MARTHA VERONIKA KOZAKA

VINICIUS ITO IWASSO

SOFTWARE PARA RECONHECIMENTO DE EMOÇÕES NA VOZ USANDO SUPPORT VECTOR MACHINE

São Paulo

2011

MARTHA VERONIKA KOZAKA

VINICIUS ITO IWASSO

SOFTWARE PARA RECONHECIMENTO DE EMOÇÕES NA VOZ USANDO SUPPORT VECTOR MACHINE

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Bacharel em Engenharia Mecatrônica.

Área de concentração:
Engenharia Mecatrônica

Orientador:
Prof. Dr. Marcos Ribeiro Pereira Barretto

São Paulo

2011

FICHA CATALOGRÁFICA

Iwasso, Vinícius Ito
Software para reconhecimento de emoções na voz usando Support Vector Machine / V.I. Iwasso, M.V. Kozaka – São Paulo, 2011.
64 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos .

1. Reconhecimento de voz 2. Emoções 3. Interação homem-Máquina (Eficiência) 4. Softwares I. Kozaka, Martha Veronika II. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos III. t.

DEDICATÓRIA

Martha Veronika Kozaka

Dedico a todas às pessoas que me apoiaram para a realização desse trabalho: minha mãe principalmente, minha família e meu namorado.

Vinícius Ito Iwasso

Dedico esse trabalho aos meus pais pelo apoio e base que me foram dados, à minha namorada pela companhia e suporte em todos os momentos, e aos meus amigos.

AGRADECIMENTOS

Agradecemos ao Professor Marcos Ribeiro Pereira Barretto, pela orientação, pelo incentivo durante o trabalho, pela liberdade de pesquisa e pela oportunidade de nos trazer um tema tão atual e com perspectiva para o futuro.

“Chegar à perfeição não é viável, mas se buscarmos a perfeição, nós somos capazes de alcançar a excelência”

Vince Lombardi

RESUMO

O desenvolvimento de um software que possa distinguir as emoções apresentadas na voz é de grande importância, por apresentar diversas aplicações futuras. Um dos exemplos seria a melhora na interação entre computador e ser humano, no quesito comunicação. O projeto visa conseguir distinguir as quatro emoções básicas (triste, alegre, neutro e bravo) em amostras de vozes. O software é capaz de identificar falantes diferentes em uma amostra de áudio e indicar, para cada falante, qual a emoção passada. Foi aplicado um algoritmo baseado em Aprendizado de Sistemas, chamado SVM (Support Vector Machine), que é capaz de realizar a classificação de qualquer amostra, dentro dos seus limites.

Palavras-chave: reconhecimento de voz; emoções; interação homem-máquina (eficiência); softwares.

ABSTRACT

The development of a software able to recognize emotions in speech is important due to its possible future applications. An example would be the improvement in the interaction between human and machines, specifically in communication. The project's aim is to be able to recognize the four basic emotions (sad, happy, neutral and angry) in speech samples. The software will be able to identify different speakers in a sample and point out the emotion felt at that moment. It was applied an algorithm based on a Learning System, called SVM (Support Vector Machine) that classifies any sample within its limits.

Keywords: voice recognition, emotion, voice, man machine interaction (efficiency), software.

LISTAS DE ILUSTRAÇÕES

LISTAS DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de um software que possa distinguir as emoções apresentadas na voz de um indivíduo pode apresentar diversas aplicações. Um dos exemplos seria a melhora na interação entre computador e ser humano, no quesito comunicação.

Este trabalho é uma continuidade de um projeto iniciado anteriormente, onde alterações foram feitas para melhorar tanto a eficácia do software, como também aumentar as suas funcionalidades. Além de conseguir distinguir as quatro emoções básicas (triste, alegre, neutro e bravo) em amostras de vozes, o software será capaz de identificar diferentes locutores em uma mesma amostra e indicar a emoção de cada um. Para isso, é necessário diferenciar a voz humana dos outros sons presentes e realizar uma análise crítica e cautelosa em cima dos dados.

