptam-se de modo proporcional ao tamanho dos fonemas [7].

Foi fixado o número de janelas como sendo 120 [7], [8], o que resulta em janelas de aproximadamente 20 ms, para palavras recortadas com duração entre 0,5 e 1 s.

A Fig. 4 ilustra o procedimento adotado, para segmentação do sinal com janelas adaptativas, com superposição de 75 %.

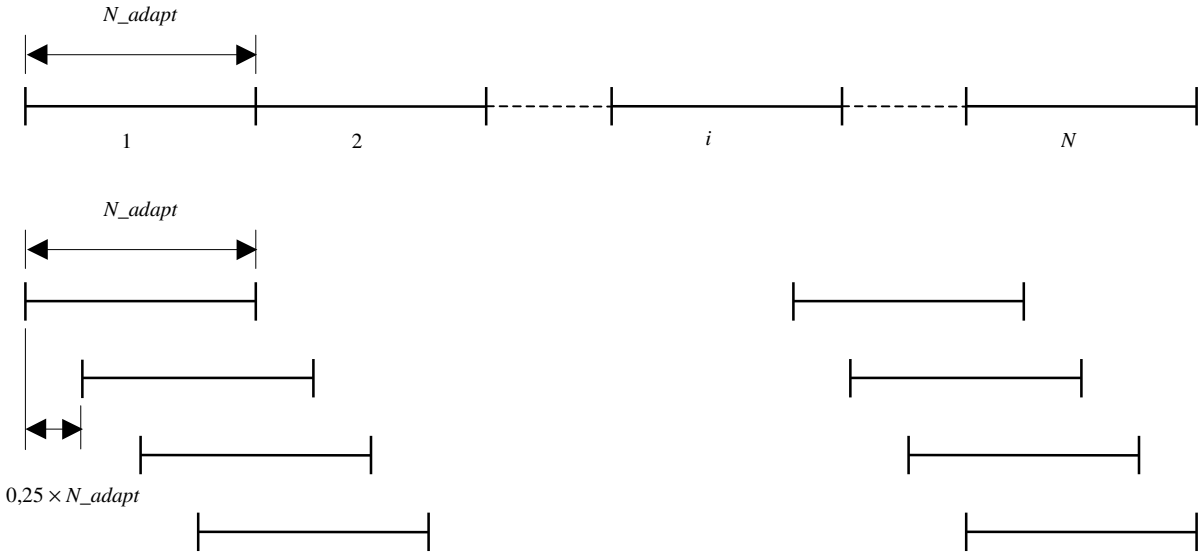


Fig. 4 - Janelamento com superposição de 75 %.

O sinal de voz a ser segmentado é ilustrado na parte superior da figura; a segmentação é ilustrada na parte inferior, onde cada segmento, com tamanho *N_adapt*, é tomado do sinal com deslocamentos de ¼ do tamanho do segmento, resultando em uma superposição de 75 %. *N* é o número de divisões correspondentes no sinal. Após este processo, cada segmento é multiplicado pela janela de Hamming de mesmo tamanho.

Caso a Transformada de Fourier seja obtida através da FFT, ao invés da DFT, devem ser acrescentados zeros em cada segmento (*zero-padding*), até completar um número de pontos igual a uma potência de 2; neste trabalho, foi adotada FFT de 512 pontos.

4.2 Análise Cepstral

A Análise Cepstral é parte de uma classe de técnicas de processamento de sinais conhecida como Processamento Homomórfico [3], que têm aplicação na desconvolução de sinais. Para processamento de sinais de voz, o cepstrum real mostra-se suficiente, não sendo necessário o cálculo do cepstrum complexo.

Os parâmetros extraídos para classificação são os coeficientes cepstrais. Estes coeficientes poderiam ser obtidos a partir dos coeficientes LPC, porém neste trabalho são obtidos a partir da Transformada de Fourier [3].

A Fig. 5 mostra, de forma mais simplificada, o efeito do trato vocal na geração do sinal de voz. O sinal de excitação $u(n)$ é convoluído com a resposta impulsiva $\hat{h}(n)$ do trato vocal, gerando o sinal de voz $s(n)$, conforme a Eq. (3).

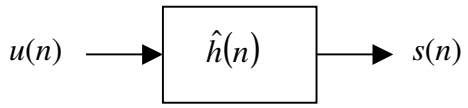


Fig. 5 - Modelagem simplificada da voz.

$$s(n) = u(n) * \hat{h}(n) \tag{3}$$

A desconvolução pode ser obtida através do logaritmo da Transformada de Fourier do sinal de voz *s*(*n*), para cada segmento, conforme mostrado a seguir.

$$\begin{aligned} C_S\left(e^{j\omega}\right) &= \log\left|S\left(e^{j\omega}\right)\right| \\ &= \log\left|U\left(e^{j\omega}\right)\hat{H}\left(e^{j\omega}\right)\right| \\ &= \log\left|U\left(e^{j\omega}\right)\right| + \log\left|\hat{H}\left(e^{j\omega}\right)\right| \\ &= C_U\left(e^{j\omega}\right) + C_{\hat{H}}\left(e^{j\omega}\right) \end{aligned} \tag{4}$$

onde:

- $S(e^{j\omega})$: Transformada de Fourier do sinal de voz (para cada segmento),
- $U(e^{j\omega})$: Transformada de Fourier da excitação,
- $\hat{H}(e^{j\omega})$: Transformada de Fourier da resposta impulsiva do trato vocal,
- $C_S(e^{j\omega})$: espectro modificado do sinal de voz,
- $C_U(e^{j\omega})$: espectro modificado da excitação,
- $C_{\hat{H}}(e^{j\omega})$: espectro modificado impulsiva do trato vocal.

Aplicando a Transformada Inversa de Fourier, obtém-se os coeficientes cepstrais no domínio das quefrências (*k*).

$$c_S(k) = IDFT\{C_S(e^{j\omega})\} = c_U(k) + c_{\hat{H}}(k) \tag{5}$$

onde:

- $c_S(k)$: cepstrum do sinal de voz,
- $c_U(k)$: cepstrum da excitação,
- $c_{\hat{H}}(k)$: cepstrum da resposta impulsiva do trato vocal.

Conforme mostram a Eq. (4) e a Eq. (5), a excitação e a resposta impulsiva, antes convoluídas, após este processo passam a ser aditivas. Além disso, os coeficientes $c_{\hat{H}}(k)$ situam-se nas baixas quefrências, enquanto os coeficientes $c_U(k)$ situam-se nas altas [3]; com exceção do primeiro coeficiente (índice zero), que faz parte da excitação [9], [4]. Portanto, é possível separá-las através de uma liftragem passa-baixas no domínio das quefrências. Neste caso, é utilizado geralmente um liftro senoidal [10], [4], dado pela Eq. (6), cuja resposta é mostrada na Fig. 6.

$$l(k) = \begin{cases} 1 + \dfrac{L}{2}\operatorname{sen}\left(\dfrac{\pi k}{L}\right), & k = 0,1,2,...,L \\ 0, & \text{qualquer outro valor de } k \end{cases} \tag{6}$$

onde:

- *l*(*k*): liftro passa-baixas,
- *L*: nº de coeficientes cepstrais a serem obtidos.

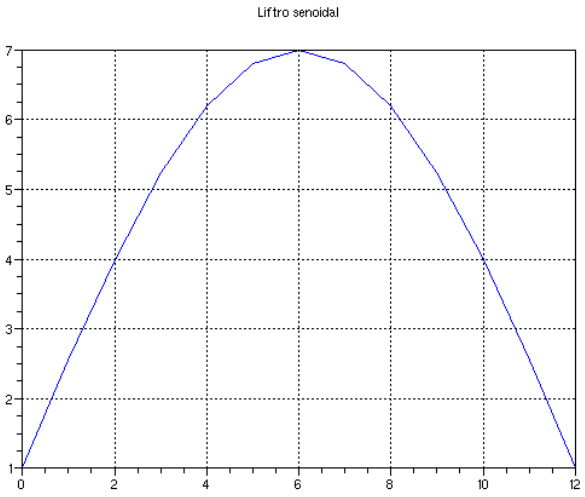


Fig. 6 - Liftro senoidal.

O tamanho do liftro deve ser tal que elimine a freqüência de *pitch*; na prática, costuma-se obter entre 8 e 15 coeficientes cepstrais [4]. Neste trabalho, são extraídos os doze primeiros coeficientes, já que no caso de usar-se análise LPC, são usados também doze destes coeficientes.

A Fig. 7 mostra um diagrama em blocos da análise cepstral.

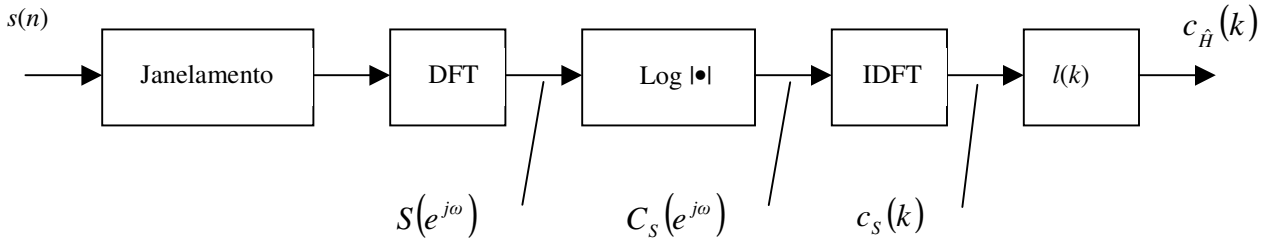


Fig. 7 - Análise cepstral.

5 RESULTADOS DO PRÉ-PROCESSAMENTO

As locuções dos quatro comandos: "abaixo", "acima", "direita" e "esquerda", foram gravadas, diversas vezes cada locução, por cada um de quatro locutores.

Os sinais foram amostrados com as seguintes características: mono, taxa de amostragem de 11025 Hz e quantização de 16 bits por amostra. Após a aquisição, os sinais foram normalizados para amplitudes entre – 0,5 e 0,5.

Conforme já descrito, foi fixado o número de 120 segmentos, obtidos com janelas de Hamming, com superposição de 75%, Inicialmente, o que resulta em segmentos com duração em torno de 20 ms.

A seguir, são mostrados, a título de exemplo, os resultados para uma locução do comando "acima". A Fig. 8 mostra o sinal original da locução do comando "acima", e a análise da energia de tempo curto, enquanto a Fig. 9 mostra os segmentos considerados válidos (com Energia de tempo curto acima de 1% da Energia máxima ao longo da locução), e o sinal recortado. Nota-se, na Fig. 8 e na Fig. 9, que os sinais são quase periódicos nos trechos sonoros, que correspondem às vogais e às consoantes vozeadas, e ruidosos nos trechos surdos.

