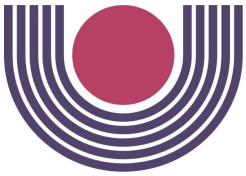


Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Reconhecimento de Fala Utilizando Modelo Oculto de Markov

Maycon de Queiroz Oliveira

CASCAVEL
2018

MAYCON DE QUEIROZ OLIVEIRA

RECONHECIMENTO DE FALA UTILIZANDO MODELO OCULTO DE MARKOV

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada no dia 28/06/2018 pela Comissão formada pelos professores:

Prof. M. Eng. Adriana Postal (Orientadora)
Colegiado de Ciência da Computação,
UNIOESTE - Campus de Cascavel

Prof. M. Eng. Josué Pereira de Castro
Colegiado de Ciência da Computação,
UNIOESTE - Campus de Cascavel

Prof. Dr. Gustavo Henrique Paetzold
Colegiado de Engenharia de Software,
UTFPR - Campus de Dois Vizinhos

CASCAVEL
2018

EPÍGRAFE

Suaviza contornos, quebra istmos estreitos e elimina proeminências delgadas.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

MFCC	*Mel-Frequency Cepstral Coefficients*
GMM	*Gaussian Mixture Models*

Lista de Símbolos

T	Comprimento da sequência de observações
N	Número de estados do modelo
M	Número de observações possíveis
Q	Conjunto finito de estados ocultos
V	Conjunto finito discreto das possíveis observações
A	Matriz de probabilidades de transição entre os estados ocultos
B	Matriz de probabilidades de observação
π	Distribuição de probabilidades iniciais
O	$O_0, O_1, O_2, ..., O_{T-1}$. Sequência de observações
X	Sequência de estados que representam a observação de O
r	Vetor de aproximação de Viterbi
γ	Vetor de probabilidades retroativas do cálculo de Viterbi
ϕ	Peso para funções gaussianas
η	Função de distribuição gaussiana
G	Número de curvas gaussianas no somatório

Sumário

Resumo

Este trabalho descreve um estudo do Modelo Oculto de Markov juntamente com *Mel-Frequency Cepstral Coefficients*, Modelo de Mistura Gaussiana e Algoritmo de Viterbi para estudos e testes de uma solução para o problema de reconhecimento de fala em escopos delimitados sobre os comandos “frente”, “trás”, “esquerda”, “direita” e “para” através do uso do software MATLAB®.

Palavras-chave: Reconhecimento de padrões, *Mel-Frequency Cepstral Coefficients*, Modelo de Mistura Gaussiana, Algoritmo de Viterbi, Comandos de voz.

Capítulo 1

Introdução

1.1 Reconhecimento de fala

O reconhecimento de fala tem sido meta para muitos cientistas da computação desde a década de 50 (DAVIS; BIDDULPH; BALASHEK, 1952) principalmente por poder criar um modelo de iteração homem-máquina mais amigável e fácil de ser manipulado. O aumento na capacidade de processamento dos computadores atuais acabou transferindo o problema para a programação de sistemas eficientes ao invés de poder computacional para processá-lo (SILVA, 2009).

Um sistema de reconhecimento de fala deve ser capaz de captar as ondas sonoras produzidas por uma pessoa com um instrumento de captura e converter as ondas em sinais digitais para futuro processamento e iteração com o usuário (MANICA, 2014) como ilustrado na Figura 1.1, adaptada de Braga (2006).

Dentre as áreas envolvidas com reconhecimento de fala, podem ser citados processamento de sinais, ciência da computação, reconhecimento de padrões, inteligência artificial, neurofisiologia, linguística e acústica. O sistema deve ser robusto para poder operar sob condições adversas como ruídos, baixo volume do locutor e até mesmo diferenças de sotaque que podem comprometer sistemas que exijam muita regularidade dos dados (SILVA, 2009). Certamente tal variabilidade de condições é um fator preocupante para o desenvolvimento desse tipo de sistema, entretanto não deve ser desconsiderada e sim implementada junto ao sistema.

O sinal de fala é captado por um transdutor (microfone), que converte os sinais mecânicos através do diafragma em sinais elétricos que correspondem a frequência do som, os sinais são então salvos em um arquivo com um conjunto de parâmetros espectrais e temporais para que

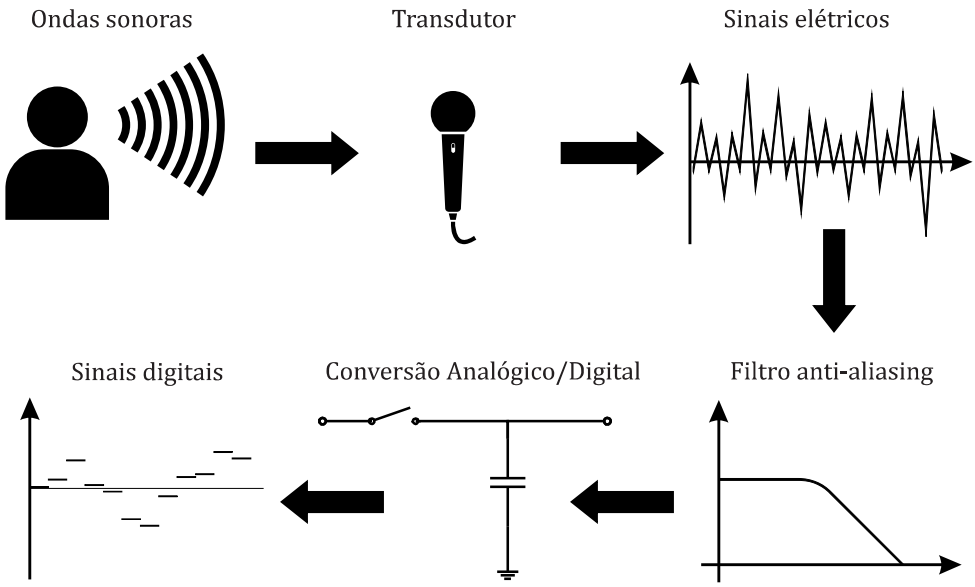


Figura 1.1: Processo de captura de sinais de som

seja possível ser analisado pelo computador; esse método é utilizado porque o simples tratamento de ondas sonoras não é muito interessante por abranger uma gama muito longa de ondas possíveis; assim como falamos a mesma palavra de várias formas diferentes, uma mudança de tom, ou variação de tempo de fala em milissegundos poderia atrapalhar senão inviabilizar o processamento de tais ondas (MANICA, 2014). Para termos o resultado de uma análise espectral, pode-se usar a *Mel-Frequency Cepstral Coefficients*, que extrai componentes essenciais ao ouvido humano e converte o áudio do domínio espacial discreto para o domínio da frequência, melhor descrita na seção 3.2.1.

O reconhecimento de fala é um problema importante e antigo na computação (DAVIS; BIDDULPH; BALASHEK, 1952), com inúmeras aplicações, desde aplicações de entretenimento que podem divertir o usuário até atividades que passam a ser possíveis devido a existência de tal recurso como a iteração com pessoas com alguma deficiência motora; uma área com avanços nos métodos de iteração humano-computador. O Modelo Oculto de Markov é um método conhecidamente eficiente para a elaboração de uma possível solução para tal problema, como pode ser observado em publicações como as de Juang e Rabiner (1991), Bandeira, Junior e Costa (2008), Nilsson e Ejnarsson (2002).

No Capítulo 2 veremos que o Modelo Oculto de Markov é uma máquina de estados com transições representadas através de matrizes. Por esse motivo, a plataforma de desenvolvimento

MATLAB® foi escolhida por ser um ambiente para operações matriciais (MOLER, 2004) e por que o algoritmo usado como base encontra-se nesta linguagem.

1.2 Objetivos

O trabalho consiste em utilizar a implementação de um módulo de reconhecimento de fala isolado, que utiliza o Modelo Oculto de Markov para distinguir entre as palavras "‘Frente"’, "‘Trás"’, "‘Esquerda"’, "‘Direita"’ e "‘Para"’, a última em sua forma paroxítona e analisar seu comportamento e acurácia.

1.3 Trabalhos correlatos

Diversos trabalhos abordam aplicações que efetuam algum reconhecimento de fala como é o caso do trabalho de Manica (2014), que desenvolve um sistema de reconhecimento de fala para robôs LEGO® MINDSTORMS NXT 2.0 através de uma rede neural. O trabalho de Tevah (2006), que implementa um reconhecedor de fala com amplo vocabulário, utilizando a ferramenta HTK (*Hidden Markov Model Toolkit*). E o trabalho de Bresolin (2003), que elabora um estudo do reconhecimento de fala para acionamento de equipamentos elétricos.

Vários trabalhos também citam aplicações envolvendo Modelos Ocultos de Markov, sejam estas para reconhecimento de fala ou não, como o trabalho de Karlof e Wagner (2003), que apresenta uma modelagem de Modelo Oculto de Markov para criptoanálise. O trabalho de Yoon (2009), apresenta uma série de aplicações dos Modelos Ocultos de Markov para análises genéticas. E o trabalho de Alam, MezbahUddin e Shoma (2015) que utiliza tais modelos para previsão de resultados das eleições baseado em fatores como partido, popularidade, tempo na política.

Há também diversos trabalhos que tratam especificamente de reconhecimento de fala utilizando o Modelo Oculto de Markov, como é o trabalho de Bandeira, Junior e Costa (2008) que levanta fundamentação teórica e Bissoli et al. (2013) que utiliza tal recurso para comandar uma cadeira de rodas elétrica.

1.4 Organização do trabalho

- O capítulo 2 aborda o funcionamento do Modelo Oculto de Markov.
- O capítulo 3 aborda os algoritmos e ferramentas utilizadas neste trabalho.
- O capítulo 4 aborda a implementação e os testes realizados.
- O capítulo 5 aborda considerações finais e trabalhos futuros propostos.

Capítulo 2

Modelo Oculto de Markov

2.1 Introdução

A introdução ao Modelo Oculto de Markov apresentada nesse capítulo foi baseada no conteúdo de Luger (2004) e Dimuro et al. (2002). O Modelo Oculto de Markov foi descrito pela primeira vez no final dos anos 60 e início dos anos 70 (BAUM, 1972) (BAUM; EAGON, 1967) (BAUM; PETRIE, 1966). Os formalismos do Modelo Oculto de Markov se apoiam na ideia de que cada evento pode estar probabilisticamente relacionado com eventos predecessores, o que é importante para programar um computador que aprenda fenômenos do mundo. O Modelo Oculto de Markov é uma generalização das Cadeias de Markov, onde eventos observáveis correspondiam aos estados da máquina de estados e não existiam estados ocultos (MARKOV, 1906). O Modelo Oculto de Markov possui estados conectados por transições que podem disparar com uma dada probabilidade, o processo sofre alterações ocultas mas provoca acontecimentos observáveis. Durante uma simulação, o sistema recebe estados observáveis (O_n) ligados a estados ocultos (X_n) por transições que disparam com uma dada probabilidade (B) e então a rede sofre adaptação para melhor representar a relação entre os estados ocultos e observáveis.

Os Modelos Ocultos de Markov são uma forma probabilística de relacionar fatos diretamente relacionados. O mais comum é o Modelo Oculto de Markov de tempo discreto, onde as observações são realizadas em um instante de tempo t, onde $0 \leqslant t \leqslant T$, com T sendo o tempo limite da observação (ESPINDOLA, 2009). Logo, pode-se pensar em um Modelo Oculto de Markov como uma Máquina de Moore (MENEZES, 1998), onde as transições são vazias e probabilísticas e em cada estado pode haver a emissão de símbolos (itens observáveis) com uma dada probabilidade. O Modelo Oculto de Markov não possui memória, dependendo apenas do

estado atual da máquina de estados para adotar as probabilidades para a sequência de um novo estado. Indicamos o estado da máquina no tempo t como σ_t, onde uma descrição do sistema requer a especificação do estado σ_t em relação ao estado predecessor direto. Sendo σ_t o estado atual da máquina de estados, então:

$$P(\sigma_t) = P(\sigma_t|\sigma_{t-1}) \tag{2.1}$$

Onde pela definição do Teorema de Bayes descrito em Morettin (2000), sabemos que o sistema está no estado σ_t dado que o estado σ_{t-1} ocorreu anteriormente.