O funcionamento básico do software seria: analisar quais as propriedades do som pertinentes, tendo em vista as diferenças apresentadas em cada tipo de emoção. Com a aplicação de um algoritmo, baseado em Aprendizado de Sistemas, pode-se implementar uma amostra de vozes do Banco de Dados, para calibrar os pesos utilizados, priorizando assim as propriedades mais relevantes. As amostras que apresentarem maior similaridade com um tipo de emoção serão classificadas como tal.

O tema escolhido para o trabalho foi sugerido pelo Professor Doutor Marcos Ribeiro Pereira Barretto, cuja linha de pesquisa é desenvolvimento de humanóides. Este tema de estudo é recente e pode ser muito aprimorado e desenvolvido em estudos futuros.

O reconhecimento de emoção na fala atua com duas frentes da engenharia mecatrônica: programação e desenvolvimentos de robôs. Depende de inúmeros fatores, como contexto da conversa, volume, entonação da voz, gestos e expressões faciais. Essas características variam de pessoa para pessoa, portanto não se pode aplicar uma lógica generalizada para se realizar a classificação.

Cada vez mais a interação entre computador e ser humano está presente em nossas vidas. Buscando uma experiência mais real e confortável, uma das condições importantes para serem analisadas seria o reconhecimento das emoções das pessoas, bem como a capacidade de reproduzi-las. A partir desse ponto, as aplicações seriam inúmeras. Um exemplo seria para uma melhor comunicação entre um professor virtual e seus alunos.

Um tema que envolve não somente a área de engenharia mecatrônica. A detecção de emoções na fala pode ser útil em várias outras áreas. Comunicação entre pessoas com deficiência, atores podem checar a consistência de suas atuações, qualquer tipo de interatividade entre homem e meio de comunicação (poderosa ferramenta de vendas), possibilitar a identificação do estado emocional de um cliente de call-center e quem sabe poder identificar trotes.

1.1 OBJETIVOS

Esse trabalho é uma continuidade de outro trabalho “Desenvolvimento de Algoritmo de Reconhecimento de Emoções por Processamento da Voz” por Carlos Eduardo Pinheiro Corrêa e Lucas Bertan Menegassi.

O reconhecimento automatizado de emoções com o banco de dados utilizado no estudo não consegue atingir uma taxa de sucesso muito acima de 50% para as quatro emoções básicas (raiva, medo, felicidade e tristeza). Emoções naturais são mais complicadas de serem classificadas do que emoções simuladas.

A emoção é um fator muito importante na comunicação. Num texto lido sem emoção, não se pode transmitir a correta semântica do texto. Atualmente, busca-se desenvolver algoritmos eficientes para síntese e reconhecimento de emoções na fala. O desenvolvimento de interfaces homem-máquina é um grande campo para aplicações desse tipo de estudo.

Para que isso seja possível, um banco de dados de faixas com emoções é um pré-requisito. O uso de gravações com situações reais seria o melhor modo de se construir um banco de dados. Mas devido à grande dificuldade de se criar uma situação em que a pessoa se encontre num estado de emoção pura, que deveria ser gravado em condições especiais, para minimizar o nível de ruído, a utilização de atores interpretando textos com emoções simuladas são as mais utilizadas para a sua construção.

Os atores devem interpretar um mesmo texto seguindo emoções diferentes para que seja possível estabelecer uma base de comparação. As emoções escolhidas, que aparentam diferenças perceptíveis entre elas são: raiva, felicidade, tristeza e o neutro. Se forem escolhidas mais emoções, a análise de cada uma seria muito mais complexa. E algumas delas poderiam entrar em contradição ou serem muito parecidas.

Nesse trabalho é proposta uma solução para esse problema: algoritmos de aprendizado de sistemas. Tanto para Redes Neurais, SVM e Classificadores Bayesianos (que serão explicadas de forma mais detalhada posteriormente), o conceito básico seria de que de um banco de dados de vozes, uma amostra seria utilizada para que o software consiga “aprender” sobre as características principais dessas amostras e como que elas estão relacionadas na decisão de classificação. A outra parte do banco de dados de vozes seria testada e classificada por esse software já “calibrado”.