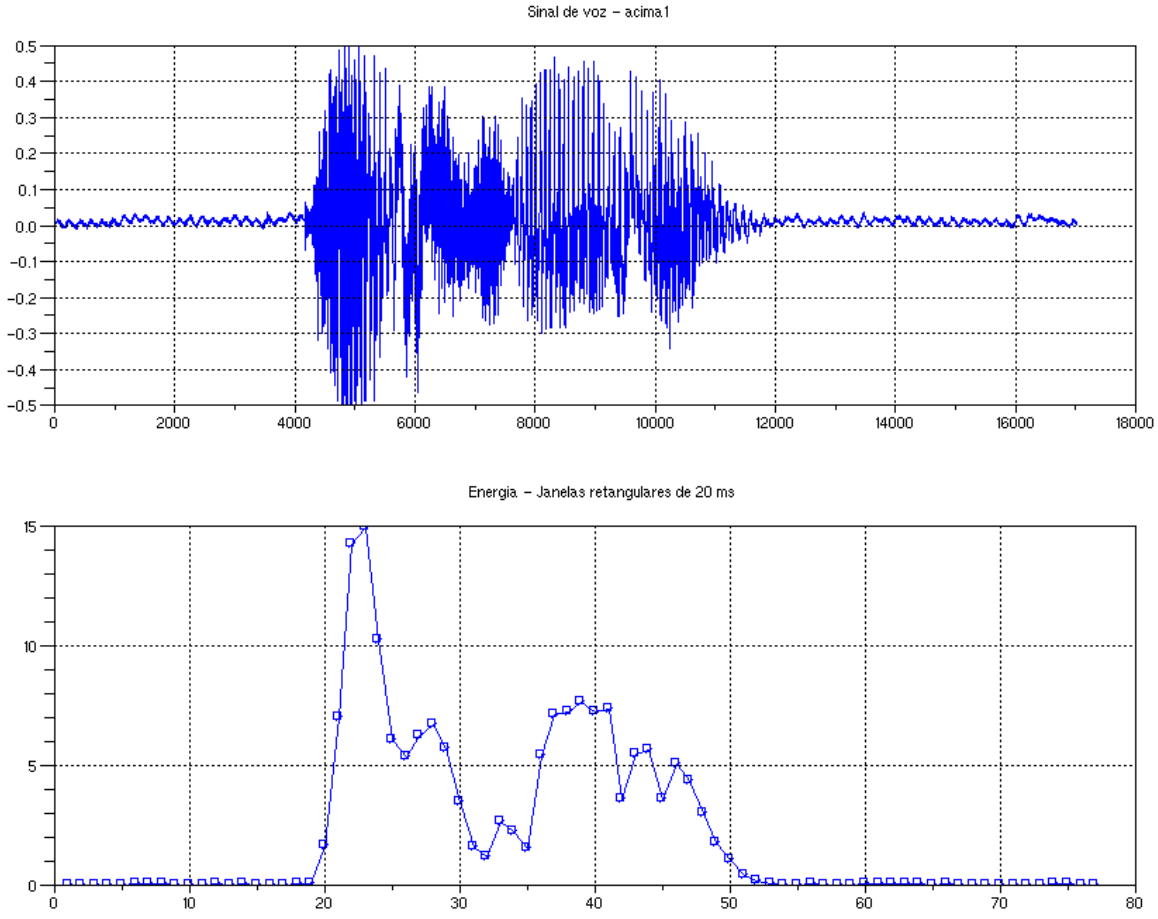


Fig. 8 - Sinal de uma das locuções da palavra "acima", original e Energia de tempo curto.

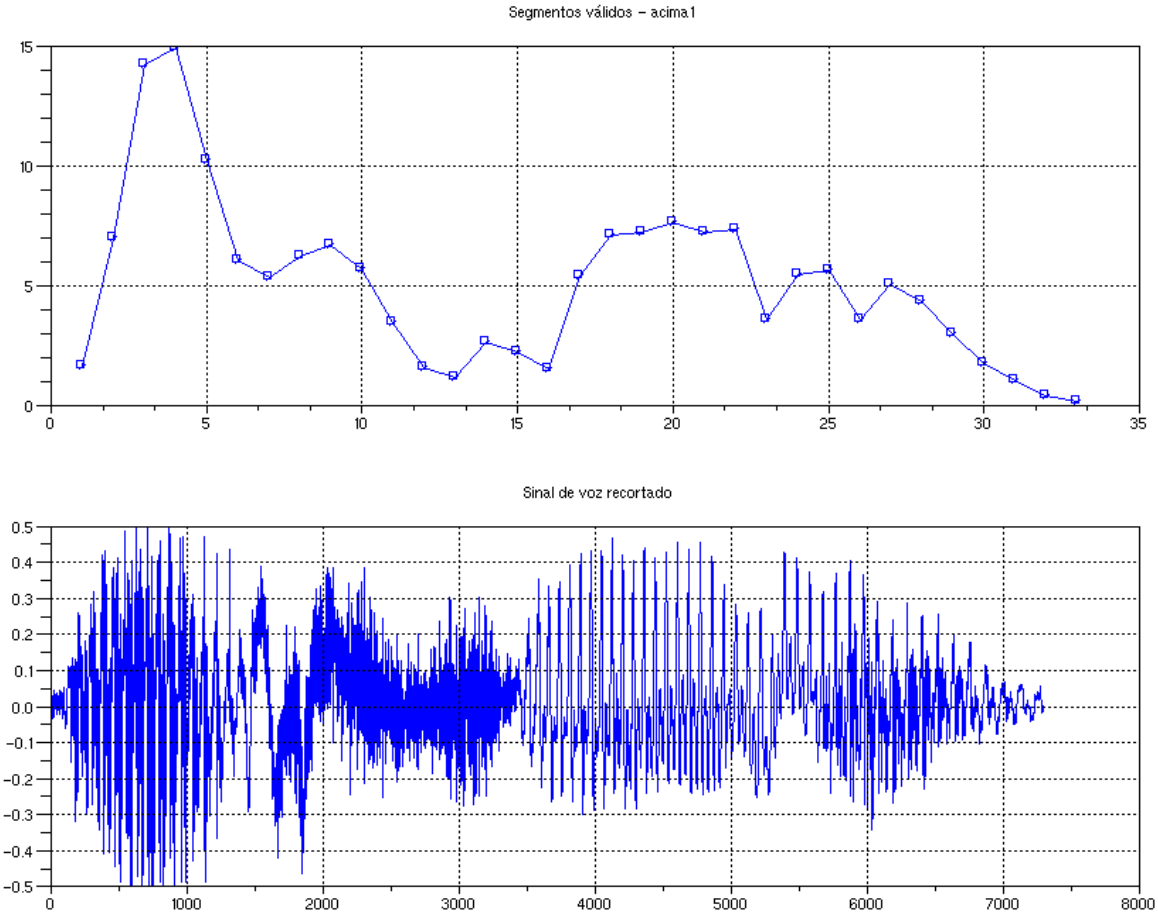


Fig. 9 - Segmentos válidos, e sinal da palavra "acima" recortado.

A Fig. 10 e a Fig. 11 mostram, respectivamente, a Transformada de Fourier (freqüência dada em kHz), e os coeficientes cepstrais, ao longo dos segmentos recortados para este mesmo exemplo; o primeiro coeficiente cepstral é desconsiderado.

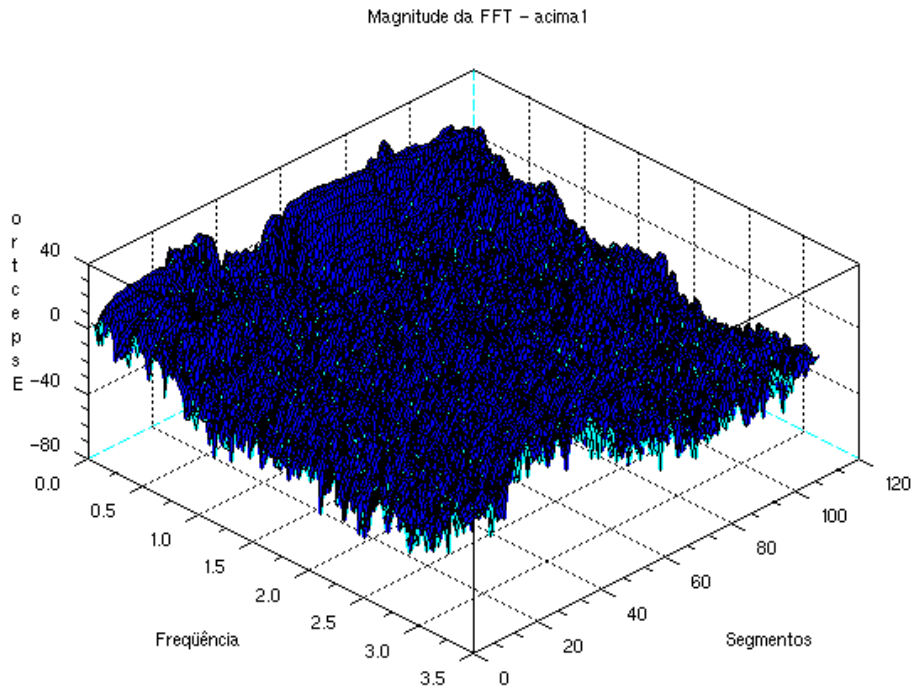


Fig. 10 - Transformada de Fourier da locução da palavra "acima", mostrada anteriormente.

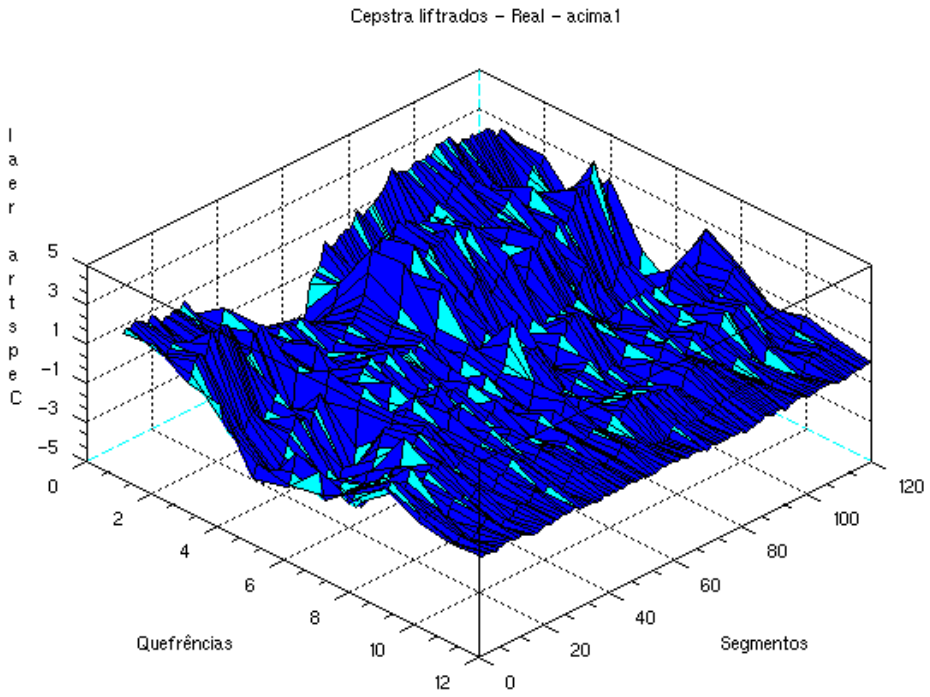


Fig. 11 - Coeficientes cepstrais desta locução da palavra "acima".

5.1 Análise Qualitativa

Os gráficos seguintes mostram os resultados da análise cepstral, agrupados para as locuções de cada comando falado, em uma primeira análise qualitativa, onde são observadas as formas dos sinais nos gráficos dos coeficientes cepstrais, em função das quefrências e dos segmentos. É possível notar a semelhança destes sinais, para as locuções de um mesmo comando, indicando a existência de uma assinatura. Das diversas locuções gravadas de cada comando, são mostradas, a título de exemplo, apenas as 4 primeiras locuções de cada comando.