A definição formal de um Modelo Oculto de Markov usada neste trabalho, adaptada de Stamp (2015), se dá por:

T = Comprimento da sequência de observações;

N = Número de estados do modelo;

M = Número de observações possíveis;

Q = Conjunto finito de estados ocultos;

V = Conjunto finito discreto das possíveis observações;

A = Matriz de probabilidades de transição entre os estados ocultos;

B = Matriz de probabilidades de observação;

π = Distribuição de probabilidades iniciais;

O = $O_0, O_1, O_2, ..., O_{T-1}$. Sequência de observações;

X = Sequência de estados que representam a observação de O.

Considerando as matrizes de transições A e B genéricas, onde a matriz $A = a_{ij}$ é NxN, com cada linha somando 1, onde

$$a_{ij} = P(\text{estado } q_j \text{ em } t+1 \mid \text{estado } q_i \text{ no tempo } t) \tag{2.2}$$

A matriz $B = b_j(k)$ é NxM, com cada linha somando 1, onde

$$b_j(k) = P(\text{observação de } k \text{ no tempo } t \mid \text{estado } q_j \text{ no tempo } t) \tag{2.3}$$

Um Modelo Oculto de Markov é definido por A, B e π, sendo denotada formalmente por $\lambda = (A, B, \pi)$. Considere uma sequência genérica de estados de comprimento quatro dada por X com suas observações correspondentes dada por O:

$$X = (x_0, x_1, x_2, x_3) \tag{2.4}$$

$$O = (O_0, O_1, O_2, O_3) \tag{2.5}$$

Uma cadeia de Markov genérica é apresentada na Figura 2.1. Os estados de Markov (que são ocultos acima da linha tracejada) são determinados pelo estado atual e a matriz de transições A. Nós apenas temos as observações O que estão diretamente relacionadas com os processos pela matriz B.

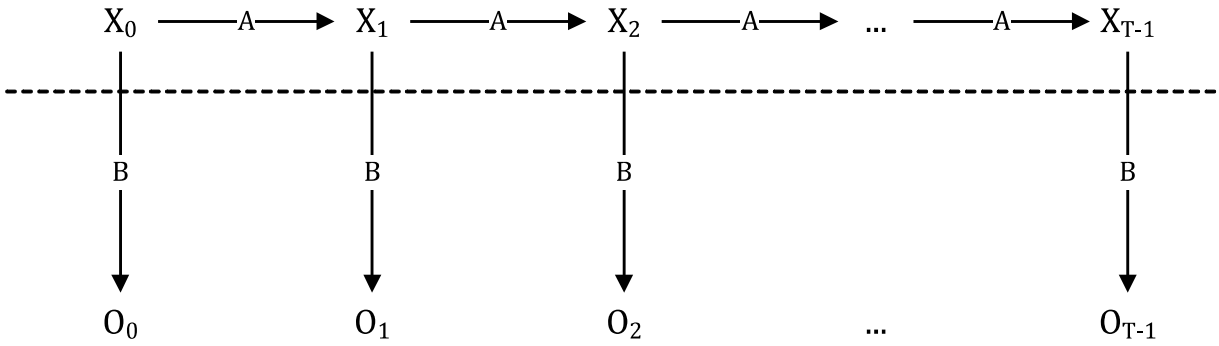


Figura 2.1: Apresentação de um processo de Markov genérico

Perceba que todas as probabilidades são independentes do tempo t. O valor π_{x0} é a probabilidade de que a máquina de estados no tempo t_0, tempo inicial estar no estado $x0$. Logo, $b_{x0}(O_0)$ é a probabilidade de se iniciar observando O_0 no estado $x0$ e $a_{x0,x1}$ é a probabilidade de transição do estado $x0$ para o estado $x1$. Logo, a probabilidade de ocorrência da sequência X é dada por

$$P(X, O) = \pi_{x0} b_{x0}(O_0) a_{x0,x1} b_{x1}(O_1) a_{x1,x2} b_{x2}(O_2) a_{x2,x3} b_{x3}(O_3) \tag{2.6}$$

2.1.1 Exemplo dos anéis de árvores

Suponha que queremos a temperatura de um determinado ambiente ao passar dos anos (exemplo adaptado de Stamp (2015)), onde não teremos um número exato, mas sim uma variável qualitativa da temperatura, sendo ela quente (representada pela letra Q) ou fria (representada pela letra F); entretanto, não temos observações diretas de moradores locais, apenas evidências atuais das temperaturas que acontecem durante os anos. Suponha que a probabilidade de um ano frio seguido de outro ano frio seja de 0.6 (60%) e de um ano quente seguido de outro ano quente seja de 0.7 (70%). Assumindo que essas probabilidades valham para um passado mais distante, essa informação pode ser representada pelo diagrama na Figura 2.2 e na Matriz 2.7.

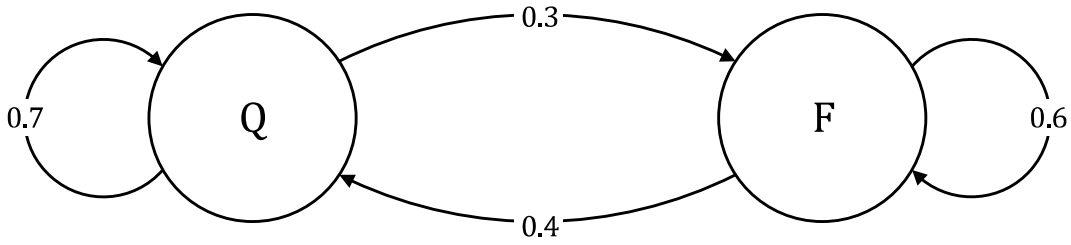


Figura 2.2: Diagrama de transições sobre as temperaturas dos anos

$$A = \begin{matrix} \\ \text{Quente} \\ \text{Frio} \end{matrix} \begin{matrix} \begin{matrix} \text{Quente} & \text{Frio} \end{matrix} \\ \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \end{matrix} \tag{2.7}$$

Sabemos entretanto, que o tamanho dos anéis nos troncos das árvores da região são excelentes indicadores da temperatura média anual. Por simplicidade, consideremos esse dado também uma variável qualitativa, variando entre pequeno (representado pela letra P), médio (representado pela letra M) e grande (representado pela letra G), como parte do exemplo, a temperatura anual pode ser dada pela matriz na tabela 2.8, onde a soma das linhas deve ser igual a 1.

$$B = \begin{matrix} \\ \text{Quente} \\ \text{Frio} \end{matrix} \begin{matrix} \begin{matrix} \text{Pequeno} & \text{Médio} & \text{Grande} \end{matrix} \\ \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{bmatrix} \end{matrix} \tag{2.8}$$

Temos, então, os dados observáveis (tamanho do anel) e as probabilidades para algo oculto (as temperaturas), que podem ser ambas expressadas pelo diagrama de estados na Figura 2.3.

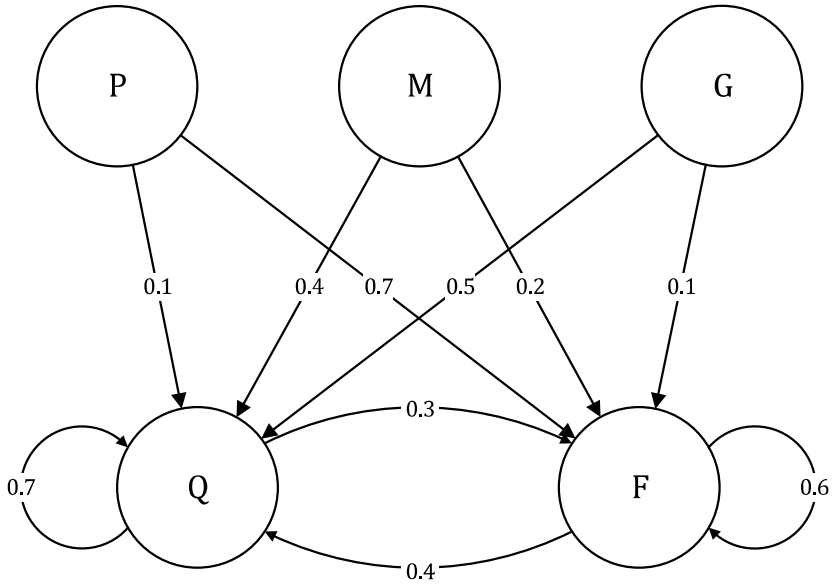


Figura 2.3: Diagrama de estados para representação da relação tamanho dos anéis/temperaturas

Consideremos então, uma amostra de uma série de anéis P, M, P, G; onde queremos determinar a mais provável sequência de temperaturas destes anos, considerando a probabilidade $\pi = [0.6, 0.4]$, temos um valor para cada possibilidade de acordo com a equação das probabilidades dos estados, como por exemplo a probabilidade de dois anos quentes seguidos de dois anos frios seguindo a equação apresentada em 2.6.

$$P(PMPG, QQFF) = 0.6 \cdot 0.1 \cdot 0.7 \cdot 0.4 \cdot 0.3 \cdot 0.7 \cdot 0.6 \cdot 0.1 = 0.000212 \tag{2.9}$$

Baseado nesses resultados, podemos construir a Tabela 2.1, com as probabilidades de todas as possíveis combinações quente/frio para as quatro observações. A coluna "Probabilidade Normalizada" trata da normalização dos valores para que esta coluna some valor 1.

Podemos concluir através dos valores da Tabela 2.1 que provavelmente as quatro observações foram tomadas a partir de três anos frios seguidos de um ano quente, baseado na sequência que possui a maior probabilidade.

Tabela 2.1: Probabilidades de Sequências dos Estados

Sequência	Probabilidade	Probabilidade Normalizada
QQQQ	0.000412	0.042787
QQQF	0.000035	0.003635
QQFQ	0.000706	0.073320
QQFF	0.000212	0.022017
QFQQ	0.000050	0.005193
QFQF	0.000004	0.000415
QFFQ	0.000302	0.031364
QFFF	0.000091	0.009451
FQQQ	0.001098	0.114031
FQQF	0.000094	0.009762
FQFQ	0.001882	0.195451
FQFF	0.000564	0.058573
FFQQ	0.000470	0.048811
FFQF	0.000040	0.004154
FFFQ	**0.002822**	**0.293073**
FFFF	0.000847	0.087963

2.1.2 Exemplo das Urnas

No exemplo adaptado de Dimuro et al. (2002), possuímos N urnas com bolas pretas ou brancas e uma moeda viciada, o observador inicia em uma das urnas, retira uma bola, observa a sua cor e recoloca-a na urna, a pessoa então arremessa a moeda e observa seu resultado, caso o resultado for cara, a pessoa continua na urna em que está, caso contrário troca de urna. Suponha então que possuímos duas urnas, a primeira possui 3 bolas brancas e 7 bolas pretas, já a segunda urna possui 5 bolas brancas e 5 bolas pretas. Suponhamos que as urnas possuem uma probabilidade de a pessoa escolher para iniciar os experimentos de 70% para a primeira urna e de 30% para a segunda urna. Também suponhamos que a moeda possui 90% de chances de apresentar cara e 10% de chances de apresentar coroa. Podemos representar o exemplo no modelo visual apresentado na Figura 2.4 adaptada de Dimuro et al. (2002). Também podemos observar o comportamento do modelo por uma representação diagramática como na Figura 2.5.