A porcentagem de acerto em reconhecimentos de emoções por um ser humano é de cerca de 80%. Um fato interessante de ressaltar seria de que em um sistema de reconhecimento de quatro emoções básicas, caso seja realizada uma classificação randômica de uma amostra, a porcentagem de acerto seria de aproximadamente 25 [3].

Em busca de uma melhor eficácia nos resultados de identificação de emoções, formam feitas as seguintes atividades: implementação um novo algoritmo de Aprendizado de Sistemas, verificação das características da voz que seriam mais

relevantes para a distinção entre emoções, ou seja, um estudo mais focado no tema além da complementação de algumas funcionalidades.

As etapas seguidas durante o estudo foram:

- Revisão da monografia anterior e bibliografia da mesma;
- Pesquisa de artigos relevantes para o tema e estudo de conceitos de Sistemas de Aprendizado;
- Aquisição de um novo banco de dados;
- Implementação do SVM no nosso software.
- Experimento Fatorial com os parâmetros do software, para analisarmos quais são as características da voz mais relevantes para distinguir emoções;
- Algoritmo capaz de Identificação dos falantes diferentes do outro.
- Integração dos Algoritmos, resultando no Software Final.

1.2 ESTRUTURA DO TRABALHO

O trabalho será apresentado começando sobre a revisão bibliográfica realizada. Serviram como base de estudo, uma seleção de artigos técnicos e capítulos de livros cujos assuntos eram compatíveis com o tema do trabalho e poderiam agregar mais conhecimento ao resultado final. Esses estudos trouxeram conhecimentos sobre como a voz humana pode ser analisada e como ela é afetada de acordo com as emoções sentidas pelo falante.

Após a apresentação do estudo sobre voz e emoção, será apresentado o método de aprendizado para máquinas que foi utilizado para identificação de padrões. O *Support Vector Machines* é um método utilizado atualmente que possui uma implementação

rápida e grande literatura. O SVM nativamente identifica dois padrões, mas com manipulação matemática é possível que sejam identificados mais de dois padrões.

O trabalho “Desenvolvimento de Algoritmo de Reconhecimento de Emoções por Processamento da Voz” realizado por Carlos Eduardo Pinheiro Corrêa e Lucas Bertan Menegassi foi o ponto de partida para esse trabalho. O algoritmo foi reimplementado e testado novamente.

Durante o trabalho foram realizados experimentos visando validar resultados e melhorar a taxa de sucesso do algoritmo. Primeiramente realizou-se um experimento com voluntários. Esses voluntários serviram de base para uma taxa de acerto considerada satisfatória. Após esse teste inicial, passou-se para a implementação de fato do algoritmo. Realizou-se um experimento fatorial para obtenção de parâmetros otimizados para a análise e também uma alteração para que fosse possível identificar falantes diferentes em amostras bem como suas respectivas emoções.

Por fim é apresentada a conclusão sobre o trabalho.

2 REVISÃO BIBLIOGRÁFICA

2.1 VOZ

O órgão principal envolvido na produção de voz é a laringe, mas para a formulação de fonemas, várias outras partes do corpo interagem entre si.

Como qualquer outro tipo de som, a produção de som necessita quatro componentes principais: fluxo de ar, oscilador, ressonador e amplificador. Os pulmões, abdômen, peito, costas, pernas e quadril são responsáveis pelo fluxo de ar, interagindo entre si, podem gerar um fluxo maior ou menor de ar, um fluxo com mais ou menos volume de ar, resultando em sons mais ou menos intensos. As cordas vocais funcionam como o oscilador, transformando o fluxo de ar gerado internamente em ondas mecânicas que constituem o som da voz de cada um. A voz possuirá um tom e um

timbre dependendo do ritmo de abertura, tamanho e tensão das cordas vocais. A laringe, faringe, cavidade bucal, cavidade nasal e a cabeça funcionam como a câmara de ressonância e amplificadores do som gerado pelas cordas vocais [12].