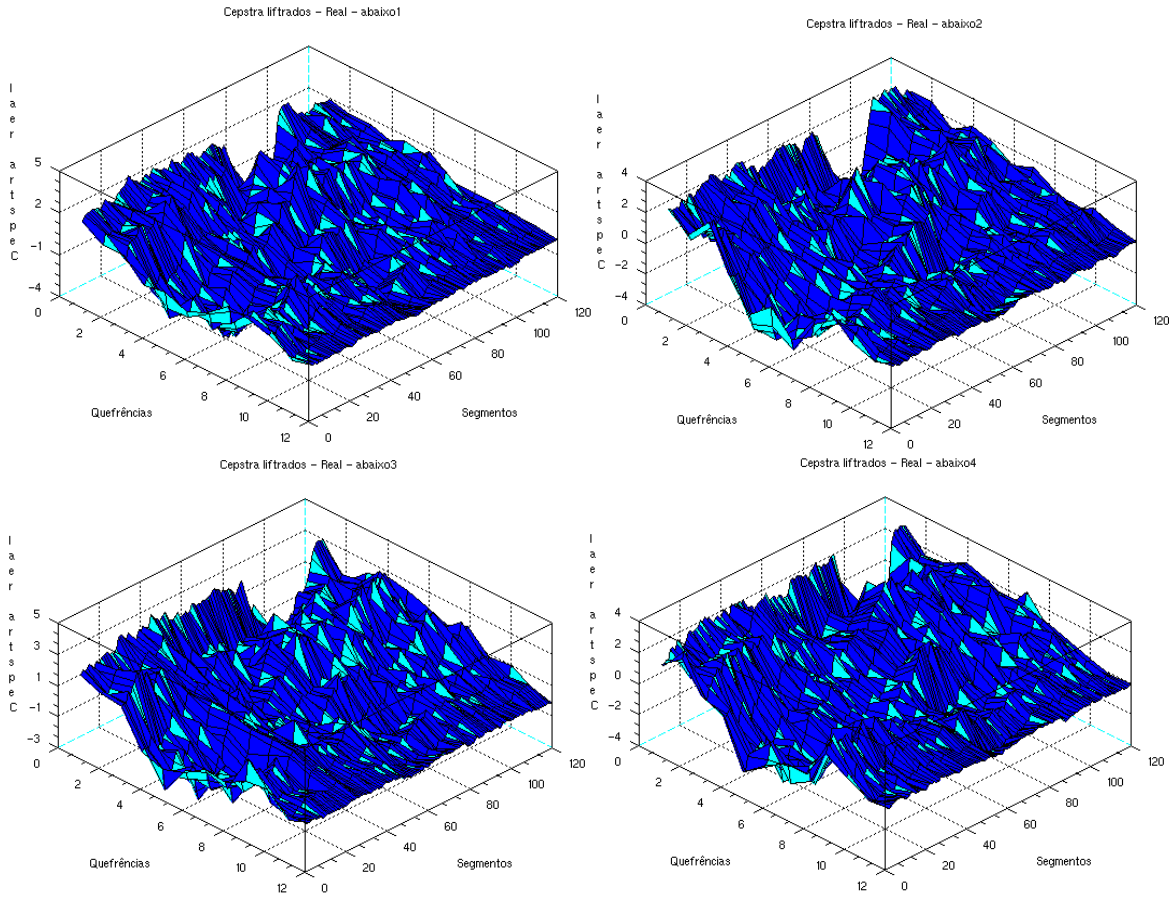


Fig. 12 - Coeficientes cepstrais para locuções do comando "abaixo".

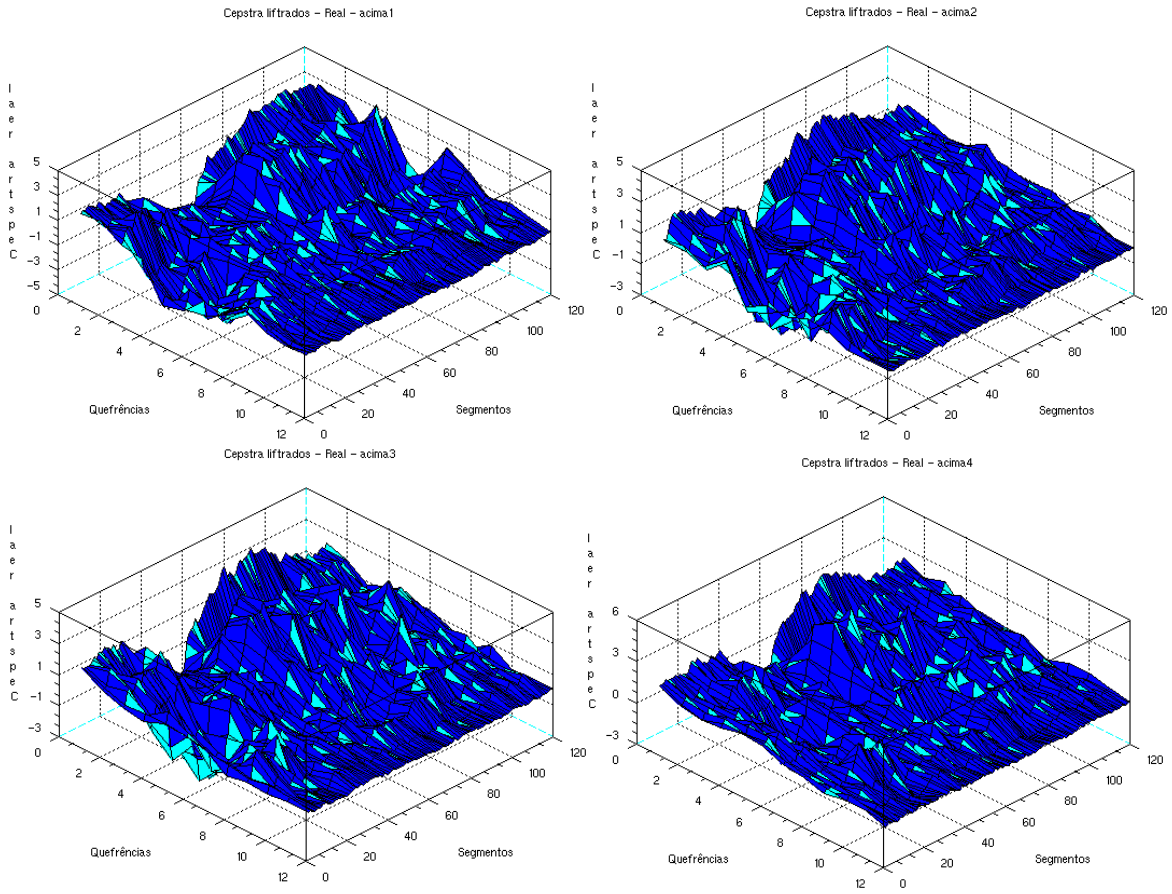


Fig. 13 - Coeficientes cepstrais para locuções do comando "acima".

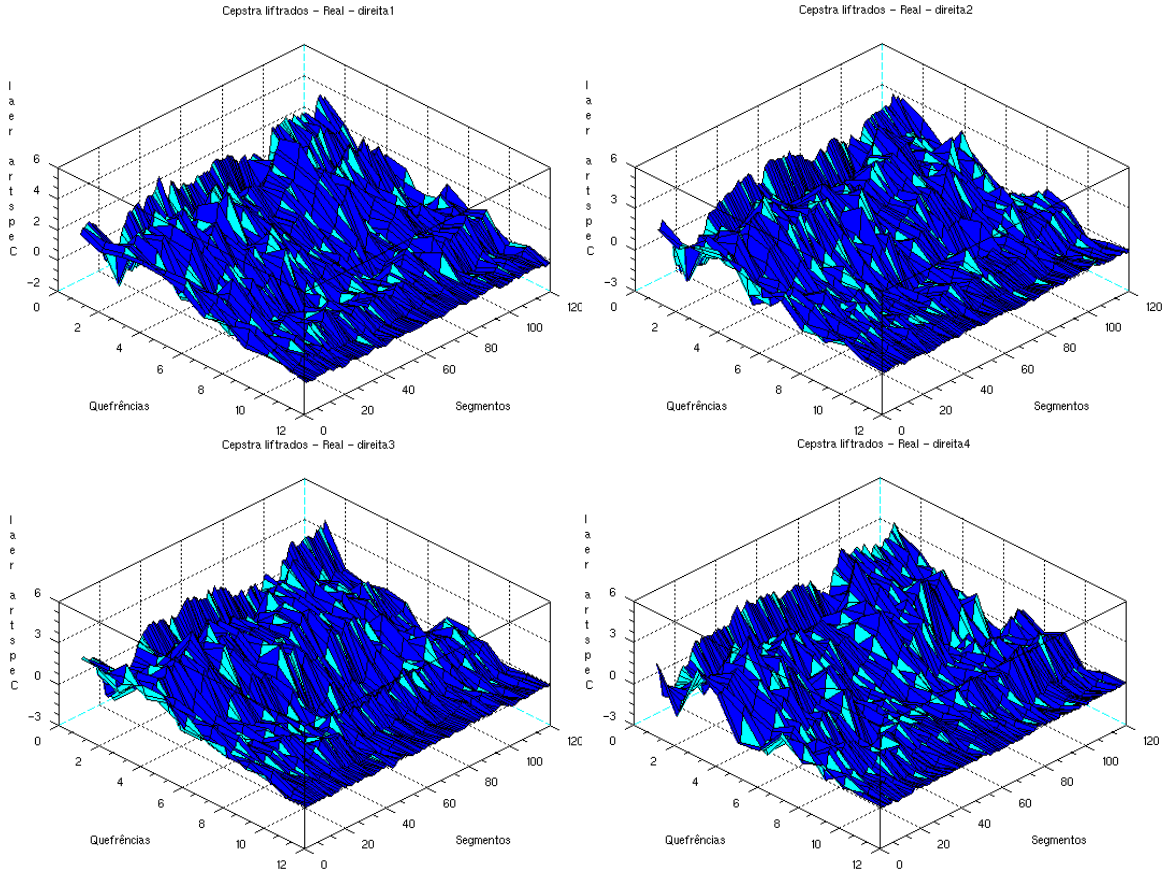


Fig. 14 - Coeficientes cepstrais para locuções do comando "direita".

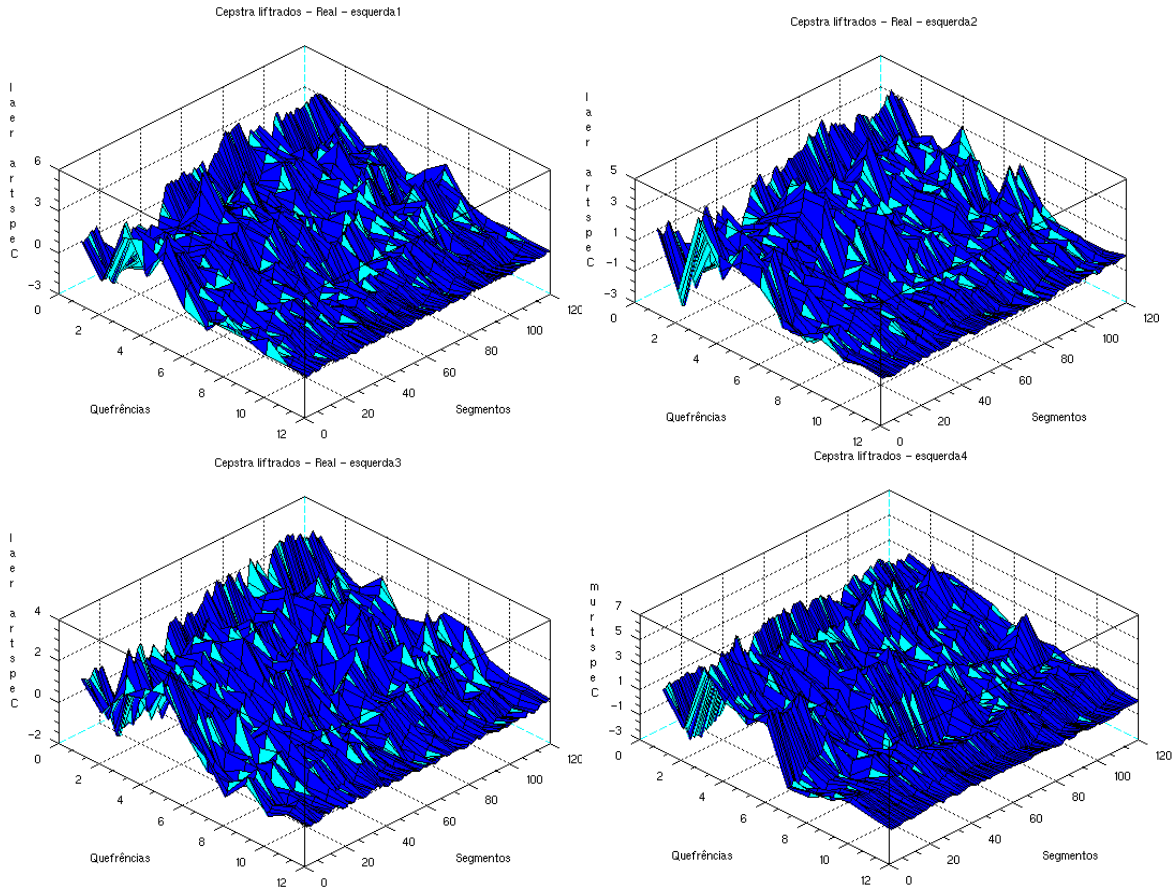


Fig. 15 - Coeficientes cepstrais para locuções do comando "esquerda".