Suponha então que tenhamos três observações, sendo estas preto, branco, branco. Podemos calcular a probabilidade de tais bolas terem vindo de uma dada sequência de urnas através do teorema de Bayes (MORETTIN, 2000), onde temos a probabilidade de uma observação dado que um evento anterior ocorreu. Calculando todas as probabilidades para todas as possíveis combinações, podemos capturar o melhor resultado como a melhor aproximação de sequência

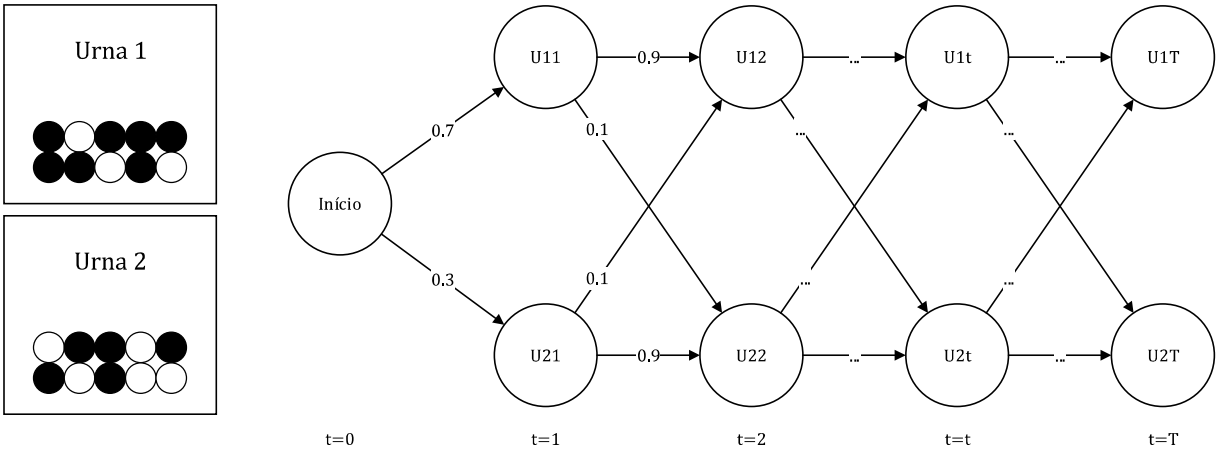


Figura 2.4: Representação visual do Modelo Oculto de Markov ao decorrer do tempo de 0 a T

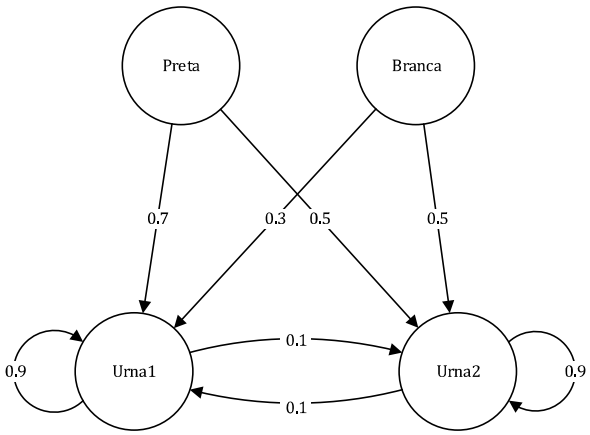


Figura 2.5: Representação diagramática do modelo das urnas

provável para tal geração de observações.

A partir das três observações citadas acima e dos pesos fornecidos anteriormente para a rede, podemos calcular as probabilidades como por exemplo para a sequência urna 1, urna 1, urna 1.

$$P(preto, branco, branco|urna1, urna1, urna1) = 0.7 \cdot 0.7 \cdot 0.9 \cdot 0.3 \cdot 0.9 \cdot 0.3 = 0.035721 \tag{2.10}$$

Analogamente podemos calcular as probabilidades para todas as sequências de estados como citado anteriormente, os resultados podem ser observados na Tabela 2.2.

Tabela 2.2: Probabilidades de Sequências dos Estados

Sequência	Probabilidade	Probabilidade normalizada
urna 1, urna 1, urna 1	**0.035721**	**0.406216**
urna 1, urna 1, urna 2	0.006615	0.075225
urna 1, urna 2, urna 1	0.000735	0.008358
urna 1, urna 2, urna 2	0.011025	0.125375
urna 2, urna 1, urna 1	0.001215	0.013817
urna 2, urna 1, urna 2	0.000225	0.002559
urna 2, urna 2, urna 1	0.002025	0.023028
urna 2, urna 2, urna 2	0,030375	0.345422

Com o valor de probabilidade de todas as sequências possíveis, podemos assumir que a sequência têm maior probabilidade de ter sido emitida pela sequência urna 1, urna 1, urna 1; com probabilidade de 40,62%.

Capítulo 3

Algoritmos e Ferramentas Utilizadas

3.1 Materiais

As amostras foram coletadas de 21 homens e 9 mulheres, utilizando um microfone Microsoft LifeCam HD-5000 e o software Audacity, versão 2.1.3, os áudios então foram exportados para o formato wav com frequência de 44100 Hz, mono com representação em 16 bits PCM sinalizado, em uma sala de aproximadamente 50 metros quadrados sem proteção acústica, apenas com o locutor em frente ao microfone previamente citado.

Para cada locutor, foram capturadas duas amostras das palavras “Frente”, “Trás”, “Esquerda”, “Direita” e “Para”, a última em sua forma paroxítona, totalizando 300 amostras de áudio que foram separadas em partes que continham apenas presença de voz.

Foi utilizado o software MATLAB® R2015a para execução dos códigos e coleta de resultados.

3.2 Algoritmos

3.2.1 *Mel-Frequency Cepstral Coefficients*

Geralmente empregada em processamento de áudio, a *Mel-Frequency Cepstral Coefficients* (MFCC) é uma transformação que busca retirar características dos sons de forma próxima a do ouvido humano (JARRETT, 2010). O áudio é cortado em pequenos segmentos com sobreposição como na Figura 3.1.

Para cada segmento é aplicada a Transformada de Fourier e o resultado já no domínio da frequência passa pela escala MEL para adaptação das frequências. Então, o espectro passa por

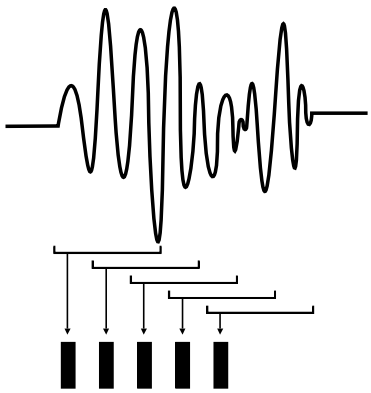


Figura 3.1: Representação visual da realização do método com um vetor discreto

uma separação em bandas de frequência, tais bandas sofrem então a aplicação de um filtro, tipicamente consistindo de triângulos sobrepostos como na Figura 3.2 (ALLEN, 2003), (MOLAU et al., 2001). Tais filtros procuram reduzir a influência de frequências altas já que são menos importantes ao ouvido humano.

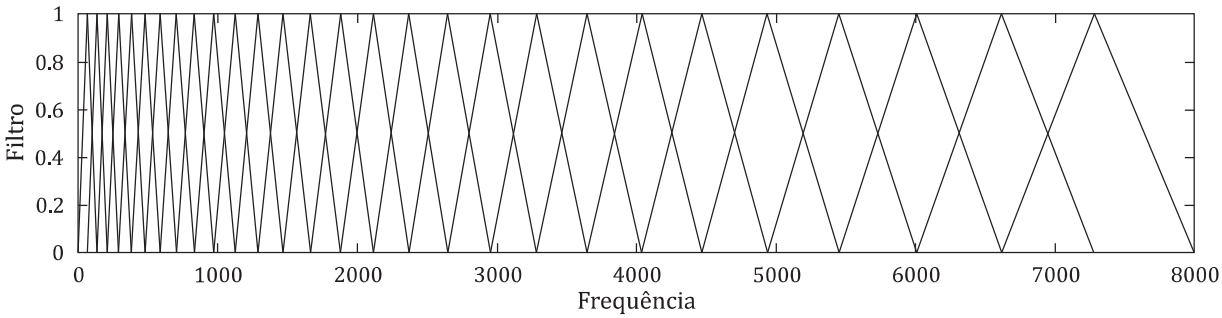


Figura 3.2: Representação gráfica do banco de filtros Mel

Escala MEL

A escala MEL relaciona a forma como percebemos o som à fórmula logarítmica da Equação 3.1, que pode ser visualizada no gráfico da Figura 3.3.

$$Mel = 2595 \log_{10}\left(1 + \frac{Hz}{700}\right) \tag{3.1}$$

Analogamente é possível definir a função inversa (JARRETT, 2010), que converte de Escala MEL novamente para Hz pela Equação 3.2.

$$Hz = 700(10^{\frac{m}{2595}} - 1) \tag{3.2}$$

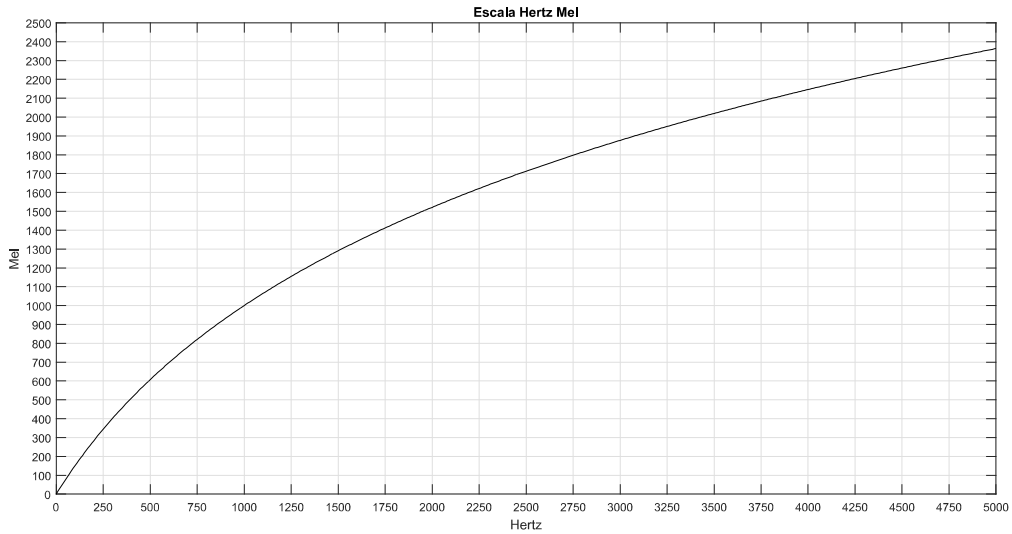


Figura 3.3: Representação gráfica da conversão de Hertz para escala MEL

3.2.2 Modelo de Mistura Gaussiana

Um Modelo de Mistura Gaussiana (GMM, do inglês *Gaussian Mixture Models*) é um modelo probabilístico para representar a presença de subpopulações em uma população geral sem que isto esteja necessariamente identificado (ALLILI, 2010), isto é representado através de um somatório de distribuições gaussianas (VYAS, 2013) como pode ser observado na Figura 3.4.