A articulação do som em palavras depende da interação coordenada da língua, palato, bochechas, lábios e dentes.

O corpo inteiro interfere na forma que a voz é produzida. Mudanças na postura, como alterações no centro de gravidade da pessoa, alteram a forma que os músculos irão se contrair.

A frequência fundamental média para os homens é de 120 Hz, 200 Hz para as mulheres e maiores ainda para crianças. Homens conseguem chegar a frequências abaixo de 100 Hz e acima de 600 Hz. As mulheres possuem uma faixa de 150 Hz até acima de 1500 Hz.

A prosódia é o estudo dos atributos presentes na fala. Ela engloba aquilo que não pode ser inferido, decifrado apenas pelo texto ou pela gramática, como o ritmo, entonação e ênfase ou acentuação dada pelo interlocutor durante a fala.

A prosódia pode refletir características do locutor, suas emoções, presente em todo diálogo, revelando se está sendo realizada uma afirmação, um comando ou uma pergunta. Erros de prosódia podem levar a erros de interpretação pelo receptor da mensagem. Para otimizar os resultados de análise por prosódia, deve-se criar módulos específicos para cada tipo de idioma. Os módulos não são necessariamente distintos, mas possuem características que precisam ser analisadas de maneiras específicas [1].

A prosódia na linguagem falada envolve a variação no tom, intensidade, comprimento de sílaba e frequências formantes na fala.

2.2 EMOÇÃO

Os estados de emoção estudados e analisados neste estudo foram: raiva, felicidade, tristeza e o estado neutro.

Estado de raiva: raiva é a forte emoção que você sente quando você pensa que alguém se comportou de uma forma injusta, cruel, ou inaceitável. O estado de raiva inclui o estado emocional, como tenso, preocupado, irritado, com medo, irritado e louco.

Estado de felicidade: A felicidade é um estado emocional ou afetivo caracterizado por sentimentos de alegria e satisfação. O estado de felicidade inclui o bem-estar, prazer, excitação, saúde, segurança e amor.

Estado de tristeza: tristeza é um estado de espírito exibir sentimentos de desvantagem e perda. O estado de tristeza inclui triste, miserável, deprimido, entediado e cansado.

Estado de neutralidade: O estado de neutralidade inclui sonolência, calma e relaxamento.

Existe um conceito conhecido que diz que o modo de falar de uma pessoa (conseqüentemente o sinal da voz resultante) funcionasse como uma impressão digital, ou seja, ela nunca pode ser reproduzida igualmente por outra pessoa. Esse conceito é que torna a atividade de reconhecimento de voz tão complicada. A voz é um sinal que apresenta infinitas informações. Antes de podermos realizar o aprendizado do sistema, temos que fazer um pré-processamento, que resultarão em parâmetros que representarão a voz em si [10].

Podemos definir os parâmetros em algumas classes:

Prosódia:

Parâmetros relacionados à prosódia são muito utilizados para a maioria dos sistemas de reconhecimento. A prosódia é capaz de diferenciar as emoções melhor que

qualquer outro tipo de parâmetro, por apresentar informações muito úteis. Ela pode ser representada com algumas características independentes entre si: freqüência fundamental, energia e duração, que podem gerar muitos outros coeficientes como valor mínimo, máximo, médio e entre outros. E essa é vista como uma desvantagem, pois ela é muito limitada. Em muitos trabalhos de pesquisa, foi constatado que utilizar somente parâmetros relacionados à Prosódia resulta em uma eficácia muito baixa.

Espectral:

Parâmetros relacionados à fonética são mais abrangentes em contraposição. Porém, as suas informações são mais limitadas na parte de diferenciação de emoções, ou seja, as informações contidas nesses parâmetros não são tão relevantes como os de prosódia.