6 RESULTADOS DO RECONHECIMENTO DE VOZ

Para a classificação, foi utilizada uma rede neural direta treinada com algoritmo *backpropagation*, com 3 camadas: uma camada de entrada, uma camada escondida e uma de saída, as duas últimas contendo neurônios com não linearidade do tipo sigmoidal.

As entradas da rede neural correspondem aos coeficientes cepstrais ao longo de todos os segmentos. Análises preliminares foram realizadas, considerando todos os (doze) coeficientes cepstrais, e o segundo utilizando somente os três coeficientes cepstrais mais discriminantes, conforme explicado[2] no item Apêndice C.

Conforme pode ser notado, não há diferença significativa entre os resultados dos dois testes, quando considerado o conjunto total de amostras, o

[2] A análise para verificar quais os coeficientes cepstrais mais discriminantes foi deixada para um Apêndice, uma vez que foi usada apenas em um estágio inicial do trabalho, não tendo sido mais usada posteriormente.

que indica que realmente os 3 coeficientes cepstrais mais significativos permitem uma classificação com aproximadamente o mesmo índice de acertos que os obtidos com o conjunto total de (12) coeficientes na entrada.

Não se pode afirmar o quanto uma abordagem é superior ou não à outra, havendo necessidade de efetuar mais testes. Conforme indicado no Apêndice C.4, o menor valor do Discriminante Linear de Fischer corresponde a 13,5 % do máximo. Poderiam ser realizados outros testes variando-se o limiar mínimo, afim de selecionar quais entradas poderiam ser utilizadas.

A solução adotada foi simplesmente utilizar todos os coeficientes cepstrais afim de que a rede neural possa aprender por si mesma o problema em questão, a partir de todas as entradas. A geração das entradas é detalhada no Apêndice D.

A entrada da rede neural, para o caso do uso de todos os coeficientes cepstrais, resulta em:

- 12 coeficientes cepstrais $\times$ 120 segmentos = 1440 entradas.

A camada de saída da rede neural é composta por 4 neurônios, correspondendo às quatro classes (comandos); a saída correspondente à classe identificada é ativada com valor "1", permanecendo as demais desativadas, ou seja, com valor "0".

A camada escondida foi gerada automaticamente, contendo 187 neurônios.

6.1 Reconhecimento de Palavras

Após estes testes preliminares, foi realizado o treinamento de uma rede neural, da seguinte forma:

- Quatro locutores foram considerados, os sinais misturados;
- Cada locutor gravou, por diversas vezes[3], todos os quatro comandos.

Logo, o conjunto completo de dados contém gravações dos quatro comandos, falados pelos quatro locutores, várias repetições cada um.

O conjunto completo de dados foi também sorteado nos três subconjuntos seguintes: treinamento, teste e produção, com as seguintes percentagens: 70 % para treinamento, 15 % para teste e 15 % para produção.

Nesta análise, são mostrados um a um os resultados obtidos pela rede neural. A Tab. 1 mostra os resultados para o conjunto de produção, enquanto a Tab. 2 mostra os resultados para o conjunto de teste. Os comandos "*abaixo*", "*acima*", "*direita*" e "*esquerda*", são representados respectivamente por "C1", "C2", "C3" and "C4".

[3] Dos quatro locutores, um deles havia gravado quinze locuções de cada comando, enquanto os outros três gravaram apenas dez repetições de cada comando. Para este primeiro trabalho, foram aproveitados todos os dados disponíveis, apesar da diferença no número de repetições de comandos para um dos locutores.

Tab. 1 - Conjunto de produção.

Saídas desejadas				Saídas obtidas			
C1	C2	C3	C4	C1	C2	C3	C4
1	0	0	0	0.993787	0.093789	0.190655	0
1	0	0	0	0.856958	0.051491	0.098181	0.447898
1	0	0	0	0.971785	0	0.18834	0.210627
1	0	0	0	1	0	0	0.513327
1	0	0	0	0.972453	0	0	0.226166
0	1	0	0	0	0.759732	0.065483	0
0	1	0	0	0	0.850423	0	0
0	0	1	0	0.009139	0.231809	0.264204	0
0	0	1	0	0	0	0.714916	0.185912
0	0	1	0	0	0	0.810875	0.07209
0	0	0	1	0	0	0.07837	1
0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0.730365
0	0	0	1	0	0.015004	0.076821	0.921378
0	0	1	0	0	0.036012	0.986411	0.183953
0	0	0	1	0	0	0.175193	0.922909
1	0	0	0	1	0	0	0.102289
0	1	0	0	0	1	0	0.134824
0	0	1	0	0.086702	0.146146	0.79823	0.035219
0	0	1	0	0	0.035443	0.876442	0.319351
1	0	0	0	1	0.160665	0.02948	0
1	0	0	0	1	0	0.052995	0
0	0	1	0	0.026809	0	1	0.052781
0	0	1	0	0.037334	0	1	0
0	0	1	0	0	0	0.999491	0.023369
0	0	0	1	0.045213	0	0	1

Tab. 2 - Conjunto de teste.

Saídas desejadas				Saídas obtidas			
C1	C2	C3	C4	C1	C2	C3	C4
1	0	0	0	0.908805	0	0.078555	0
1	0	0	0	1	0.077061	0	0.067706
0	1	0	0	0	0.982088	0	0
0	1	0	0	0	1	0	0.188277
0	0	1	0	0.063464	0	0.512572	0.220332
0	0	1	0	0.192168	0	0.61476	0.030474
0	0	1	0	0	0.118288	0.911985	0.112232
0	0	0	1	0.004041	0	0.047082	0.781381
0	0	0	1	0	0	0.048695	0.952201
1	0	0	0	0.88125	0.040296	0.071579	0.025161
0	1	0	0	9.59E-05	0.638531	0.196255	0
0	1	0	0	0.134635	0.779713	0	0
0	0	1	0	0	0.083125	0.761442	0
0	0	0	1	0	0.021612	0.294693	1
0	0	0	1	0	0	0.016211	1
0	0	0	1	0	0.084402	0.066101	0.592593
0	1	0	0	0.033442	0.895954	0	0
0	1	0	0	0	0.94737	0	0
0	1	0	0	0.047789	0.910425	0	0.104964
0	0	1	0	0.018622	0	0.873061	0.153087
0	0	1	0	0.018332	0	0.895765	0.162628
0	0	1	0	0	0.241774	0.991409	0
0	0	0	1	0	0	0.053508	1
0	0	0	1	0	0.036555	0.226021	1
0	0	0	1	0	0.229024	0.264163	0.813907
0	0	0	1	0.04035	0	0.107756	0.759386

Considerando que a resposta correta da rede neural é a do neurônio vencedor, estes resultados são encorajadores, pois a rede neural acertou a resposta para todos os dados dos conjuntos de teste e de produção. Em alguns casos, a rede neural deu valores próximos nas saídas de dois neurônios, o caso mais importante verificado na amostra número 8 da Tab. 1. Mas mesmo assim, o neurônio correto foi o vencedor.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste primeiro trabalho, o sistema proposto foi capaz de fornecer bons resultados para amostras não usadas durante o treinamento. No item 6.1 é

mostrado o resultado de uma rede neural treinada com sinais de voz provenientes de quatro locutores, misturados.

A rede neural usada é uma rede direta, treinada com o algoritmo de "*backpropagation*", porém melhoras podem ser feitas tanto no tipo de rede, como na estratégia de treinamento, visando melhorar os resultados.

Por outro lado, melhorias podem ser implementadas na etapa de pré-processamento, com o uso de outras técnicas, como: LPC, LPC-Cepstral, Mel-Cepstral, ou uma combinação de diferentes parâmetros, através da análise dos mais discriminantes, pelo cálculo do Discriminante Linear de Fischer [11], [8].

Como os parâmetros cepstrais contêm informação sobre o trato vocal, este sistema pode facilmente ser adaptado para a tarefa de reconhecimento de locutores, bastando apenas trocar as colunas de saída de "comandos" para "locutores" a serem identificados.

REFERÊNCIAS

[1] Rabiner, L. R.; Schafer, R. W.; *Digital Processing of Speech Signals*, Prentice-Hall, Inc., 1978.

[2] Pinto, R. G.; Pinto, H. L.; Calôba, L. P.; "Using Neural Networks for Automatic Speaker Recognition", *Proceedings of the 38th (IEEE) Midwest Symposium on Circuits and Systems*, v. 2, pp. 1078-1080, 1995.

[3] Oppenheim, A. V.; Schafer, R. W.; *Discrete-Time Signal Processing*, Prentice-Hall, Inc., 1989.

[4] Lima, A. A. de; *Análises Comparativas em Sistemas de Reconhecimento de Voz*, Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 2000.

[5] Pinto, H. L.; Pinto, R. G.; Calôba, L. P.; "A Speaker Verification Method Using LPC Singularity Locations", *Records of the Bi-Annual International Telecommunications Symposium, ITS/ROC&C'96*, pp. 39-43, 1996.

[6] www.scilab.org.

[7] Diniz, S.; Thomé, A. G.; "Uso de Técnica Neural para o reconhecimento de Comandos à Voz", *IV Simpósio Brasileiro de Redes Neurais*, pp. 23-26, 1997.

[8] Diniz, S. dos S.; *Uso de Técnicas Neurais para o Reconhecimento de Comandos à Voz*, Tese de Mestrado, Departamento de Engenharia Elétrica, IME, 1997.

[9] Deller, Jr., J. R.; Proakis, J. G.; Hansen, J. H.; *Discrete-Time Processing of Speech Signals*, MacMillan, 1993.

[10] Huang, B.-H.; Rabiner, L. R.; Wilpon, J. G.; "On the Use of Bandpass Liftering in Speech Recognition", *IEEE Transactions on Acoustics, Speech and Signal Processing*, v. ASSP-35, n. 7, pp. 947-854, 1987.

[11] Duda, O. R.; Hart, E. P.; *Pattern Classification and Scene Analysis*, Wiley-Interscience, 1973.

Apêndice A – RECORTE DO SINAL DE VOZ

A.1 Pseudo-algoritmo para Recorte

Ler sinal de voz a partir do arquivo no formato ".wav", obtendo também o nº de amostras, e a freqüência de amostragem.

Normalizar amplitude para os limites [-0,5; 0,5].

Calcular tamanho (nº de amostras) N de cada segmento de 20 ms;

Segmentar o sinal (janelas retangulares sem superposição), truncando-o (eliminando algumas das últimas amostras do sinal).

Calcular a Energia de tempo curto para cada segmento.

Eliminar os segmentos com Energia menor que 1% da Energia máxima do sinal, gerando novas seqüências de energia por segmentos e de sinal de voz recortados, salvando este último em novo arquivo ".wav".

Apêndice B – EXTRAÇÃO DE PARÂMETROS

B.1 Extração de Parâmetros - detalhamento

A abordagem adotada para a segmentação é descrita detalhadamente a seguir. O número de divisões correspondente à segmentação com superposição de 75 % é dado por:

$$N = \frac{n\_jan\_sup - 1}{4} + 1$$

onde:

- *n_jan_sup*: nº de janelas adaptativas,
- *N*: nº de divisões correspondentes.

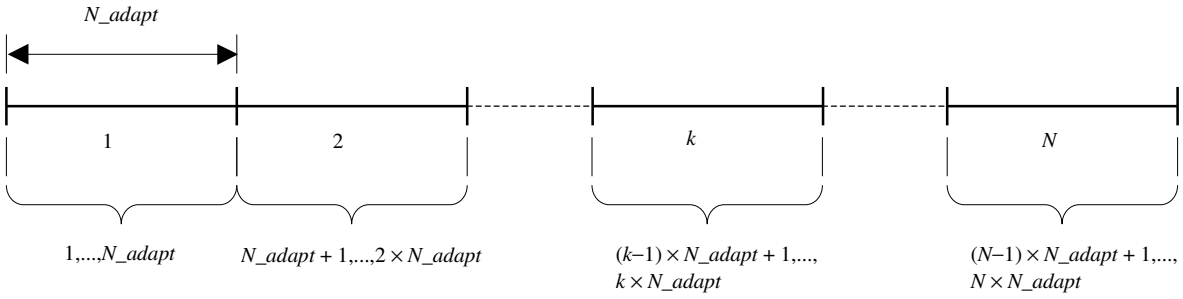
O número de janelas adaptativas desejado, *n_jan_sup*, é fixado neste trabalho em 120 janelas; logo, o número *N* resulta em torno de ¼ de *n_jan_sup*.