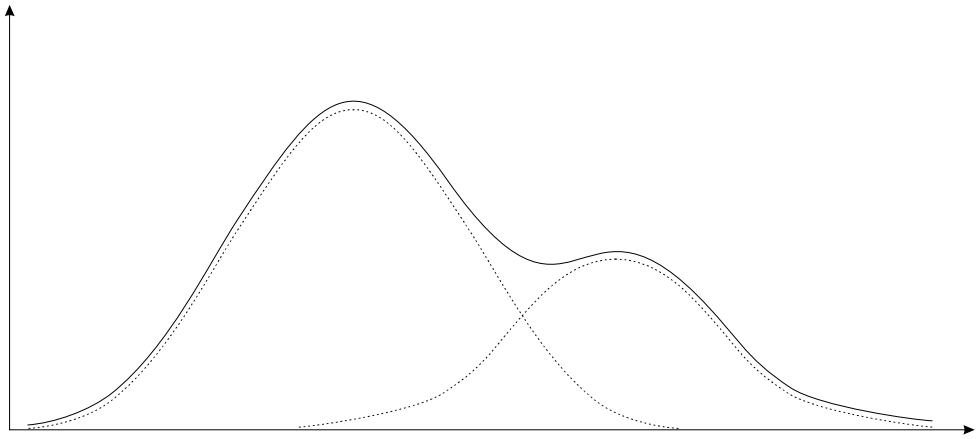


Figura 3.4: Exemplo de representação com uma distribuição (linha contínua) representada por duas distribuições gaussianas (linhas tracejadas)

Utilizando uma quantidade adequada de distribuições gaussianas e parâmetros adequados para tais, praticamente qualquer distribuição contínua pode ser representada com alguma acurácia (ALLILI, 2010). Matematicamente, um GMM é representado como uma somatória de

G distribuições gaussianas como representado pela Equação 3.3, que pode ser visualizada na Figura 3.5 adaptada de Reynolds e Rose (1995).

$$\eta(\vec{x}|\lambda) = \sum_{i=1}^{G} \eta_i \phi_i(\vec{x}) \tag{3.3}$$

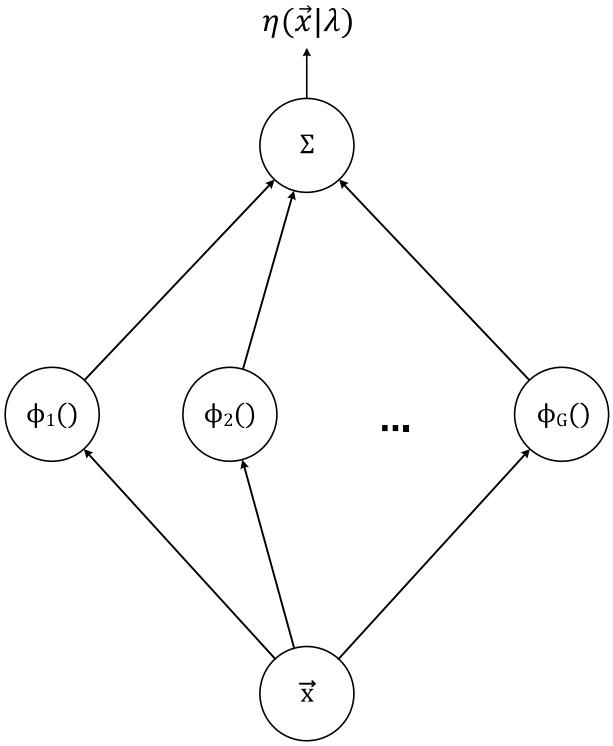


Figura 3.5: Representação visual da equação 3.3

Onde $\vec{x}$ é um vetor com D dimensões, $\phi_i(\vec{x})$ com $i = 1, \ldots, G$ são as funções de distribuição gaussiana e η_i com $i = 1, \ldots, G$ são os pesos de mistura das distribuições.

Para se criar um GMM, normalmente se criam parâmetros aleatórios que são otimizados através de uma série iterativa (STUTTLE, 2003). Uma das abordagens mais comumente usadas é o algoritmo *Expectation-Maximization* (ALLILI, 2010) que consiste em um algoritmo iterativo de duas etapas, uma etapa de expectativa e uma etapa de maximização (FILHO; DREWS; MARCOLINO, 2013).

Na etapa de estimação (Etapa E) é calculada a probabilidade de cada ponto pertencer a cada Gaussiana e é calculada a acurácia da estimativa. Já na etapa de maximização (Etapa M) os pesos e componentes das curvas são otimizadas. Após otimizado, um GMM provê as curvas Gaussianas para utilização mais pontual em um Modelo Oculto de Markov, por exemplo.

3.2.3 Algoritmo de Viterbi

A introdução do Algoritmo de Viterbi a seguir foi adaptada de Dimuro et al. (2002). O Algoritmo de Viterbi é um algoritmo que encontra de maneira direta em passos a sequência ótima de estados de um Modelo Oculto de Markov dada uma sequência de observações. Seja γ_i^t a probabilidade de se observar, no estado x_i no tempo t a observação O_t. Dada uma sequência de observações, γ^1 pode ser calculada como:

$$\gamma^1 = [\pi_i b_i(O_1)] \tag{3.4}$$

Para o cálculo dos demais passos, utiliza-se a probabilidade dos estados anteriores com as probabilidades de transição como apresentado na Equação 3.5.

$$\gamma^n = [(\gamma_i^{n-1} b_i(O_n))] \tag{3.5}$$

Para registrar os caminhos mais prováveis, define-se um vetor $r(t)$ que recebe as posições do máximo em γ como apresentado na Equação 3.6, onde N representa o tamanho do vetor γ.

$$r(i) = max_{1 \leqslant j \leqslant N} \{\gamma_j^i\} \tag{3.6}$$

Exemplo das urnas

Retomando o exemplo de 2.1.2, podemos calcular a sequência de urnas mais provável para a sequência de observações preto, branco, branco. Sabendo que

$$O = (preto, branco, branco) \tag{3.7}$$

$$\pi = \begin{array}{c} \text{Urna 1} \quad \text{Urna 2} \\ \left[\begin{array}{cc} 0.7 & 0.3 \end{array}\right] \end{array} \tag{3.8}$$

$$B(preto) = \begin{array}{c} \text{Urna 1} \quad \text{Urna 2} \\ \left[\begin{array}{cc} 0.7 & 0.5 \end{array}\right] \end{array} \tag{3.9}$$

têm-se para γ^1

$$\gamma^1 = [\pi_i b_i(O_1)] = \begin{bmatrix} \pi_1 b_1(preto) \\ \pi_2 b_2(preto) \end{bmatrix} = \begin{bmatrix} 0.49 \\ 0.15 \end{bmatrix} \tag{3.10}$$

$$r(1) = (1(x_1(O_1)) \quad \ldots \quad \ldots) \tag{3.11}$$

Para γ^2 e sucessivamente têm-se

$$\gamma^2 = [\gamma_i^1 b_i(O_2)] = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix} \begin{bmatrix} \gamma_1^1 b_1(branco) \\ \gamma_2^1 b_2(branco) \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.082 \end{bmatrix} \tag{3.12}$$

$$r(2) = (1(x_1(O_1)) \quad 1(x_1(O_2)) \quad \ldots) \tag{3.13}$$

$$\gamma^3 = [\gamma_i^2 b_i(O_3)] = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix} \begin{bmatrix} \gamma_1^2 b_1(branco) \\ \gamma_2^2 b_2(branco) \end{bmatrix} = \begin{bmatrix} 0.042 \\ 0.041 \end{bmatrix} \tag{3.14}$$

$$r(3) = (1(x_1(O_1)) \quad 1(x_1(O_2)) \quad 1(x_1(O_3))) \tag{3.15}$$

Onde o vetor r carrega a sequência ideal, sendo esta urna 1, urna 1, urna 1 como constatado por verificação na Tabela 2.2. Para o Modelo Oculto de Markov, o Algoritmo de Viterbi sempre fornecerá a sequência mais provável dadas as observações (RABINER, 1989).

Capítulo 4

Implementação e testes

4.1 Implementação

Foi utilizado o algoritmo fornecido em Lee (2012), apresentado no Apêndice A, que implementa uma solução para o problema de reconhecimento de fala utilizando o Modelo Oculto de Markov, *Mel-Frequency Cepstral Coefficients* e Modelo de Mistura Gaussiana feito em MATLAB®.

A função principal inicia com a chamada das funções que geram as listas de arquivos e dados necessários para execução, em seguida lê os arquivos em formato 'wav', extrai os parâmetros MFCC e salva-os. Após a extração, se iniciam as iterações de treinamento, onde a primeira iteração não possui matrizes anteriores para serem aprimoradas e é responsável pelos cálculos do primeiro modelo, nas iterações seguintes o modelo anterior é lido e aprimorado para melhorar a acurácia. Para cada iteração do treinamento, é executada uma iteração de testes para demonstração da situação atual de treinamento. Um diagrama pode ser observado na Figura 4.1.

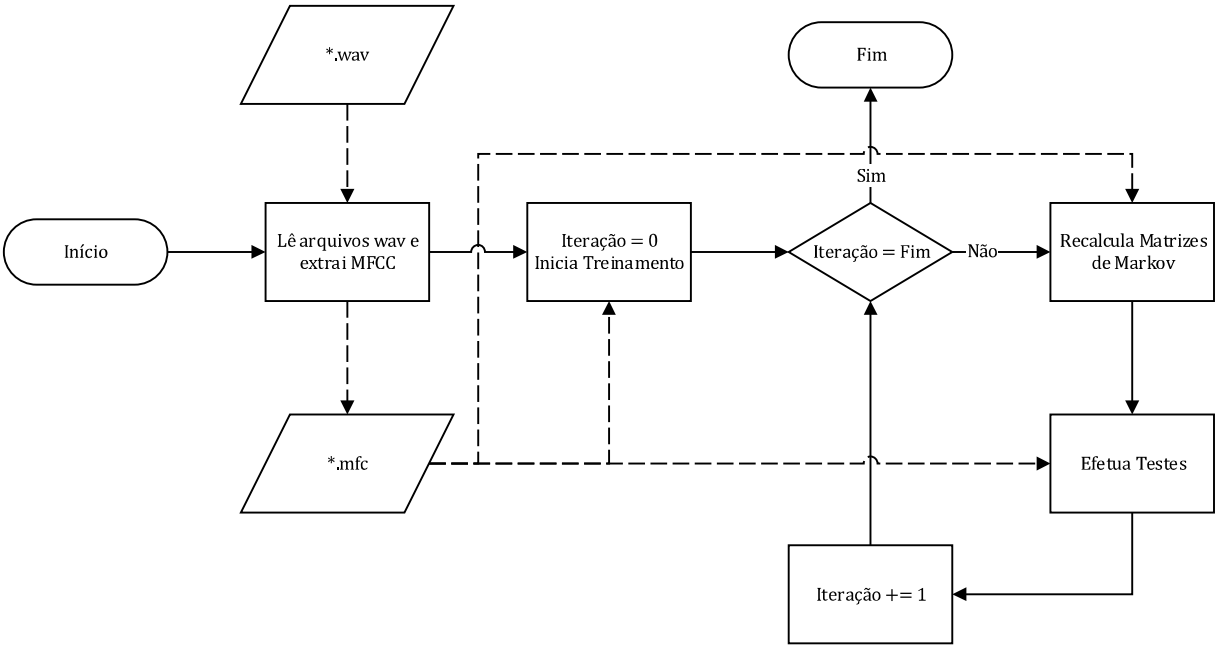


Figura 4.1: Diagrama simplificado do funcionamento do algoritmo

Para a execução dos testes geram-se as listas de arquivos e dados de testes necessários, extraem-se os parâmetros MFCC dos arquivos ‘wav’, carrega-se a máquina de transições salva pelo procedimento de treinamento e realizam-se os testes sobre esta máquina com os arquivos especificados pelas listas de testes.

A partir de tais funcionalidades, foi implementada a tela apresentada na Figura 4.2, onde o botão “Gravar” inicia a gravação de áudio para reconhecimento. Ao ser pressionado, o botão troca seu texto para “Parar” como apresentado na Figura 4.3, que ao ser pressionado novamente encerra a gravação e inicia automaticamente o reconhecimento. Após concluído o reconhecimento, a palavra reconhecida será escrita no campo de texto cercado pela borda “Palavra Reconhecida”, como no exemplo da Figura 4.4 onde foi reconhecida a palavra “Direita” para a gravação feita, com suas amplitudes exibidas no painel superior.