Existem alguns métodos que são mais comumente utilizados:

MFCC (Mel-frequency Cepstrum Coefficients)

- LFPC (Log-frequency Power Coefficients)
- LPC (Linear Predictive Coefficients)
- PLP (Perceptual Linear Prediction)
- Qualidade da voz:
- Jitter (pitch modulation),
- Shimmer (amplitude modulation),
- Quantidade de silêncio na amostra normalizada,
- HNR (harmonics to noise ratio)

O que faremos nesse tópico será uma análise dos parâmetros atuais. Se eles são consistentes, ou seja, independentes do ambiente, equipamento utilizado para captura da amostra, o que está sendo dito e diferenças entre o estilo de fala das pessoas.

2.2.1 Parâmetros iniciais

a) Relacionados à frequência fundamental: Média (P1), Mediana (P2), Máximo (P3), Mínimo (P4), Fracional correspondente a 95% da amostra (P5), Fracional correspondente a 5% da amostra (P6), Amplitude (P3 –P4) (P7), Amplitude entre os fracionais 5%-95% (P5- P6) (P8), Desvio Padrão (P9).

b) Relacionados à intensidade: Média (P10), Mediana (P11), Máximo (P12), Fracional correspondente a 95% da amostra (P13), Fracional correspondente a 5% da amostra (P14), Amplitude entre os fracionais (P13-P14) (P15), Desvio Padrão (P16)

c) Relacionados à duração do som: Duração média de um segmento contínuo de voz (P17), Duração média das pausas (P18), Desvio padrão das pausas (P19), Razão voz/silencio (P20)

d) Relacionados à dinâmica da frequência fundamental: Incremento médio de um segmento contínuo de voz (P21), Incremento máximo de um segmento contínuo de voz (P22), Desvio padrão do incremento de F0UP (P23), Decremento médio de um segmento contínuo de voz (P24), Decremento máximo de um segmento contínuo de voz (P25), Desvio padrão do incremento de F0DOWN (P26), Razão entre incremento e decremento (P27)

e) Relacionadas à frequência fundamental normalizada: Média (P28), Mediana (P29), Desvio Padrão (P30)

f) Esforço Vocal (P31)

Desvantagem dos parâmetros iniciais

A frequência fundamental é um das informações mais importantes para caracterização da emoção na voz. Porém a sua utilização pode ser feita de diversas maneiras: pelo valor máximo, valor mínimo e muitas outras informações. Em todos os

casos, a frequencia utilizada é a normalizada, para que a frequencia da fala de cada usuário não interfira no resultado final.

Apesar de a intensidade ser um parâmetro relevante na identificação das emoções, pois é uma das características que mais mudam de acordo com a emoção sentida no momento (por exemplo: para casos em que se sente alegria e raiva, a amplitude é maior e para os casos em que se sente tristeza ou neutralidade, a amplitude é menor), ela pode mudar muito dependendo das condições ambientais e de como foi capturada essa amostra de voz. Além disso, ela pode mudar de acordo com o estilo de falar de uma pessoa para outra, principalmente de culturas diferentes (por exemplo, alemães falam com uma amplitude maior do que japoneses) [5].

No quesito “relacionado à duração do som”, alguns parâmetros, como tempo de continuidade de som e de silêncio, individualmente não trariam tantas informações relevantes para o processo. Um exemplo de parâmetro desse tema, que seria relevante, é o P(20), que mostra a razão entre eles e é um valor normalizado.

Modelo novo

Os parâmetros do estudo anterior foram baseados em prosódia e em qualidade da voz. Esse conceito foi mantido, mas alguns parâmetros foram desconsiderados no processo de identificação da emoção. Alguns parâmetros dessas mesmas áreas foram implementados e complementaram as informações necessárias para a classificação. O método utilizado foi o experimento fatorial [14]. Os parâmetros relevantes e escolhidos para a análise foram decididos empiricamente, por testes.

2.3 SVM

2.3.1 Teoria

O método SVM (*Support Vector Machine)* foi introduzido em 1992, por Boser, Guyon e Vapnik [8]. O SVM é um classificador binário, muito utilizado atualmente e apresenta diversas áreas de pesquisa ativas. É um método alternativo para as Redes Neurais.

Mas para se conseguir melhores resultados com SVM, é necessário um bom entendimento do seu funcionamento e dos critérios que influenciam na sua eficácia.