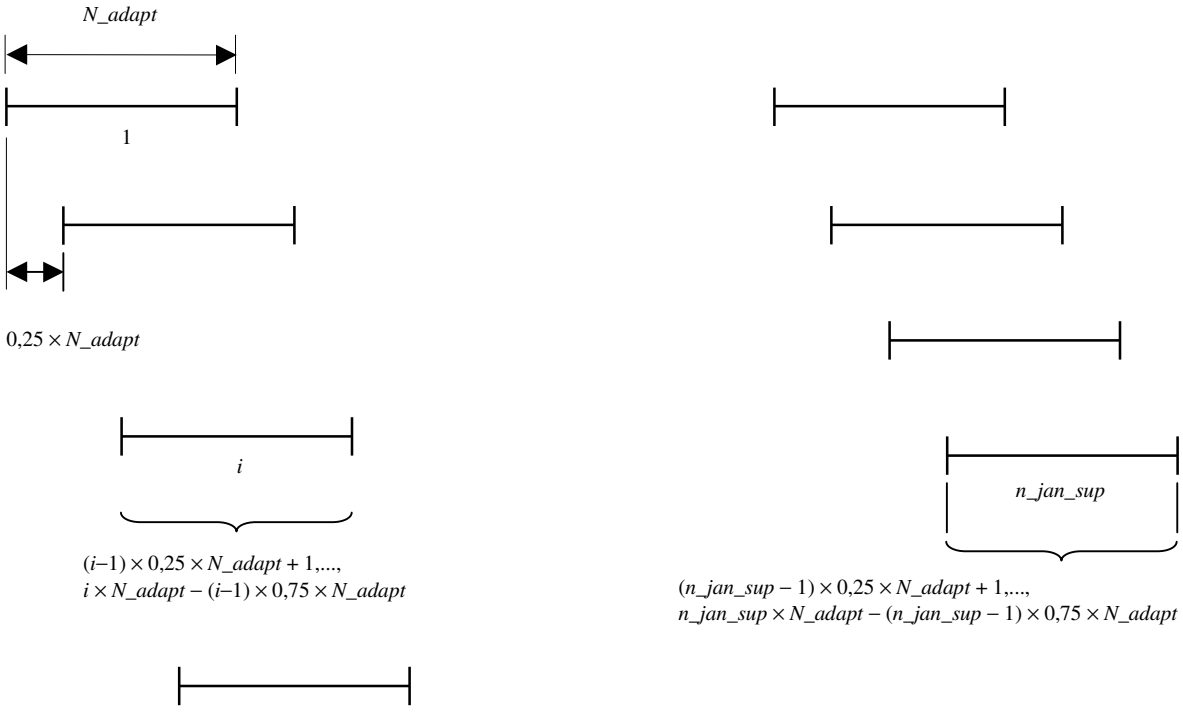
O número *N* é utilizado como um artifício para o cálculo do tamanho da janela adaptativa, *N_adapt*, dado por:

$$N\_adapt = n\_rec / N$$

onde *n_rec* é o número de amostras no sinal de voz recortado.

A figura seguinte ilustra esta abordagem.


N_adapt
1
2
k
N
1,...,N_adapt
N_adapt + 1,...,2 × N_adapt
(k–1) × N_adapt + 1,...,
k × N_adapt
(N–1) × N_adapt + 1,...,
N × N_adapt


N_adapt
1
0,25 × N_adapt
i
(i–1) × 0,25 × N_adapt + 1,...,
i × N_adapt – (i–1) × 0,75 × N_adapt
n_jan_sup
(n_jan_sup – 1) × 0,25 × N_adapt + 1,...,
n_jan_sup × N_adapt – (n_jan_sup – 1) × 0,75 × N_adapt

B.2 Pseudo-algoritmo para Extração de Parâmetros

Calcular o tamanho das janelas adaptativas *N_adapt*, a partir do nº fixado de janelas *n_jan_sup* (120), e do nº de amostras *n_rec* do sinal recortado.

Segmentar o sinal recortado, usando janelas de Hamming, com superposição de 75 %, conforme descrito anteriormente.

Efetuar "*zero-padding*" em cada segmento, até 512 pontos.

Calcular a Transformada de Fourier de cada segmento, através da FFT de 512 pontos.

Calcular os coeficientes cepstrais de cada segmento, a partir da Transformada de Fourier.

Extrair os 12 (doze) primeiros coeficientes cepstrais de cada segmento (excetuando o primeiro), através de liftragem com um liftro senoidal.

Apêndice C – ANÁLISE QUANTITATIVA

C.1 Discriminante Linear de Fischer

Sinais são descritos, de forme genérica, em termos de um conjunto de variáveis independentes $\mathbf{x} = \{x_1, x_2, ..., x_i, ..., x_N\}$. Em sistemas de reconhecimento de padrões, é importante conhecer quais dessas variáveis independentes possuem maior variabilidade entre classes diferentes (variabilidade inter-classes), e/ou menor variabilidade dentro de cada classe (variabilidade intra-classes), de modo a facilitar a classificação.

Neste trabalho, as variáveis independentes são consideradas como os coeficientes cepstrais. Esta análise tem como objetivo verificar, portanto, quais coeficientes cepstrais oferecem maior capacidade de discriminação, através do cálculo do Discriminante Linear de Fischer [11], [8], definido pela Eq. (7).

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \tag{7}$$

onde:

- *w*: matriz de transformação das amostras,
- S_B: variabilidade inter-classes,
- S_W: variabilidade intra-classes.

Ou seja: quanto mais os centróides das classes estiverem distanciados, ou quanto mais as amostras de cada classe estiverem agrupadas em torno de seus respectivos centróides, tanto maior será o Discriminante Linear de Fischer, indicando que as classes são mais separáveis.

Como o objetivo neste trabalho é a análise relativa entre os coeficientes cepstrais, a matriz *w* é suposta como uma matriz identidade, ficando o Discriminante Linear de Fischer reduzido à Eq. (8).

$$J = \frac{S_B}{S_W} \tag{8}$$

As variabilidades inter-classe e intra-classe são dadas, respectivamente, pela Eq. (9) e pelas Eq. (10a) e (10b).

$$S_B = \sum_{i=1}^{n-1} \sum_{j=2}^{n} (m_i - m_j)(m_i - m_j)^T \tag{9}$$

$$S_W = \sum_i \tilde{S}_i^2 \tag{10a}$$

sendo:

$$\tilde{S}_i^2 = \sum_{x \in i} (x - m_i)(x - m_i)^T \qquad (10b)$$

onde:

- m_i: centróide de uma classe *i*,
- $\tilde{S}_i^2$: avaliada para todas as amostras *x* pertencentes à classe *i*.

C.2 Discriminante Linear de Fischer – detalhamento

Como o objetivo desta análise é avaliar o Discriminante Linear de Fischer em função somente dos coeficientes cepstrais, optou-se por calcular as médias (centróides) de cada classe (comando), através da média de cada coeficiente cepstral ao longo de todos os segmentos dos sinais, conforme descrito a seguir.

$$m\_cepst_c(j) = \frac{\sum_{k=1}^{n\_am} \sum_{i=1}^{n\_jan} cepst_{ikc}(j)}{n\_am \cdot n\_jan}$$

onde:

- $m\_cepst_c(j)$: centróide da classe *c*, para o *j*-ésimo coeficiente cepstral,
- *n_jan*: nº de segmentos por amostra (locução),
- *n_am*: nº de amostras por classe,
- $cepst_{ikc}(j)$: *j*-ésimo coeficiente cepstral, do *i*-ésimo segmento, da *k*-ésima amostra, pertencente à classe *c*.

De forma análoga, a variabilidade intra-classes S_W foi calculada também através de médias dos coeficientes cepstrais ao longo de todos os segmentos dos sinais.

$$S_W(j) = \sum_c \frac{\sum_{k=1}^{n\_am} \sum_{i=1}^{n\_jan} [cepst_{ikc}(j) - m\_cepst_c(j)]^2}{n\_am \cdot n\_jan}$$

onde é efetuado um somatório para todas as classes *c*, de modo a computar a variabilidade intra-classes acumulada em todas as classes, conforme as Eq. (10a) e (10b).

C.3 Pseudo-algoritmo para Cálculo do Discriminante Linear de Fischer

Ler matrizes dos coeficientes cepstrais ao longo dos segmentos, em arquivos ".dat", para cada amostra de cada classe.

Calcular as médias de cada classe, para cada coeficiente cepstral.

Calcular as distâncias de cada amostra ao centróide da respectiva classe, para cada coeficiente cepstral.

Calcular as variabilidades inter-classes e intra-classes, e o Discriminante de Fischer, para cada coeficiente cepstral, salvando o resultado em dois arquivos: um ".dat", e outro matriz a ser lida como texto ou planilha eletrônica.

C.4 Resultados da Análise Quantitativa[4]

A Tabela a seguir mostra os resultados obtidos. Conforme pode ser observado, para este conjunto de locuções, os coeficientes índices 4, 5 e 6 oferecem melhor discriminação das classes. Para outros conjuntos de locuções, este resultado pode variar.

Não há um limite mínimo do valor do Discriminante Linear de Fischer, para a seleção dos coeficientes cepstrais a serem utilizados, a escolha do limite é subjetiva. Podem ser utilizados também todos os coeficientes cepstrais.

Discriminante Linear de Fischer para cada coeficiente cepstral

Coef. cepstral	*J*
1	0,2923454
2	0,2258417
3	0,2708826
4	0,8612820
5	0,9500568
6	1,0977337
7	0,4341068
8	0,3948290
9	0,3847671
10	0,1482143
11	0,4286628
12	0,3042747

[4] Estes resultados foram obtidos para um conjunto de locuções de um teste preliminar.

Apêndice D – ENTRADAS PARA O CLASSIFICADOR NEURAL

D.1 Geração do Arquivo

Os coeficientes cepstrais de todas as amostras, de todas as classes, são dispostos em uma matriz, salva em arquivo, a ser utilizada como entrada para o classificador neural.

Dois testes foram realizados: um considerando todos os coeficientes cepstrais, e outro considerando apenas os mais relevantes (para os testes realizados, 4, 5 e 6, conforme mostrado na Tabela do item C.4).

Para cada um dos dois testes, cada linha da matriz corresponde a uma amostra; a linha é montada da seguinte forma:

- Nas primeiras colunas da linha, a partir da esquerda, são dispostos seqüencialmente os coeficientes cepstrais de cada segmento, perfazendo um total de colunas que corresponde a: nº de coeficientes cepstrais utilizados × nº de segmentos;
- As últimas quatro colunas correspondem nº de saídas da rede neural, ou seja, ao nº de valores desejados, ou classes (para os testes realizados, quatro classes); a saída correspondente à classe correta vale "1", enquanto as demais valem "0".

No primeiro teste, a tabela de entrada tem a configuração mostrada a seguir:

$$\begin{bmatrix} [1,...,12] & \cdots \\ \vdots & \ddots \end{bmatrix}$$

onde:

- A matriz possui o nº de linhas correspondendo a: 4 classes × nº de amostras de cada classe;
- 1444 colunas, correspondendo a: 12 coeficientes cepstrais × 120 segmentos + 4 classes.

No segundo, a tabela de entrada tem a configuração:

$$\begin{bmatrix} [4,5,6] & \cdots \\ \vdots & \ddots \end{bmatrix}$$

onde:

- A matriz possui o nº de linhas correspondendo a: 4 classes × nº de amostras;

- 364 colunas, correspondendo a: 3 coeficientes cepstrais × 120 segmentos + 4 classes.