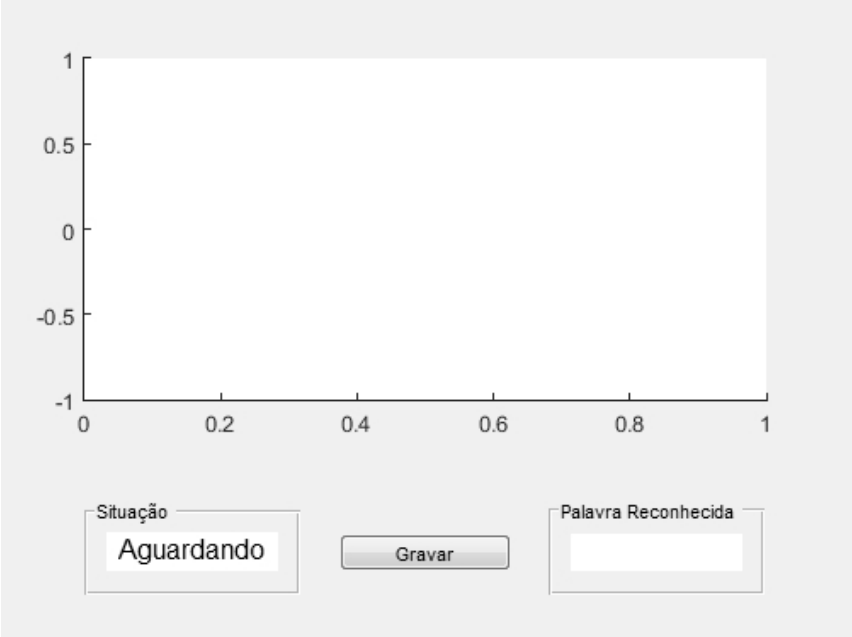


Figura 4.2: Tela Inicial do programa

Figura 4.3: Tela após pressionar o botão “Gravar”

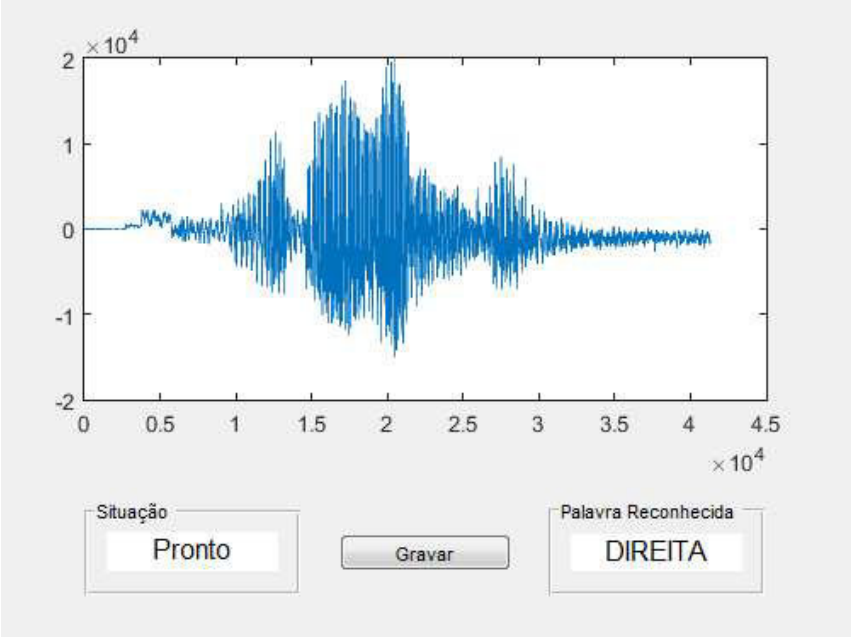
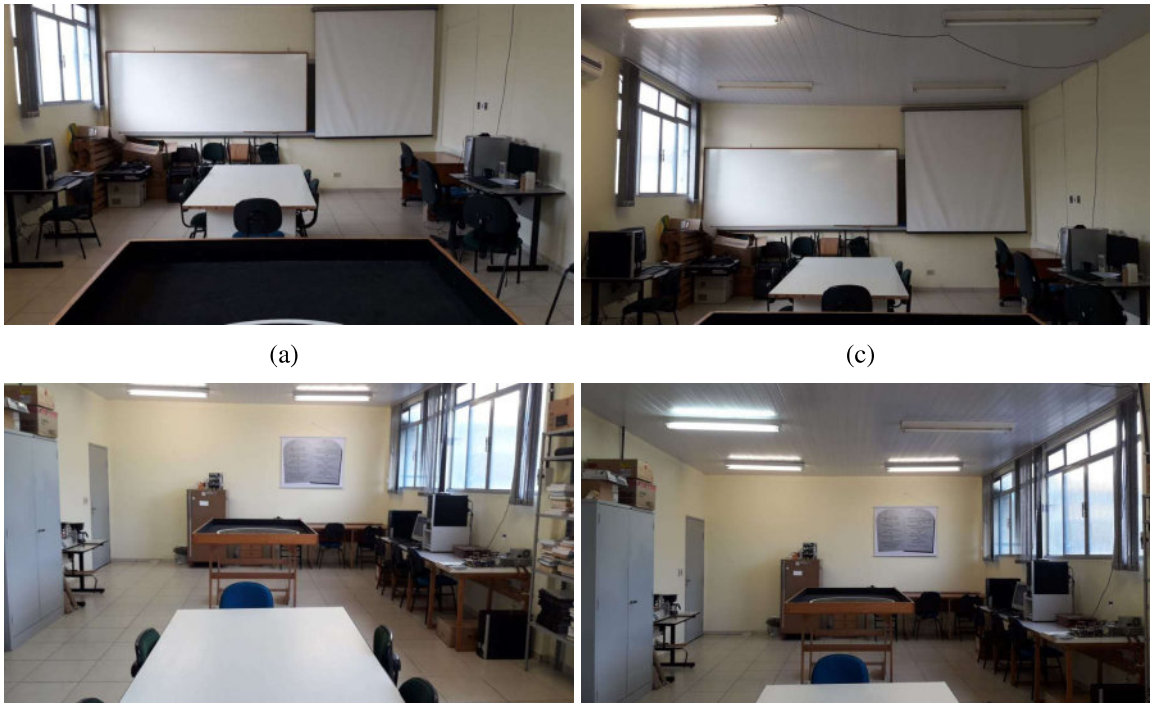


Figura 4.4: Tela após o reconhecimento da palavra “Direita”

4.2 Testes

Com o microfone citado na Seção 3.1, foram realizados testes, onde o locutor foi colocado em frente ao microfone e pronunciou as mesmas palavras para que o modelo previamente treinado pudesse tentar reconhecer as palavras ditas.

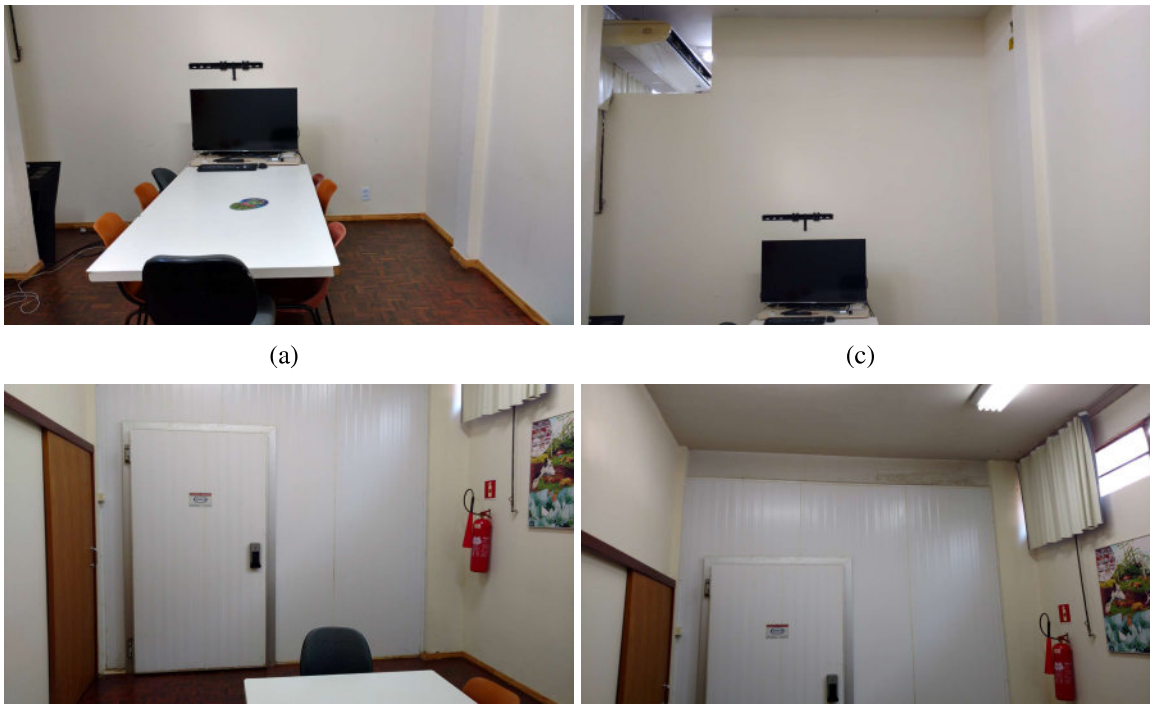
A aplicação foi testada em dois ambientes: Uma sala com aproximadamente 50 metros quadrados em completo silêncio sem proteção acústica, apresentada nas Figuras 4.5(a), 4.5(b), 4.5(c) e 4.5(d); e em um ambiente empresarial com aproximadamente 20 metros quadrados, apresentado nas Figuras 4.6(a), 4.6(b), 4.6(c) e 4.6(d), também sem proteção acústica, que possuía bastante ruídos como ar-condicionado, telefones tocando constantemente e em alguns momentos barulho de veículos.



(a) (c)

(b) (d)

Figura 4.5: Laboratório de Robótica



(a) (c)

(b) (d)

Figura 4.6: Sala empresarial

4.2.1 Resultados

Foram efetuados testes com 16 pessoas que participaram do treinamento e 41 pessoas que não participaram do treinamento. Das pessoas que não participaram do treinamento, 14 foram gravadas em ambiente empresarial e o restante em ambiente controlado. Todas as pessoas ditaram duas vezes todas as palavras no microfone em ordem aleatória e seus resultados foram analisados e inseridos nas Tabelas 4.1, 4.2 e 4.3. A primeira apresenta os resultados dos testes com as pessoas que participaram do treino, a segunda para as pessoas que não participaram e a terceira apresenta uma média de resultados.

Tabela 4.1: Resultados obtidos com as palavras ditadas por pessoas que participaram do treino

Palavra	% de reconhecimento correto	Quantidade de reconhecimentos em				
		Frente	Trás	Esquerda	Direita	Para
Frente	100.00	32	0	0	0	0
Trás	100.00	0	32	0	0	0
Esquerda	84.00	0	0	27	5	0
Direita	100.00	0	0	0	32	0
Para	59.00	5	4	2	2	19

Tabela 4.2: Resultados obtidos com as palavras ditadas por pessoas que não participaram do treino

Palavra	% de reconhecimento correto	Quantidade de reconhecimentos em				
		Frente	Trás	Esquerda	Direita	Para
Frente	97.00	80	0	1	1	0
Trás	95.00	2	78	0	0	2
Esquerda	84.00	2	0	69	11	0
Direita	91.00	7	0	0	75	0
Para	73.00	9	6	2	5	60

Tabela 4.3: Resultados obtidos com as palavras ditadas - Média geral

Palavra	% de reconhecimento correto	Quantidade de reconhecimentos em				
		Frente	Trás	Esquerda	Direita	Para
Frente	98.00	112	0	1	1	0
Trás	96.00	2	110	0	0	2
Esquerda	84.00	2	0	96	16	0
Direita	93.00	7	0	0	107	0
Para	69.00	14	10	4	7	79

Para as pessoas que participaram do treinamento, a taxa média de acertos foi de 88.60%, para as pessoas não treinadas, a taxa foi de 88.00%. Ambos os casos tiveram juntos uma taxa média de acertos de 88.00%. Apesar de existirem amostras de grupos de ambientes distintos, ambos ambientes apresentaram taxa semelhante de erros e acertos.