D.2 Pseudo-algoritmo para Geração da Entrada

A partir da matriz dos coeficientes cepstrais ao longo dos segmentos, para a amostra corrente:

{
Montar a primeira parte da linha correspondente à amostra, dispondo seqüencialmente os coeficientes cepstrais de cada segmento, conforme explicado anteriormente (até completar 1440 colunas, por exemplo),

Dispor as quatro últimas colunas, conforme a classe correta.
}

Verificar se existe o arquivo "entrada", onde são salvos os dados:

{
Caso exista, acrescenta a linha montada abaixo das existentes,

Caso contrário, cria o arquivo.
}

Apêndice E – PROGRAMAS DESENVOLVIDOS

E.1 Programas para Recorte, Extração e Geração da Entrada da Rede Neural

Nome do arquivo: "PreProcessa_mod.sce"

Observações:

- Deve ser verificado o diretório na primeira linha de código;
- O nome do arquivo a ser processado deve ser digitado na segunda linha;
- As instruções para plotagem dos resultados estão desativadas (em comentário), devido à demora na execução destas; para plotá-los separadamente, os comandos podem ser copiados e colados no ambiente do Scilab;
- A geração do arquivo de entrada para a rede neural está implementada considerando todos os coeficientes cepstrais, a seleção de alguns coeficientes através do Discriminante Linear de Fischer necessita ser implementada de forma mais genérica.

```
//Programas para Processamento de Sinais de Voz no Scilab
//Autor: Carlos Alexandre Fructuoso Jorge
//Serviço/Divisão: SEESC/DICH
//2006

//Programas para pré-processamento de sinais de voz, visando:
//Segmentação (janelamento) para recorte; e
//Extração de parâmetros (coeficientes cepstrais) para classificação.
//Extrai os parâmetros para cada locução, salvando em arquivo.
//Modificado para criar o arquivo de texto de entrada para a rede neural, a cada locução pré-processada.

//---------------------------------RECORTE--------------------------------

//Abordagem adotada:
//Segmentação com janelas retangulares de 20 ms, sem superposição;
//Eliminação de espaços em branco, conforme:
//Pinto, R. G.; Pinto, H. L.; Calôba, L. P.; "Using Neural Networks for Automatic Speaker Recognition: A Practical Approach", 38 (IEEE) MWSCAS, 1995.

//Leitura do tamanho (nº de amostras) do arquivo ".wav":
diretorio='c:\labrv\usuarios\CarlosAlexandre\voz\jan_hm_sup\';
arquivo='abaixo1'; //Obs.: o arquivo a ser processado deve ser digitado pelo usuário.
chdir(diretorio);
n_am=wavread(arquivo,'size');
```

```
//Leitura das amostras, incluindo a freqüência de amostragem e o número de bits para cada amostra:
[y,Fs,bits]=wavread(arquivo);

//Normalização da amplitude entre 0,5 e -0,5:
ampl_max=max(abs(y));
y=y/(2*ampl_max);

//Dada a duração dos segmentos (20 ms), calcula o tamanho "N" do segmento, truncando o sinal lido para "m" segmentos:
N=round(Fs*20/1000);
m=floor(n_am(2)/N);

//Truncagem do vetor original do sinal de voz y[n], para s=[1:n_max]:
n_max=m*N;
s(1:n_max)=y(1:n_max);

//Obs.: Plotagem desativada, devido à demora.
//Plota o sinal "s" x "n":
//subplot(2,1,1); plot(1:n_max,s); xtitle('Sinal de voz - '+arquivo); xgrid()

//Cálculo da energia de tempo curto para cada segmento "i", de tamanho "N": E(i)=sum(s(i).^2):
E=[1:m];
for i=1:m
  E(i)=sum(s((i-1)*N+1:i*N).^2);
end;

//Obs.: Plotagem desativada, devido à demora.
//Plota a energia "E" x "m":
//subplot(2,1,2); plot(1:m,E,'o-'); xtitle('Energia - Janelas retangulares de 20 ms'); xgrid()

//Eliminação de espaços em branco/ruído, eliminando segmentos com energia < 1% da máxima:
Emax=max(E);
n_seg=0; //Número de segmentos válidos
for i=1:m
  if E(i)>=0.01*Emax then
    n_seg=n_seg+1;
    //Recorte do sinal de voz:
    E_rec(n_seg)=E(i); //E_rec: Energia dos segmentos válidos
    s_rec((n_seg-1)*N+1:n_seg*N)=s((i-1)*N+1:i*N); //s_rec: sinal de voz recortado
  end;
end;
n_rec=n_seg*N; //Número de amostras válidas, recortadas do sinal original

//Reescalando para [-1,1] V, e gravando em arquivo .wav:
a_max=max(abs(s_rec));
s_esc=2*s_rec;
wavwrite(s_esc,Fs,arquivo+'_rec.wav');

//Obs.: Plotagem desativada, devido à demora.
//Plota a energia dos segmentos válidos e o sinal recortado:
//scf();
//subplot(2,1,1); plot(1:n_seg,E_rec,'o-'); xtitle('Segmentos válidos - '+arquivo); xgrid()
```

```
//subplot(2,1,2); plot(1:size(s_rec,'*'),s_rec); xtitle('Sinal de voz recortado'); xgrid()

//-------------------------EXTRAÇÃO DE PARÂMETROS-----------------------

//Abordagem adotada:
//Calcula os coeficientes cepstrais, a partir da FFT, para cada segmento de voz.
//Segmentação do sinal usando janelas de Hamming, com superposição de 75%;
//Mantém fixo o número de janelas, pelo método: segmentação adaptativa da voz, para
processamento (FFT e cepstrum), cf.:
//Diniz, S. S.; Uso de Técnicas Neurais para o Reconhecimento de Comandos à Voz, Tese de
Mestrado, IME, 1997.
//Cálculo da FFT com 512 pontos (faz zero-padding).

n_jan_sup=120; //Número desejado para as janelas, com superposição.
N=round((n_jan_sup-1)/4+1); //Número de divisões correspondentes no trecho recortado de voz.
N_adapt=floor(n_rec/N); //Tamanho adaptativo da janela.

//Cálculo da FFT de 512 pontos, para cada segmento:
L=512;
for i=1:n_jan_sup
  //Zero-padding: extensão dos segmentos para 512 pontos:
  seg(1:L)=zeros(1:L);
  seg(1:N_adapt)=s_rec((i-1)*0.25*N_adapt+1:i*N_adapt-(i-1)*0.75*N_adapt);
  //Multiplicação pela janela de Hamming.
  seg(1:N_adapt)=seg(1:N_adapt).*window('hm',N_adapt);
  X=fft(seg,-1); //Último parâmetro igual a -1 denota Transformada de Fourier.
  f=Fs*(0:(L-1)/2)/L; //Plota até metade da freqüência (equivalente ao intervalo de 0 a pi).
  w=2*%pi*f/Fs;
  X_log=20*log10(abs(X));
  X_mat(i,1:size(w,'*'))=X_log(1:size(w,'*')); //Matriz com a FFT de todos os segmentos.

  //Cálculo dos coeficientes cepstrais (cepstrum real):
  C=fft(log2(abs(X)),1); //Último parâmetro igual a 1 denota Transformada Inversa de Fourier.
  //Provavelmente por problema de cálculo no Scilab (vide manual), aparecem pequenos resíduos
imaginários (da ordem de 10E-17, aproximadamente). Por isso considera-se apenas a parte real.
  C=real(C);  //Parte real.
//  C_magn=abs(C);   //Magnitude.

  //Liftragem dos coeficientes cepstrais (extração de 12 coeficientes):
  L_lift=12;
  k=[1:L_lift];
  l=1+(L_lift/2)*sin(%pi*k/L_lift); //Liftro senoidal.
  //Liftragem no domínio das quefrências, desconsiderando o primeiro coeficiente.
  lift_C(i,:)=C(2:L_lift+1).*l(1:L_lift);
//  lift_C_magn(i,:)=C_magn(2:L_lift+1).*l(1:L_lift);
end;

//Salva a FFT (ao longo de todos os segmentos) em arquivo.
save(arquivo+'_spect.dat',X_mat);
//Salva coeficientes cepstrais (ao longo de todos os segmentos) em arquivo.
save(arquivo+'_cepst.dat',lift_C);
//save(arquivo+'_cepst_magn.dat',lift_C_magn);
```

```
//Obs.: Plotagem desativada.
//x=[1:n_jan_sup];
//plot3d(x,w,X_mat);
//a=get('current_axes'); a.cube_scaling='on'; a.axes_reverse(1)='on'; xgrid;
//xtitle('Magnitude da FFT - '+arquivo,'Segmentos','Freqüência','Espectro');

//Obs.: Plotagem desativada.
//scf();
//plot3d(x,k,lift_C);
//a=get('current_axes'); a.cube_scaling='on'; a.axes_reverse(1)='on'; xgrid;
//xtitle('Cepstra liftrados - '+arquivo,'Segmentos','Quefrências','Cepstrum');

//----------CRIAÇÃO DO ARQUIVO DE ENTRADA PARA A REDE NEURAL----------

n_classes=4; //nº de classes (quatro comandos)
//n_ptos=15; //nº de locuções de cada comando (quatro locuções)

//Lê, para a locução corrente, a variável "lift_C", que contém os coeficientes cepstrais para todos os
segmentos, criando  o arquivo de texto de entrada para a rede neural:
//A matriz "lift_C" é organizada na forma: (janela,cepstrum);
//A matriz (vetor linha) "entr" é organizada na forma: (1, [1,...,12] [1,...,12] ...), ou seja: os
coeficientes cepstrais de cada segmento são justapostos em seqüência, resultando em
12*120=1440 colunas;
//adicionalmente, há mais quatro colunas no final da matriz "entr", correspondentes aos neurônios
da camada de saída, ou seja, aos quatro comandos, resultando em um total de 1444 colunas.
for i=1:n_jan_sup
  for j=1:L_lift
    entr(1,(i-1)*L_lift+j)=lift_C(i,j);
    //Valores desejados (saídas da rede neural) correspondentes à classe "abaixo")
    if grep(arquivo,'abaixo')==1 then
      for coluna=1:4
        if coluna==1 then //Acrescenta "1" na primeira coluna de saída, e "0" nas demais.
          entr(1,n_jan_sup*L_lift+coluna)=1;
        else
          entr(1,n_jan_sup*L_lift+coluna)=0;
        end;
      end;
    //Valores desejados correspondentes à classe "acima")
    elseif grep(arquivo,'acima')==1 then
      for coluna=1:4
        if coluna==2 then
          entr(1,n_jan_sup*L_lift+coluna)=1;
        else
          entr(1,n_jan_sup*L_lift+coluna)=0;
        end;
      end;
    //Valores desejados correspondentes à classe "direita")
    elseif grep(arquivo,'direita')==1 then
      for coluna=1:4
        if coluna==3 then
          entr(1,n_jan_sup*L_lift+coluna)=1;
        else
          entr(1,n_jan_sup*L_lift+coluna)=0;
```

```
      end;
    end;
  //Valores desejados correspondentes à classe "esquerda")
  elseif grep(arquivo,'esquerda')==1 then
    for coluna=1:4
      if coluna==4 then
        entr(1,n_jan_sup*L_lift+coluna)=1;
      else
        entr(1,n_jan_sup*L_lift+coluna)=0;
      end;
    end;
  else
  end;
 end;
end;

//Verifica se o arquivo "entrada" já existe:
[x,ierr]=fileinfo('entrada');
//Caso exista, acrescenta o novo resultado na linha seguinte da matriz "entrada";
if ierr==0 then
  entrada=fscanfMat('entrada');
  entrada=[entrada; entr];
//Caso contrário, cria a matriz "entrada".
else
  entrada=entr;
end;
save('entrada.dat',entrada);
fprintfMat('entrada',entrada,'%10.7f');
```

E.2 Programa para Cálculo do Discriminante Linear de Fischer

Nome do arquivo: "Classes_cepst_mod.sce"

Observação:

- Programa implementado para o caso particular de 15 amostras de cada classe; necessita ainda ser otimizado para casos mais genéricos.

```
//Programas para Processamento de Sinais de Voz no Scilab
//Autor: Carlos Alexandre Fructuoso Jorge
//Serviço/Divisão: SEESC/DICH
//2006

//Cálculo do discriminante linear de Fischer.
//Modificado para tratamento dos nomes e números das locuções.

diretorio='c:\labrv\usuarios\CarlosAlexandre\voz\jan_hm_sup\';
chdir(diretorio);
```

```
njan_sup=120; //nºde segmentos (precisa fazer a média de cada coeficiente cepstral ao longo de todos os segmentos)
//n_classes=4; //nº de classes (quatro comandos)
n_ptos=15; //nº de locuções de cada comando (quinze locuções)
k=[1:12]; //nº de coeficientes cepstrais (equivalente a coordenadas)

//Leitura das matrizes:
//Abaixo: (n_classes=1)
load('abaixo1_cepst.dat','lift_C'); //n_ptos=1
abaixo_c1=lift_C;
load('abaixo2_cepst.dat','lift_C'); //n_ptos=2
abaixo_c2=lift_C;
load('abaixo3_cepst.dat','lift_C'); //n_ptos=3
abaixo_c3=lift_C;
load('abaixo4_cepst.dat','lift_C'); //n_ptos=4
abaixo_c4=lift_C;
load('abaixo5_cepst.dat','lift_C'); //n_ptos=5
abaixo_c5=lift_C;
load('abaixo6_cepst.dat','lift_C'); //n_ptos=6
abaixo_c6=lift_C;
load('abaixo7_cepst.dat','lift_C'); //n_ptos=7
abaixo_c7=lift_C;
load('abaixo8_cepst.dat','lift_C'); //n_ptos=8
abaixo_c8=lift_C;
load('abaixo9_cepst.dat','lift_C'); //n_ptos=9
abaixo_c9=lift_C;
load('abaixo10_cepst.dat','lift_C'); //n_ptos=10
abaixo_c10=lift_C;
load('abaixo11_cepst.dat','lift_C'); //n_ptos=11
abaixo_c11=lift_C;
load('abaixo12_cepst.dat','lift_C'); //n_ptos=12
abaixo_c12=lift_C;
load('abaixo13_cepst.dat','lift_C'); //n_ptos=13
abaixo_c13=lift_C;
load('abaixo14_cepst.dat','lift_C'); //n_ptos=14
abaixo_c14=lift_C;
load('abaixo15_cepst.dat','lift_C'); //n_ptos=15
abaixo_c15=lift_C;

//Acima: (n_classes=2)
load('acima1_cepst.dat','lift_C');
acima_c1=lift_C;
load('acima2_cepst.dat','lift_C');
acima_c2=lift_C;
load('acima3_cepst.dat','lift_C');
acima_c3=lift_C;
load('acima4_cepst.dat','lift_C');
acima_c4=lift_C;
load('acima5_cepst.dat','lift_C');
acima_c5=lift_C;
load('acima6_cepst.dat','lift_C');
acima_c6=lift_C;
```

```
load('acima7_cepst.dat','lift_C');
acima_c7=lift_C;
load('acima8_cepst.dat','lift_C');
acima_c8=lift_C;
load('acima9_cepst.dat','lift_C');
acima_c9=lift_C;
load('acima10_cepst.dat','lift_C');
acima_c10=lift_C;
load('acima11_cepst.dat','lift_C');
acima_c11=lift_C;
load('acima12_cepst.dat','lift_C');
acima_c12=lift_C;
load('acima13_cepst.dat','lift_C');
acima_c13=lift_C;
load('acima14_cepst.dat','lift_C');
acima_c14=lift_C;
load('acima15_cepst.dat','lift_C');
acima_c15=lift_C;

//Direita: (n_classes=3)
load('direita1_cepst.dat','lift_C');
direita_c1=lift_C;
load('direita2_cepst.dat','lift_C');
direita_c2=lift_C;
load('direita3_cepst.dat','lift_C');
direita_c3=lift_C;
load('direita4_cepst.dat','lift_C');
direita_c4=lift_C;
load('direita5_cepst.dat','lift_C');
direita_c5=lift_C;
load('direita6_cepst.dat','lift_C');
direita_c6=lift_C;
load('direita7_cepst.dat','lift_C');
direita_c7=lift_C;
load('direita8_cepst.dat','lift_C');
direita_c8=lift_C;
load('direita9_cepst.dat','lift_C');
direita_c9=lift_C;
load('direita10_cepst.dat','lift_C');
direita_c10=lift_C;
load('direita11_cepst.dat','lift_C');
direita_c11=lift_C;
load('direita12_cepst.dat','lift_C');
direita_c12=lift_C;
load('direita13_cepst.dat','lift_C');
direita_c13=lift_C;
load('direita14_cepst.dat','lift_C');
direita_c14=lift_C;
load('direita15_cepst.dat','lift_C');
direita_c15=lift_C;

//Esquerda: (n_classes=4)
load('esquerda1_cepst.dat','lift_C');
```

```
esquerda_c1=lift_C;
load('esquerda2_cepst.dat','lift_C');
esquerda_c2=lift_C;
load('esquerda3_cepst.dat','lift_C');
esquerda_c3=lift_C;
load('esquerda4_cepst.dat','lift_C');
esquerda_c4=lift_C;
load('esquerda5_cepst.dat','lift_C');
esquerda_c5=lift_C;
load('esquerda6_cepst.dat','lift_C');
esquerda_c6=lift_C;
load('esquerda7_cepst.dat','lift_C');
esquerda_c7=lift_C;
load('esquerda8_cepst.dat','lift_C');
esquerda_c8=lift_C;
load('esquerda9_cepst.dat','lift_C');
esquerda_c9=lift_C;
load('esquerda10_cepst.dat','lift_C');
esquerda_c10=lift_C;
load('esquerda11_cepst.dat','lift_C');
esquerda_c11=lift_C;
load('esquerda12_cepst.dat','lift_C');
esquerda_c12=lift_C;
load('esquerda13_cepst.dat','lift_C');
esquerda_c13=lift_C;
load('esquerda14_cepst.dat','lift_C');
esquerda_c14=lift_C;
load('esquerda15_cepst.dat','lift_C');
esquerda_c15=lift_C;

//Cálculo das médias (centróides) para cada classe:
//Calculada a média de cada coordenada (cada coeficiente cepstral).

//Inicialização:
for j=k
  m_cepst_abaixo(j)=0;
  m_cepst_acima(j)=0;
  m_cepst_direita(j)=0;
  m_cepst_esquerda(j)=0;
end;
//Calcula a média, para cada classe (comando), de cada coeficiente cepstral (coordenada)
for j=k //Loop para cada coeficiente cepstral (12)
  for i=1:njan_sup //Loop para cada segmento (120), faz a média ao longo dos segmentos
    //Média do k-ésimo coeficiente cepstral, para todos os segmentos, para cada classe.
    //Abaixo:
m_cepst_abaixo(j)=m_cepst_abaixo(j)+abaixo_c1(i,j)+abaixo_c2(i,j)+abaixo_c3(i,j)+abaixo_c4(i,j)..
    +abaixo_c5(i,j)+abaixo_c6(i,j)+abaixo_c7(i,j)+abaixo_c8(i,j)..
    +abaixo_c9(i,j)+abaixo_c10(i,j)+abaixo_c11(i,j)+abaixo_c12(i,j)..
    +abaixo_c13(i,j)+abaixo_c14(i,j)+abaixo_c15(i,j);

    //Acima:
m_cepst_acima(j)=m_cepst_acima(j)+acima_c1(i,j)+acima_c2(i,j)+acima_c3(i,j)+acima_c4(i,j)..
    +acima_c5(i,j)+acima_c6(i,j)+acima_c7(i,j)+acima_c8(i,j)..
```

```
    +acima_c9(i,j)+acima_c10(i,j)+acima_c11(i,j)+acima_c12(i,j)..
    +acima_c13(i,j)+acima_c14(i,j)+acima_c15(i,j);

    //Direita:
m_cepst_direita(j)=m_cepst_direita(j)+direita_c1(i,j)+direita_c2(i,j)+direita_c3(i,j)+direita_c4(i,j)..
    +direita_c5(i,j)+direita_c6(i,j)+direita_c7(i,j)+direita_c8(i,j)..
    +direita_c9(i,j)+direita_c10(i,j)+direita_c11(i,j)+direita_c12(i,j)..
    +direita_c13(i,j)+direita_c14(i,j)+direita_c15(i,j);

    //Esquerda:
m_cepst_esquerda(j)=m_cepst_esquerda(j)+esquerda_c1(i,j)+esquerda_c2(i
,j)+esquerda_c3(i,j)+esquerda_c4(i,j)..
   +esquerda_c5(i,j)+esquerda_c6(i,j)+esquerda_c7(i,j)+esquerda_c8(i,j)..
+esquerda_c9(i,j)+esquerda_c10(i,j)+esquerda_c11(i,j)+esquerda_c12(i,j)..
    +esquerda_c13(i,j)+esquerda_c14(i,j)+esquerda_c15(i,j);
  end;

//Calcula a média
  //Média abaixo:
  m_cepst_abaixo(j)=m_cepst_abaixo(j)/(n_ptos*njan_sup);
  //Média acima:
  m_cepst_acima(j)=m_cepst_acima(j)/(n_ptos*njan_sup);
  //Média direita:
  m_cepst_direita(j)=m_cepst_direita(j)/(n_ptos*njan_sup);
  //Média esquerda:
  m_cepst_esquerda(j)=m_cepst_esquerda(j)/(n_ptos*njan_sup);
end;

//Cálculo do discriminante linear de Fischer:

//Calcula distâncias entre os elementos de cada classe e seus respectivos centróides:
for j=k
Sw_ab1(j)=0; Sw_ab2(j)=0; Sw_ab3(j)=0; Sw_ab4(j)=0; Sw_ab5(j)=0; Sw_ab6(j)=0; Sw_ab7(j)=0;
Sw_ab8(j)=0; Sw_ab9(j)=0; Sw_ab10(j)=0; Sw_ab11(j)=0; Sw_ab12(j)=0; Sw_ab13(j)=0;
Sw_ab14(j)=0; Sw_ab15(j)=0;
Sw_ac1(j)=0; Sw_ac2(j)=0; Sw_ac3(j)=0; Sw_ac4(j)=0; Sw_ac5(j)=0;   Sw_ac6(j)=0; Sw_ac7(j)=0;
Sw_ac8(j)=0; Sw_ac9(j)=0; Sw_ac10(j)=0; Sw_ac11(j)=0; Sw_ac12(j)=0; Sw_ac13(j)=0;
Sw_ac14(j)=0; Sw_ac15(j)=0;
Sw_di1(j)=0; Sw_di2(j)=0; Sw_di3(j)=0; Sw_di4(j)=0; Sw_di5(j)=0; Sw_di6(j)=0; Sw_di7(j)=0;