Capítulo 5

Considerações finais e trabalhos futuros

5.1 Considerações finais

A partir dos testes realizados, pode-se comprovar que o Modelo Oculto de Markov com auxílio de recursos providos por *Mel-Frequency Cepstral Coefficients* e Modelo de Mistura Gaussiana possui uma boa taxa de acerto das palavras do teste realizado, atingindo 88% de acerto nos testes realizados.

Provavelmente, a diferença entre a fonética e energia das palavras contribuiu para tal resultado já que a influência de pequenos ruídos pareceu não ser muito significativa.

Para análise dos resultados, foram utilizados como base trabalhos semelhantes encontrados na literatura. O primeiro é o trabalho de Rodrigues (2009) que implementou uma rede neural artificial treinada com o algoritmo *backpropagation*, atingindo taxa de acerto médio de 91%; em seguida, o trabalho de Manica (2014), que também utilizou uma rede neural artificial do tipo *Feed-forward backpropagation*, obteve uma taxa de acerto de 87% com amostras que participaram do treinamento e na fase de testes com pessoas que não participaram do treinamento, acabou por obter taxas de acerto entre 17 e 43%; o trabalho de Bissoli et al. (2013) que treinaram o software *Julius* que pode assumir um estado de confusão e pedir que o usuário repita o comando, neste caso eles obtiveram 89,1% de acertos sendo que no restante não tiveram erros, apenas estados de confusão; por fim, Barcelos et al. (2007) utilizaram o software *IBM ViaVoice* e obtiveram uma taxa de acertos em torno de 95%.

Com base nos trabalhos citados, o Modelo Oculto de Markov obteve resultados bastante satisfatórios já que, além de atingir uma taxa de acertos próxima a de outros modelos da literatura, manteve a taxa de acerto com pessoas que participaram ou não do treinamento e até mesmo em

ambiente ruidoso.

5.2 Trabalhos futuros

Como sugestão para trabalhos futuros, o software poderia contar com algum algoritmo de redução de ruídos, podendo assim trabalhar em ambientes mais ruidosos. Poderia-se também desenvolver um treinamento mais extensivo, com mais amostras e mais sotaques para maior acurácia do modelo.

Pode-se utilizar o software como base para uma aplicação em um aparelho móvel como um robô ou uma cadeira de rodas para prover comandos automáticos para tais aparelhos.

Apêndice A

Códigos de Lee (2012)

Algoritmo 1 main_train_test_EM

```
generate_isolated_digits_ti_training_list_mat;
generate_isolated_digits_ti_testing_list_mat;
dr_wav2mfcc_e_d_a_gen;

MODEL_NO=7;
feature_file_format='htk';
ITERATION_BEGIN=1;
ITERATION_END=10;

for STATE_NO=10:10
    DIM=39;
    model_filename_prefix='models/EM_models';
    traininglist_filename='training_list.mat';
    testinglist_filename='testing_list.mat';
    accuracy_filename=sprintf('recog_rate_S

    total_log_prob=zeros(1,ITERATION_END);
    recog_rate_EM_Viterbi=zeros(1,ITERATION_END);

    for iter=ITERATION_BEGIN:ITERATION_END
        model_filename_new=[model_filename_prefix, '_S', int2str(S...
TATE_NO), '_iter', int2str(iter),  '.mat'];
        if iter==1
            gau_hmm_init_train(traininglist_filename,model_filenam...
e_new,MODEL_NO,STATE_NO,DIM)
        else
            if ( exist(accuracy_filename,'file')  )
                load(accuracy_filename,'recog_rate_EM_Viterbi');
```

```
            end
            model_filename_old=[model_filename_prefix, '_S', int2s...
tr(STATE_NO), '_iter', int2str(iter-1),  '.mat'];
            total_log_prob(iter)=gau_hmm_EM_reestimation(trainingl...
ist_filename,model_filename_old,model_filename_new)
        end
        recog_rate_EM_Viterbi(iter)=gau_hmm_Viterbi_recognition(te...
stinglist_filename,model_filename_new,feature_file_format)
        save(accuracy_filename,'recog_rate_EM_Viterbi');
    end
end
```

Algoritmo 2 main_test

```
generate_isolated_digits_ti_testing_list_mat;
dr_wav2mfcc_e_d_a_gen;

feature_file_format='htk';
testinglist_filename='testing_list.mat';
model_filename_new='models\EM_models_S10_iter5.mat';
gau_hmm_Viterbi_recognition(testinglist_filename,model_filename_ne...
w,feature_file_format);
```

Algoritmo 3 generate_isolated_digits_ti_training_list_mat

```
list_filename='training_list.mat';
MODEL_NO=11;
dir1='mfcc_e_d_a/isolated_vogals_ti_train';
dir2={'OD'};
dir3={{'MQ','EB','JC','MD','LR','ML','MB','MO','RT','MA','AM','CB'...
,'JS','BC','FM','BA','LA','GP','JP','CA','AS','AB','AP','CD','GG',...
'AG','LT','AH'},{}};
wordids={'F','T','E','D','P'};

k=1;
for d2=1:length(dir2)
    for d3=1:length(dir3{d2})
        for w=1:length(wordids)
            for pass='1':'2'
                list{k,1}=w;
                list{k,2}=sprintf('%s/%s/%s/%c%c.mfc', dir1,dir2{d...
2},dir3{d2}{d3},wordids{w},pass);
                k=k+1;
            end
        end
    end
```

```
end
save(list_filename,'list');
```

Algoritmo 4 generate_isolated_digits_ti_testing_list_mat

```
clear;
list_filename='testing_list.mat';
MODEL_NO=11;
dir1='mfcc_e_d_a/isolated_vogals_ti_test';
dir2={'OD'};
dir3={{'MS'},{}};
wordids={'F','T','E','D','P'};

k=1;
for d2=1:length(dir2)
    for d3=1:length(dir3{d2})
        for w=1:length(wordids)
            for pass='1':'2'
                list{k,1}=w;
                list{k,2}=sprintf('%s/%s/%s/%c%c.mfc', dir1,dir2{d...
2},dir3{d2}{d3},wordids{w},pass);
                k=k+1;
            end
        end
    end
end
save(list_filename,'list');
```

Algoritmo 5 dr_wav2mfcc_e_d_a_gen

```
clear;
indir='wav';
in_filter='\.[Ww][Aa][Vv]';
outdir='mfcc_e_d_a';
out_ext='.mfc';
outfile_format='htk';
frame_size_sec = 0.025;
frame_shift_sec= 0.010;
use_hamming=1;
pre_emp=0;
bank_no=26;
cep_order=12;
lifter=22;

delta_win=2;
delta_win_weight = ones(1,2*delta_win+1);
```

```
dr_wav2mfcc_e_d_a(indir,in_filter,outdir,out_ext,outfile_format,fr...
ame_size_sec,frame_shift_sec,use_hamming,pre_emp,bank_no,cep_order...
,lifter,delta_win_weight);
```

Algoritmo 6 dr_wav2mfcc_e_d_a

```
function dr_wav2mfcc_e_d_a(indir,in_filter,outdir,out_ext,outfile_...
format,frame_size_sec,frame_shift_sec,use_hamming,pre_emp,bank_no,...
cep_order,lifter,delta_win_weight)
  if  indir(end) == '/' || indir(end) == '\'
      indir=indir(1:(end-1));
  end
  if  outdir(end) == '/' || outdir(end) == '\'
      ourdir=outdir(1:(end-1));
  end

  if exist(outdir) ~=7
      mkdir(outdir);
  end

  filelist=dir(indir);
  filelist_len=length(filelist);

  for k=3:filelist_len
      [pathstr,filenamek,ext] = fileparts(filelist(k).name);
      if filelist(k).isdir
          dr_wav2mfcc_e_d_a([indir filesep filenamek],in_filter,[o...
utdir filesep filenamek],out_ext,outfile_format,frame_size_sec,fra...
me_shift_sec,use_hamming,pre_emp,bank_no,cep_order,lifter,delta_wi...
n_weight);
      else
          if regexp(filelist(k).name,in_filter)
              infilename=fullfile(indir, filelist(k).name);
              outfilename=[outdir filesep filenamek out_ext];
              fwav2mfcc_e_d_a(infilename,outfilename,outfile_forma...
t,frame_size_sec,frame_shift_sec,use_hamming,pre_emp,bank_no,cep_o...
rder,lifter,delta_win_weight);
          end
      end
  end
```

Algoritmo 7 fwav2mfcc_e_d_a

```
function feature_seq=fwav2mfcc_e_d_a(infilename,outfilename,outfil...
e_format,frame_size_sec,frame_shift_sec,use_hamming,pre_emp,bank_n...
o,cep_order,lifter,delta_win_weight)
  [speech_raw, fs, bit_res]=wavread(infilename,'native');
  speech_raw=double(speech_raw);

  feature_seq=wav2mfcc_e_d_a(speech_raw,fs,frame_size_sec,frame_sh...
ift_sec,use_hamming,pre_emp,bank_no,cep_order,lifter,delta_win_wei...
ght);

  [dim frame_no]=size(feature_seq);

  switch lower(outfile_format)
      case 'htk'
          fout=fopen(outfilename,'w','b');
          fwrite(fout,frame_no,'int32');
          sampPeriod=round(frame_shift_sec*1E7);
          fwrite(fout,sampPeriod,'int32');
          sampSize=dim*4;
          fwrite(fout,sampSize,'int16');
          parmKind=838;
          fwrite(fout,parmKind,'int16');
      case 'b'
          fout=fopen(outfilename,'w','b');
      case 'ieee-be'
          fout=fopen(outfilename,'w','b');
      case 'l'
          fout=fopen(outfilename,'w','l');
      case 'ieee-le'
          fout=fopen(outfilename,'w','l');
      otherwise
          fout=fopen(outfilename,'w','n');
  end

  fwrite(fout, feature_seq,'float32');

  fclose(fout);
```

Algoritmo 8 wav2mfcc_e_d_a

```
function feature_seq=wav2mfcc_e_d_a(speech_raw,fs,frame_size_sec,f...
rame_shift_sec,use_hamming,pre_emp,bank_no,cep_order,lifter,delta_...
win_weight)
```

```
   mfcc=wav2mfcc(speech_raw,fs,frame_size_sec,frame_shift_sec,use_...
hamming,pre_emp,bank_no,cep_order,lifter);
   logpow=wav2logpow(speech_raw,fs,frame_size_sec,frame_shift_sec);
   d_fea=(0.01/frame_shift_sec)*slope([mfcc;logpow],delta_win_weig...
ht);
   dd_fea=(0.01/frame_shift_sec)*slope(d_fea,delta_win_weight);
   feature_seq=[mfcc;logpow;d_fea;dd_fea];
```

Algoritmo 9 wav2mfcc

```
function [mfcc,frame_no]=wav2mfcc(speech_raw,fs,frame_size_sec,fra...
me_shift_sec,use_hamming,pre_emp,bank_no,cep_order,lifter)
len=length(speech_raw);
speech=zeros(len,1);
speech(1)=speech_raw(1);
speech(2:end)=speech_raw(2:end)-pre_emp*speech_raw(1:end-1);

frame_size  = round(fs*frame_size_sec);
frame_shift = round(fs*frame_shift_sec);
frame_no    = floor( 1 + (len - frame_size)/frame_shift );

max_mf = 2595*log10(1 + 0.5*fs/700.0);
delta_mf=max_mf/(bank_no+1);
f=zeros(bank_no+2,1);
for m=1:bank_no+2
    f(m)=(10^((m-1)*delta_mf/2595)-1)*700.0;
end
mfcc_tran=zeros(cep_order,bank_no);
for k=1:cep_order
    for m=1:bank_no
        mfcc_tran(k,m)=sqrt(2/bank_no)*cos(k*pi/bank_no * (m-0.5));
    end
end

n=(1:cep_order)';
lifter_weighting=1+(lifter/2)*sin(pi*n/lifter);

k=(1:frame_size)';
h=0.54-0.46*cos(2*pi*(k-1)/(frame_size-1));

mfcc=zeros(cep_order,frame_no);
```

```
melspec=zeros(bank_no,frame_no);
for fr=1:frame_no
    s=speech((fr-1)*frame_shift+1:(fr-1)*frame_shift+frame_size);

    if use_hamming ~= 0
        s=s.*h;
    end
    fftN=2;
    while fftN<frame_size
        fftN=fftN*2;
    end
    PowerSpec=abs(fft(s,fftN));
    mel_power=zeros(bank_no,1);
    for m=1:bank_no
        kmin=f(m)/fs*fftN +1;
        kcen=f(m+1)/fs*fftN +1;
        kmax=f(m+2)/fs*fftN +1;
        for k=ceil(kmin):floor(kmax)
            mel_k=2595*log10(1 + (k-1)/fftN*fs/700.0);
            if k < kcen
                mel_power(m) = mel_power(m) + PowerSpec(k)*(mel_k-...
(m-1)*delta_mf)/delta_mf;
            else
                mel_power(m) = mel_power(m) + PowerSpec(k)*(1 - (m...
el_k-delta_mf*(m))/delta_mf);
            end
        end
    end

    mfcc(:,fr)=mfcc_tran*log(mel_power);

    mfcc(:,fr)=lifter_weighting.*mfcc(:,fr);

end
```