Sw_di8(j)=0; Sw_di9(j)=0; Sw_di10(j)=0; Sw_di11(j)=0; Sw_di12(j)=0; Sw_di13(j)=0; Sw_di14(j)=0;
Sw_di15(j)=0;
Sw_es1(j)=0; Sw_es2(j)=0; Sw_es3(j)=0; Sw_es4(j)=0; Sw_es5(j)=0; Sw_es6(j)=0; Sw_es7(j)=0;
Sw_es8(j)=0; Sw_es9(j)=0; Sw_es10(j)=0; Sw_es11(j)=0; Sw_es12(j)=0; Sw_es13(j)=0;
Sw_es14(j)=0; Sw_es15(j)=0;
  Sw(j)=0;
  //Calcula a distância entre cada coordenada (coeficiente cepstral) e sua média para todos os
segmentos
  for i=1:njan_sup
    //abaixo:
    Sw_ab1(j)=Sw_ab1(j)+(abaixo_c1(i,j)-m_cepst_abaixo(j))^2; //distância entre cada coordenada,
para o ponto 1 (locução 1)
    Sw_ab2(j)=Sw_ab2(j)+(abaixo_c2(i,j)-m_cepst_abaixo(j))^2; //distância entre cada coordenada,
para o ponto 2 (locução 2)
```

```
   Sw_ab3(j)=Sw_ab3(j)+(abaixo_c3(i,j)-m_cepst_abaixo(j))^2; //distância entre cada coordenada,
para o ponto 3 (locução 3)
   Sw_ab4(j)=Sw_ab4(j)+(abaixo_c4(i,j)-m_cepst_abaixo(j))^2; //distância entre cada coordenada,
para o ponto 4 (locução 4)
   Sw_ab5(j)=Sw_ab5(j)+(abaixo_c5(i,j)-m_cepst_abaixo(j))^2; //distância entre cada coordenada,
para o ponto 5 (locução 5)
   Sw_ab6(j)=Sw_ab6(j)+(abaixo_c6(i,j)-m_cepst_abaixo(j))^2; //distância entre cada coordenada,
para o ponto 6 (locução 6)
   Sw_ab7(j)=Sw_ab7(j)+(abaixo_c7(i,j)-m_cepst_abaixo(j))^2; //distância entre cada coordenada,
para o ponto 7 (locução 7)
   Sw_ab8(j)=Sw_ab8(j)+(abaixo_c8(i,j)-m_cepst_abaixo(j))^2; //distância entre cada coordenada,
para o ponto 8 (locução 8)
   Sw_ab9(j)=Sw_ab9(j)+(abaixo_c9(i,j)-m_cepst_abaixo(j))^2; //distância entre cada coordenada,
para o ponto 9 (locução 9)
   Sw_ab10(j)=Sw_ab10(j)+(abaixo_c10(i,j)-m_cepst_abaixo(j))^2; //distância entre cada
coordenada, para o ponto 10 (locução 10)
   Sw_ab11(j)=Sw_ab11(j)+(abaixo_c11(i,j)-m_cepst_abaixo(j))^2; //distância entre cada
coordenada, para o ponto 11 (locução 11)
   Sw_ab12(j)=Sw_ab12(j)+(abaixo_c12(i,j)-m_cepst_abaixo(j))^2; //distância entre cada
coordenada, para o ponto 12 (locução 12)
   Sw_ab13(j)=Sw_ab13(j)+(abaixo_c13(i,j)-m_cepst_abaixo(j))^2; //distância entre cada
coordenada, para o ponto 13 (locução 13)
   Sw_ab14(j)=Sw_ab14(j)+(abaixo_c14(i,j)-m_cepst_abaixo(j))^2; //distância entre cada
coordenada, para o ponto 14 (locução 14)
   Sw_ab15(j)=Sw_ab15(j)+(abaixo_c15(i,j)-m_cepst_abaixo(j))^2; //distância entre cada
coordenada, para o ponto 15 (locução 15)
   //acima:
   Sw_ac1(j)=Sw_ac1(j)+(acima_c1(i,j)-m_cepst_acima(j))^2;
   Sw_ac2(j)=Sw_ac2(j)+(acima_c2(i,j)-m_cepst_acima(j))^2;
   Sw_ac3(j)=Sw_ac3(j)+(acima_c3(i,j)-m_cepst_acima(j))^2;
   Sw_ac4(j)=Sw_ac4(j)+(acima_c4(i,j)-m_cepst_acima(j))^2;
   Sw_ac5(j)=Sw_ac5(j)+(acima_c5(i,j)-m_cepst_acima(j))^2;
   Sw_ac6(j)=Sw_ac6(j)+(acima_c6(i,j)-m_cepst_acima(j))^2;
   Sw_ac7(j)=Sw_ac7(j)+(acima_c7(i,j)-m_cepst_acima(j))^2;
   Sw_ac8(j)=Sw_ac8(j)+(acima_c8(i,j)-m_cepst_acima(j))^2;
   Sw_ac9(j)=Sw_ac9(j)+(acima_c9(i,j)-m_cepst_acima(j))^2;
   Sw_ac10(j)=Sw_ac10(j)+(acima_c10(i,j)-m_cepst_acima(j))^2;
   Sw_ac11(j)=Sw_ac11(j)+(acima_c11(i,j)-m_cepst_acima(j))^2;
   Sw_ac12(j)=Sw_ac12(j)+(acima_c12(i,j)-m_cepst_acima(j))^2;
   Sw_ac13(j)=Sw_ac13(j)+(acima_c13(i,j)-m_cepst_acima(j))^2;
   Sw_ac14(j)=Sw_ac14(j)+(acima_c14(i,j)-m_cepst_acima(j))^2;
   Sw_ac15(j)=Sw_ac15(j)+(acima_c15(i,j)-m_cepst_acima(j))^2;
   //direita:
   Sw_di1(j)=Sw_di1(j)+(direita_c1(i,j)-m_cepst_direita(j))^2;
   Sw_di2(j)=Sw_di2(j)+(direita_c2(i,j)-m_cepst_direita(j))^2;
   Sw_di3(j)=Sw_di3(j)+(direita_c3(i,j)-m_cepst_direita(j))^2;
   Sw_di4(j)=Sw_di4(j)+(direita_c4(i,j)-m_cepst_direita(j))^2;
   Sw_di5(j)=Sw_di5(j)+(direita_c5(i,j)-m_cepst_direita(j))^2;
   Sw_di6(j)=Sw_di6(j)+(direita_c6(i,j)-m_cepst_direita(j))^2;
   Sw_di7(j)=Sw_di7(j)+(direita_c7(i,j)-m_cepst_direita(j))^2;
   Sw_di8(j)=Sw_di8(j)+(direita_c8(i,j)-m_cepst_direita(j))^2;
   Sw_di9(j)=Sw_di9(j)+(direita_c9(i,j)-m_cepst_direita(j))^2;
   Sw_di10(j)=Sw_di10(j)+(direita_c10(i,j)-m_cepst_direita(j))^2;
```

```
   Sw_di11(j)=Sw_di11(j)+(direita_c11(i,j)-m_cepst_direita(j))^2;
   Sw_di12(j)=Sw_di12(j)+(direita_c12(i,j)-m_cepst_direita(j))^2;
   Sw_di13(j)=Sw_di13(j)+(direita_c13(i,j)-m_cepst_direita(j))^2;
   Sw_di14(j)=Sw_di14(j)+(direita_c14(i,j)-m_cepst_direita(j))^2;
   Sw_di15(j)=Sw_di15(j)+(direita_c15(i,j)-m_cepst_direita(j))^2;
   //esquerda:
   Sw_es1(j)=Sw_es1(j)+(esquerda_c1(i,j)-m_cepst_esquerda(j))^2;
   Sw_es2(j)=Sw_es2(j)+(esquerda_c2(i,j)-m_cepst_esquerda(j))^2;
   Sw_es3(j)=Sw_es3(j)+(esquerda_c3(i,j)-m_cepst_esquerda(j))^2;
   Sw_es4(j)=Sw_es4(j)+(esquerda_c4(i,j)-m_cepst_esquerda(j))^2;
   Sw_es5(j)=Sw_es5(j)+(esquerda_c5(i,j)-m_cepst_esquerda(j))^2;
   Sw_es6(j)=Sw_es6(j)+(esquerda_c6(i,j)-m_cepst_esquerda(j))^2;
   Sw_es7(j)=Sw_es7(j)+(esquerda_c7(i,j)-m_cepst_esquerda(j))^2;
   Sw_es8(j)=Sw_es8(j)+(esquerda_c8(i,j)-m_cepst_esquerda(j))^2;
   Sw_es9(j)=Sw_es9(j)+(esquerda_c9(i,j)-m_cepst_esquerda(j))^2;
   Sw_es10(j)=Sw_es10(j)+(esquerda_c10(i,j)-m_cepst_esquerda(j))^2;
   Sw_es11(j)=Sw_es11(j)+(esquerda_c11(i,j)-m_cepst_esquerda(j))^2;
   Sw_es12(j)=Sw_es12(j)+(esquerda_c12(i,j)-m_cepst_esquerda(j))^2;
   Sw_es13(j)=Sw_es13(j)+(esquerda_c13(i,j)-m_cepst_esquerda(j))^2;
   Sw_es14(j)=Sw_es14(j)+(esquerda_c14(i,j)-m_cepst_esquerda(j))^2;
   Sw_es15(j)=Sw_es15(j)+(esquerda_c15(i,j)-m_cepst_esquerda(j))^2;
  end;
  //abaixo:
Sw_abaixo(j)=Sw_ab1(j)+Sw_ab2(j)+Sw_ab3(j)+Sw_ab4(j)+Sw_ab5(j)+Sw_ab6(j)+Sw_ab7(j)+Sw_
ab8(j)..
+Sw_ab9(j)+Sw_ab10(j)+Sw_ab11(j)+Sw_ab12(j)+Sw_ab13(j)+Sw_ab14(j)+Sw_ab15(j);
  Sw_abaixo(j)=Sw_abaixo(j)/(n_ptos*njan_sup); //Calcula a distância média ao longo de todos os
segmentos
  //acima:
Sw_acima(j)=Sw_ac1(j)+Sw_ac2(j)+Sw_ac3(j)+Sw_ac4(j)+Sw_ac5(j)+Sw_ac6(j)+Sw_ac7(j)+Sw_a
c8(j)..
+Sw_ac9(j)+Sw_ac10(j)+Sw_ac11(j)+Sw_ac12(j)+Sw_ac13(j)+Sw_ac14(j)+Sw_ac15(j);
  Sw_acima(j)=Sw_acima(j)/(n_ptos*njan_sup);
  //direita:
Sw_direita(j)=Sw_di1(j)+Sw_di2(j)+Sw_di3(j)+Sw_di4(j)+Sw_di5(j)+Sw_di6(j)+Sw_di7(j)+Sw_di8(j)..
+Sw_di9(j)+Sw_di10(j)+Sw_di11(j)+Sw_di12(j)+Sw_di13(j)+Sw_di14(j)+Sw_di15(j);
  Sw_direita(j)=Sw_direita(j)/(n_ptos*njan_sup);
  //esquerda:
Sw_esquerda(j)=Sw_es1(j)+Sw_es2(j)+Sw_es3(j)+Sw_es4(j)+Sw_es5(j)+Sw_es6(j)+Sw_es7(j)+Sw
_es8(j)..
+Sw_es9(j)+Sw_es10(j)+Sw_es11(j)+Sw_es12(j)+Sw_es13(j)+Sw_es14(j)+Sw_es15(j);
  Sw_esquerda(j)=Sw_esquerda(j)/(n_ptos*njan_sup);
  //Soma:
  Sw(j)=Sw_abaixo(j)+Sw_acima(j)+Sw_direita(j)+Sw_esquerda(j);
end;

//Cálculo das distâncias entre os centróides das classes:
for j=k
Sb_m1m2(j)=0; Sb_m1m3(j)=0; Sb_m1m4(j)=0; Sb_m2m3(j)=0; Sb_m2m4(j)=0; Sb_m3m2(j)=0;
Sb_m3m4(j)=0;
  Sb(j)=0;
  Sb_m1m2(j)=Sb_m1m2(j)+(m_cepst_abaixo(j)-m_cepst_acima(j))^2;
  Sb_m1m3(j)=Sb_m1m3(j)+(m_cepst_abaixo(j)-m_cepst_direita(j))^2;
```

```
  Sb_m1m4(j)=Sb_m1m4(j)+(m_cepst_abaixo(j)-m_cepst_esquerda(j))^2;
  Sb_m2m3(j)=Sb_m1m2(j)+(m_cepst_acima(j)-m_cepst_direita(j))^2;
  Sb_m2m4(j)=Sb_m1m2(j)+(m_cepst_acima(j)-m_cepst_esquerda(j))^2;
  Sb_m3m2(j)=Sb_m1m2(j)+(m_cepst_direita(j)-m_cepst_acima(j))^2;
  Sb_m3m4(j)=Sb_m1m2(j)+(m_cepst_direita(j)-m_cepst_esquerda(j))^2;

Sb(j)=Sb_m1m2(j)+Sb_m1m3(j)+Sb_m1m4(j)+Sb_m2m3(j)+Sb_m2m4(j)+Sb_m3m2(j)+Sb_m3m4(j
);
end;

//Discriminante linear de Fischer:
J(:)=Sb(:)./Sw(:)
save('dFischer.dat',J);
fprintfMat('dFischer',J,'%10.7f');
```