Algoritmo 10 wav2logpow

```
function [logpow,frame_no]=wav2logpow(speech_raw,fs,frame_size_sec...
,frame_shift_sec)
  len=length(speech_raw);

  frame_size=round(fs*frame_size_sec);
```

```
  frame_shift=round(fs*frame_shift_sec);
  frame_no= floor( 1 + (len - frame_size)/frame_shift );

  k=(1:frame_size)';
  h=0.54-0.46*cos(2*pi*(k-1)/(frame_size-1));

  logpow=-inf*ones(1,frame_no);

  for fr=1:frame_no
     s=speech_raw((fr-1)*frame_shift+1:(fr-1)*frame_shift+frame_si...
ze);

     logpow(fr)=log(s'*s);
  end
```

Algoritmo 11 slope

```
function out_seq=slope(in_seq,window)

  N=length(window);
  if rem(N,2) == 1
    delta_win = (N-1)/2;
    [dim, frame_no]=size(in_seq);

    new_in_seq=[in_seq(:,1)*ones(1,delta_win) in_seq in_seq(:,fram...
e_no)*ones(1,delta_win)];
    out_seq=0*in_seq;
    time_vec=-delta_win:delta_win;

    denominator=window*(time_vec.^2)';
    slope_vec=(window.*time_vec/denominator)';
    for fr=1:frame_no
      out_seq(:,fr)=new_in_seq(:,fr:fr+N-1) * slope_vec;
    end
  end
```

Algoritmo 12 gau_hmm_init_train

```
function  gau_hmm_init_train(traininglist_filename,model_filename,...
MODEL_NO,STATE_NO, dim )
if nargin == 0
    traininglist_filename='training_list.mat' ;
    model_filename='models.mat';
    MODEL_NO=11;
    STATE_NO=4;
```

```
    dim=12;
end
MIN_SELF_TRANSITION_COUNT=0;
load(traininglist_filename,'list');

mean_vec_i_m=zeros(dim,STATE_NO,MODEL_NO);
var_vec_i_m=zeros(dim,STATE_NO,MODEL_NO);
A_i_m=zeros(STATE_NO,MODEL_NO);

vector_sums_i_m=zeros(dim,STATE_NO,MODEL_NO);
var_vec_sums_i_m=zeros(dim,STATE_NO,MODEL_NO);
fr_no_i_m=zeros(STATE_NO,MODEL_NO);
self_tr_fr_no_i_m=zeros(STATE_NO,MODEL_NO);

utterance_no=size(list,1);
total_fr_no=0;
for k=1:utterance_no
    filename=list{k,2};
    m=list{k,1};
    fid=fopen(filename,'r');
    fseek(fid, 12, 'bof');
    c=fread(fid,'float','b');
    fclose(fid);
    fr_no=length(c)/dim;
    total_fr_no=total_fr_no+fr_no;
    c=reshape(c,dim,fr_no);

    for i=1:STATE_NO
        begin_fr=round(  fr_no*(i-1) /STATE_NO)+1;
        end_fr=round( fr_no*i /STATE_NO);
        seg_length=end_fr-begin_fr+1;
        vector_sums_i_m(:,i,m) = vector_sums_i_m(:,i,m) + sum(c(:,...
begin_fr:end_fr),2);
        var_vec_sums_i_m(:,i,m) = var_vec_sums_i_m(:,i,m) +  sum( ...
c(:,begin_fr:end_fr).*c(:,begin_fr:end_fr) , 2);
        fr_no_i_m(i,m)=fr_no_i_m(i,m)+seg_length;
        self_tr_fr_no_i_m(i,m)= self_tr_fr_no_i_m(i,m) + seg_lengt...
h-1;
    end
```

```
end

for m=1:MODEL_NO
    for i=1:STATE_NO
        mean_vec_i_m(:,i,m) = vector_sums_i_m(:,i,m) / fr_no_i_m(i...
,m);
        var_vec_i_m(:,i,m) = var_vec_sums_i_m(:,i,m) / fr_no_i_m(i...
,m);
        A_i_m(i,m)=(self_tr_fr_no_i_m(i,m)+MIN_SELF_TRANSITION_COU...
NT)/(fr_no_i_m(i,m)+2*MIN_SELF_TRANSITION_COUNT);
    end
end

overall_var_vec=sum(sum(var_vec_sums_i_m(:,:,:),3 ),2)/sum(sum(fr_...
no_i_m,2 ),1);
for m=1:MODEL_NO
    for i=1:STATE_NO
        var_vec_i_m(:,i,m)=overall_var_vec;
    end
end

save(model_filename, 'mean_vec_i_m', 'var_vec_i_m', 'A_i_m');
fprintf('init. train complete \n');
```

Algoritmo 13 gau_hmm_EM_reestimation

```
function [total_log_prob, total_fr_no]=gau_hmm_EM_reestimation(tra...
ininglist_filename,model_filename_old, model_filename_new)
if nargin < 1,  traininglist_filename='training_list10.mat';  end;
if nargin < 2,  model_filename_old='models.mat'; end;
if nargin < 3,  model_filename_new='models.mat'; end;
MIN_SELF_TRANSITION_COUNT=0.00;

load(model_filename_old, 'mean_vec_i_m', 'var_vec_i_m', 'A_i_m');
load(traininglist_filename,'list');

[dim,STATE_NO,MODEL_NO]=size(mean_vec_i_m);

vector_sums_i_m=zeros(dim,STATE_NO,MODEL_NO);
var_vec_sums_i_m=zeros(dim,STATE_NO,MODEL_NO);
fr_no_i_m=zeros(STATE_NO,MODEL_NO);
self_tr_fr_no_i_m=zeros(STATE_NO,MODEL_NO);
```

```
total_log_prob = 0;
total_fr_no = 0;
utterance_no=size(list,1);
for k=1:utterance_no
        filename=list{k,2};
        m=list{k,1};
        fid=fopen(filename,'r');
        fseek(fid, 12, 'bof');
        c=fread(fid,'float','b');
        fclose(fid);
        fr_no=length(c)/dim;
        c=reshape(c,dim,fr_no);
        [log_prob, pr_i_t, pr_self_tr_i_t ]=forward_backward_hmm_l...
og_math(c,mean_vec_i_m(:,:,m),var_vec_i_m(:,:,m),A_i_m(:,m));
        total_log_prob = total_log_prob + log_prob;
        total_fr_no = total_fr_no + fr_no;

        for i=1:STATE_NO
              fr_no_i_m(i,m)=fr_no_i_m(i,m)+sum(pr_i_t(i,:));
              self_tr_fr_no_i_m(i,m)=self_tr_fr_no_i_m(i,m)+sum(pr...
_self_tr_i_t(i,1:end-1));
              for fr=1:fr_no
                 vector_sums_i_m(:,i,m) = vector_sums_i_m(:,i,m) +...
  pr_i_t(i,fr)*c(:,fr);
                 var_vec_sums_i_m(:,i,m) =var_vec_sums_i_m(:,i,m) ...
+ pr_i_t(i,fr)*(c(:,fr)-mean_vec_i_m(:,i,m)).*(c(:,fr)-mean_vec_i_...
m(:,i,m));
              end
        end
end

old_mean_vec_i_m=mean_vec_i_m;
old_var_vec_i_m= var_vec_i_m;
old_A_i_m=A_i_m;
for m=1:MODEL_NO
   for i=1:STATE_NO;
         mean_vec_i_m(:,i,m) = vector_sums_i_m(:,i,m)/ fr_no_i_m(i...
,m);
```

```
        var_vec_i_m(:,i,m)= var_vec_sums_i_m(:,i,m) /  fr_no_i_m(...
i,m);
        A_i_m(i,m)=(self_tr_fr_no_i_m(i,m)+MIN_SELF_TRANSITION_CO...
UNT) /(fr_no_i_m(i,m)+2*MIN_SELF_TRANSITION_COUNT);
   end
end

var_new_to_old_ratio=var_vec_i_m ./ old_var_vec_i_m;
save(model_filename_new, 'mean_vec_i_m', 'var_vec_i_m', 'A_i_m');
fprintf('re-estimation complete \n');
```

Algoritmo 14 forward_backward_hmm_log_math

```
function [log_prob, pr_i_t,  pr_self_tr_i_t ]=forward_backward_hmm...
_log_math( V, mean_vec_i, var_vec_i, A_i  )
[dim , N]=size(mean_vec_i);
[dim2 , T]=size(V);

[log_prob,  logfw, logObsevation_i_t ]=forward_hmm_log_math(V, mea...
n_vec_i, var_vec_i, A_i );
pr_self_tr_i_t=zeros(N,T);

logbw=ones(N,T)*(-inf);
t=T;
logbw(N,T)=log(1-A_i(N));
for t=T-1:-1:1
    for i=1:N
        if i==N
            logbw(i,t)= log(A_i(i))+ logObsevation_i_t(i,t+1) + lo...
gbw(i,t+1);
            pr_self_tr_i_t(i,t)=exp(logfw(i,t)+  log(A_i(i))+ logO...
bsevation_i_t(i,t+1)  + logbw(i,t+1)-log_prob);
        else
            logbw(i,t)=logSum([ (log(A_i(i))+ logObsevation_i_t(i,...
t+1) + logbw(i,t+1)) , (log(1-(A_i(i))) + logObsevation_i_t(i+1,t+...
1)+ logbw(i+1,t+1) )] );
            pr_self_tr_i_t(i,t)=exp(logfw(i,t)+ log(A_i(i))+logObs...
evation_i_t(i,t+1) + logbw(i,t+1)-log_prob);
        end
    end
end
```

```
pr_i_t= exp(  logfw+logbw - log_prob );
count_at_t(1:T)=sum(pr_i_t,1);
count_at_t=squeeze(count_at_t);
if  (sum(count_at_t) -T) > 1E-6
    diff=sum(count_at_t) -T ;
end
end
```

Algoritmo 15 forward_hmm_log_math

```
function [log_pr, varargout]=forward_hmm_log_math(V, mean_vec_i, v...
ar_vec_i,  A_i  )
[dim , N]=size(mean_vec_i);
[dim2 , T]=size(V);
logObsevation_i_t=zeros(N,T);
for t=1:T
    for i=1:N
        logObsevation_i_t(i,t)=logDiagGaussian(V(:,t),mean_vec_i(:...
,i),var_vec_i(:,i));
    end
end
logfw=ones(N,T)*(-inf);
t=1;
logfw(1,1)=logObsevation_i_t(1,1);

for t=2:T
    i=1;
        logfw(i,t)=logfw(i,t-1) + log(A_i(i))+logObsevation_i_t(i,t);
   for i=2:N
        logfw(i,t)= logSum(  [ (logfw(i-1,t-1) +log(1-A_i(i-1)) ) ...
, (logfw(i,t-1) + log(A_i(i))) ] ) + logObsevation_i_t(i,t);
   end
end

 log_pr=logfw(N,T) + log(1-A_i(N));

if nargout > 1
    varargout(1)= {logfw};
end
if nargout > 2
    varargout(2)= {logObsevation_i_t};
end

end
```

Algoritmo 16 logDiagGaussian

```
function log_pr=logDiagGaussian(v,u,K1)
 dim=length(K1);
 log_pr=-1/2*dim*log(2*pi) -1/2*sum(log(K1))- 1/2*sum((v-u).*(v-u)...
./K1);
 end
```

Algoritmo 17 logSum

```
function logS=logSum(x)
    xmax=overall_max(x);
    if isinf(xmax)
        logS=xmax;
    else
        logS=xmax + log(overall_sum(exp(x-xmax)));
    end
end
```

Algoritmo 18 overall_max

```
function xmax=overall_max(x)
   n=ndims(x);
   xmax=x;
    for k=n:-1:1
        xmax=max(xmax,[ ], k);
    end
end
```

Algoritmo 19 overall_sum

```
function s=overall_sum(x)
    n=ndims(x);
    s=x;
    for k=n:-1:1
        s=sum(s,k);
    end
end
```

Algoritmo 20 gau_hmm_Viterbi_recognition

```
function [recog_rate, varargout]=gau_hmm_Viterbi_recognition(testi...
nglist_filename,model_filename,feature_file_format)
if nargin < 1,  testinglist_filename='testing_list10.mat'; end;
if nargin < 2,  model_filename='models.mat'; end;
if nargin < 3,  feature_file_format='no-head'; end;

load(testinglist_filename, 'list');
load(model_filename, 'mean_vec_i_m', 'var_vec_i_m', 'A_i_m');

[dim,STATE_NO,MODEL_NO]=size(mean_vec_i_m);
correct_count=0;
error_count=0;

utterance_no=size(list,1);
for k=1:utterance_no
        filename=list{k,2};
        fid=fopen(filename,'r');
        fprintf(fout,' %s\n', filename);
        if strcmpi(feature_file_format, 'HTK')
           fseek(fid, 12, 'bof');
        end
        c=fread(fid,'float','b');
        fclose(fid);
        fr_no=length(c)/dim;
        c=reshape(c,dim,fr_no);
        scores=ones(MODEL_NO,1)*(-inf);
        for m=1:MODEL_NO
            scores(m)=fun_viterbi_hmm( c, mean_vec_i_m(:,:,m), var...
_vec_i_m(:,:,m), A_i_m(:,m) );
        end
        fprintf(fout, scores);
        [temp, m_max]=max(scores);
        fprintf('word= %d   recog= %d \n',list{k,1},m_max);
        if (m_max==list{k,1})
            correct_count=correct_count+1;
        else
            error_count=error_count+1;
        end
end
```

```
total_count=(error_count+correct_count);
recog_rate=correct_count/total_count;
if nargout > 1
    varargout(1)= { total_count };
end

fprintf('recognition rate = %5.2f  error count =%d   correct count...
 =%d  total_count=%d \n',recog_rate*100,error_count, correct_count...
, total_count);
```

Algoritmo 21 fun_viterbi_hmm

```
function [Fopt, varargout]=fun_viterbi_hmm(V, mean_vec_i, var_vec_...
i, A_i)
[dim ,N]=size(mean_vec_i);
[dim , T]=size(V);
P=cell(N,T);
f=ones(N,T)*(-inf);

for i=1:N
        P{i,1}=i;
end
t=1;i=1;
f(i,t)=logDiagGaussian(V(:,t),mean_vec_i(:,i),var_vec_i(:,i));

for t=2:T
    i=1;
        f(i,t)=f(i,t-1)+log(A_i(i))+ logDiagGaussian(V(:,t),mean_v...
ec_i(:,i),var_vec_i(:,i) );
    for i=2:N
        [f(i,t), pi] = max(  [ f(i-1,t-1) + log(1-A_i(i-1))  ,  f(...
i,t-1) + log(A_i(i)) ] );
        f(i,t)=f(i,t)+logDiagGaussian(V(:,t),mean_vec_i(:,i),var_v...
ec_i(:,i) );
        P{i,t}=[P{i+pi-2,t-1}  i ];
   end
end

Fopt=  f(N,T) + log(1-A_i(N)) ;
if nargout > 1
     varargout(1)= { P{N,T} };
end
end
```

Algoritmo 22 logDiagGaussian

```
function log_pr=logDiagGaussian(v,u,K1)
 dim=length(K1);
 log_pr=-1/2*dim*log(2*pi) -1/2*sum(log(K1))- 1/2*sum((v-u).*(v-u)...
./K1);
 end
```

Referências Bibliográficas

ALAM, M. M.; MEZBAHUDDIN, M.; SHOMA, S. N. Election result prediction system using hidden markov model [hmm]. *International Journal of Computer Applications*, Citeseer, v. 129, p. 1–4, 2015.

ALLEN, J. F. *Signal Processing for Speech Recognition*. 2003. Disponível em: <http://www.cs.rochester.edu/u/james/CSC248/Lec13.pdf>. Acesso em: 08 jun 2018.

ALLILI, M. S. *A short tutorial on Gaussian Mixture Models*. 2010. Disponível em: <http://www.computerrobotvision.org/2010/tutorial_day/GMM_said_crv10_tutorial.pdf>. Acesso em: 05 jun 2018.

BANDEIRA, L. P. F.; JUNIOR, O. A.; COSTA, R. V. M. *Sistema de Reconhecimento de Fala Usando Cadeias Ocultas de Markov*. 2008. Disponível em: <https://www.slideshare.net/mwanalezi/reconhecimento-de-fala-usando-cadeias-de-markov>. Acesso em: 01 dez 2017.

BARCELOS, A. et al. *Reconhecimento de Voz para Aplicação em Cadeira de Rodas*. 2007. Disponível em: <https://www.aedb.br/seget/arquivos/artigos08/445_Reconhecimento\%20de\%20Voz\%20aplicado\%20na\%20cadeira\%20de\%20rodas.pdf>. Acesso em: 13 jun 2018.

BAUM, L. E. An inequality and associated maximization thechnique in statistical estimation for probabilistic functions of markov process. *Inequalities*, v. 3, p. 1–8, 1972.

BAUM, L. E.; EAGON, J. A. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, v. 73, n. 3, p. 360–363, 1967.

BAUM, L. E.; PETRIE, T. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, v. 37, n. 6, p. 1554–1563, 1966.

BISSOLI, A. L. et al. Geração e suavização de trajetórias automáticas para uma simulação residencial de uma cadeira de rodas comandada por voz. *XI Simpósio Brasileiro Automação Inteligente*, 2013.

BRAGA, P. *Reconhecimento de voz dependente de locutor utilizando Redes Neurais Artificiais*. Dissertação (Monografia) — Escola Politécnica de Pernambuco. Universidade de Pernambuco, Pernambuco, 05 2006.

BRESOLIN, A. A. *Estudo do Reconhecimento de Voz para o Acionamento de Equipamentos Elétricos via Comandos em Português*. Dissertação (Dissertação de Mestrado) — Universidade do Estado de Santa Catarina, Departamento de Engenharia Elétrica, Santa Catarina, 08 2003.

DAVIS, K.; BIDDULPH, R.; BALASHEK, S. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, ASA, v. 24, n. 6, p. 637–642, 1952.

DIMURO, G. P. et al. Modelos de markov e aplicações. *VI Oficina de Inteligência Artificial, Pelotas: Educat*, p. 37–59, 2002.

ESPINDOLA, L. S. *Um estudo sobre Modelos Ocultos de Markov*. Dissertação (Dissertação de Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, Pós Graduação em Ciência da Computaçao, Porto Alegre, 06 2009.

FILHO, S. S.; DREWS, P. J.; MARCOLINO, L. V. F. *Mistura de Gaussianas Uma Abordagem Rápida para Modelar Nuvem de Pontos*. 2013. Disponível em: <http://www.sbai2013.ufc.br/pdfs/8163.pdf>. Acesso em: 13 jun 2018.

JARRETT, N. W. *Changing the MFCC Intensity Function*. Tese (Doutorado) — University at Albany, Department of Mathematics & Statistics, 2010.

JUANG, B. H.; RABINER, L. R. Hidden markov models for speech recognition. *Technometrics*, Taylor & Francis, v. 33, n. 3, p. 251–272, 1991.

KARLOF, C.; WAGNER, D. Hidden markov model cryptanalysis. In: SPRINGER. *International Workshop on Cryptographic Hardware and Embedded Systems*. [S.l.], 2003. p. 17–34.

LEE, L.-M. *HMM Speech Recognition in Matlab*. 2012. Disponível em: <https://sourceforge.net/projects/hmm-asr-matlab/files/V1.01/>. Acesso em: 01 dez 2017.

LUGER, G. F. *Inteligência Artificial*. 6. ed. [S.l.]: Bookman, 2004.

MANICA, C. L. *Implementação de um módulo de reconhecimento de fala como interface de comando para robôs LEGO MINDSTORMS NXT 2.0*. Dissertação (Monografia) — Universidade Estadual do Oeste do Paraná, Ciência da Computaçao, Cascavel, 11 2014.

MARKOV, A. Extension of the law of large numbers to quantities, depending on each other. *Izvestia Physico-Mathematical Society at the Kazan University*, v. 15, p. 135–156, 1906.

MENEZES, P. B. *Linguagens formais e autômatos*. 2. ed. [S.l.]: Sagra-Luzzato, 1998.

MOLAU, S. et al. Computing mel-frequency cepstral coefficients on the power spectrum. In: IEEE. *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*. [S.l.], 2001. v. 1, p. 73–76.

MOLER, C. *The Origins of MATLAB*. 2004. Disponível em: <https://es.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html>. Acesso em: 01 dez 2017.

MORETTIN, L. G. *Estatística Básica - Probabilidade*. 1. ed. [S.l.]: Makron Books, 2000.

NILSSON, M.; EJNARSSON, M. *Speech recognition using hidden markov model*. 2002. Disponível em: <https://www.diva-portal.org/smash/get/diva2:831263/FULLTEXT01.pdf>. Acesso em: 01 dez 2017.

RABINER, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, Ieee, v. 77, n. 2, p. 257–286, 1989.

REYNOLDS, D. A.; ROSE, R. C. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE transactions on speech and audio processing*, IEEE, v. 3, n. 1, p. 72–83, 1995.

RODRIGUES, F. F. *Acionamento de um Robô Lego Mindstorms por Comandos Vocais Utilizando Redes Neurais Artificiais*. Dissertação (Monografia) — Universidade Federal de Ouro Preto, Engenharia de Controle e Automação, 08 2009.

SILVA, A. G. *Reconhecimento de Voz para Palavras Isoladas*. Dissertação (Monografia) — Universidade Federal de Pernambuco, Engenharia da Computaçao, Recife, 12 2009.

STAMP, M. A revealing introduction to hidden markov models. *Department of Computer Science San Jose State University*, 2015.

STUTTLE, M. N. *A Gaussian mixture model spectral representation for speech recognition*. Tese (Doutorado) — University of Cambridge, Engineering Department, 2003.

TEVAH, R. T. *Implementação de um sistema de reconhecimento de fala contínua com amplo vocabulário para o português brasileiro*. Dissertação (Dissertação de Mestrado) — COPPE - Universidade Federal do Rio de Janeiro, Engenharia Elétrica, Rio de Janeiro, 06 2006.

VYAS, M. A gaussian mixture model based speech recognition system using matlab. *Signal & Image Processing*, Academy & Industry Research Collaboration Center (AIRCC), v. 4, n. 4, p. 109, 2013.

YOON, B.-J. Hidden markov models and their applications in biological sequence analysis. *Current genomics*, Bentham Science Publishers, v. 10, n. 6, p. 402–415, 2009.