Inicialmente o SVM foi criado para somente classificar duas classes linearmente separáveis. Mas para lidar com casos que não podem ser classificados somente linearmente (casos mais comumente encontrados) são utilizadas as funções kernel. Assim, pode-se aplicar o classificador para casos não óbvios: como para o reconhecimento de emoções na voz, no nosso caso, como também para a área biológica para estudos sobre DNA, estrutura de proteínas e entre outros.

Vamos ver primeiramente como funciona o SVM como classificador linear e depois entender como ele pode ser transformado para um método não-linear utilizando funções de kernel. Outros pontos importantes são: como otimizar, como melhorar a eficácia e como os parâmetros de entrada podem influenciar no resultado do classificador. Veremos nesse tópico também alguns desafios que serão encontrados na nossa implementação e uma explicação de como iremos resolvê-los: como trabalhar com classes desbalanceadas e como trabalhar com casos com mais de duas classes de classificação?

2.3.2 Classificadores Lineares

SVM é um exemplo de um classificador linear de duas classes. As classes serão indicadas por +1 (exemplos positivos) e -1 (exemplos negativos).

Definiremos como:

- **x** é o vetor com todos os componentes x_i, que são chamados de padrões (*patterns*) ou exemplos (*examples*).
- Seja $x_i = \{x_1, x_2, ..., x_n\}$, sendo n o número de componentes do conjunto de dados.
- Seja $y_i = \{-1, +1\}$ o indicador respectivo de x_i (classificação da instância).

O classificador linear dependerá da seguinte função:

$$f(x) = \boldsymbol{w}^T \boldsymbol{x} + b \quad (1)$$

Sendo que **w** é conhecido como vetor de peso (*weight vector*) e b é chamado de *bias*. Vamos primeiramente considerar que o b é nulo. O conjunto de pontos **x** tal que $\boldsymbol{w}^T \boldsymbol{x} = \boldsymbol{0}$ são todos os pontos perpendiculares a **w** e passam pela origem – uma linha em um plano de duas dimensões, um plano em três dimensões, e de uma forma mais generalizada, um hiperplano. O bias b translada o hiperplano da origem.

O hiperplano {x: $f(x) = \boldsymbol{w}^T \boldsymbol{x} + b = 0$} divide o espaço em dois: o sinal da função f denotará qual o lado do hiperplano o ponto estará (positivo ou negativo). Este limite que determina a classificação é clamado de limite de decisão. Um classificador com um limite de decisão linear é denominado linear. Um classificador com um limite de decisão não-linear é um denominado como não-linear.

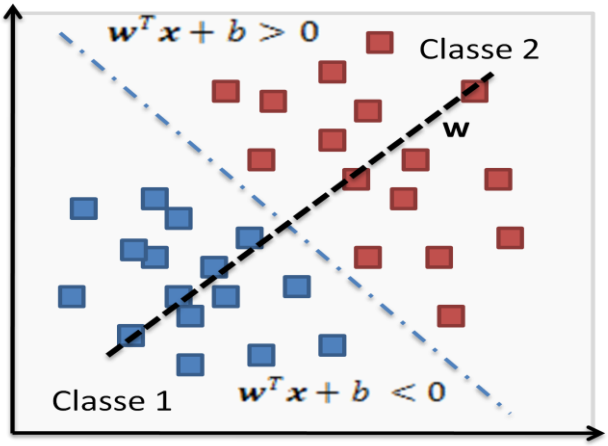


Figura 1: Representação de Classificador Liner SVM

2.3.3 Kernels: De linear para classificadores não-lineares.

Na maior parte das aplicações um classificador não-linear oferece uma melhor acurácia. Mas os classificadores lineares apresentam um algoritmo de treinamento mais simples. Então, uma forma simples de resolver esse problema seria de fazer uma

transformação do espaço de entrada X para um espaço de características F, usando uma função não-linear $\phi: X \rightarrow F$. E nesse espaço F, a função do classificador será:

$$f(x) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + b \quad (2)$$

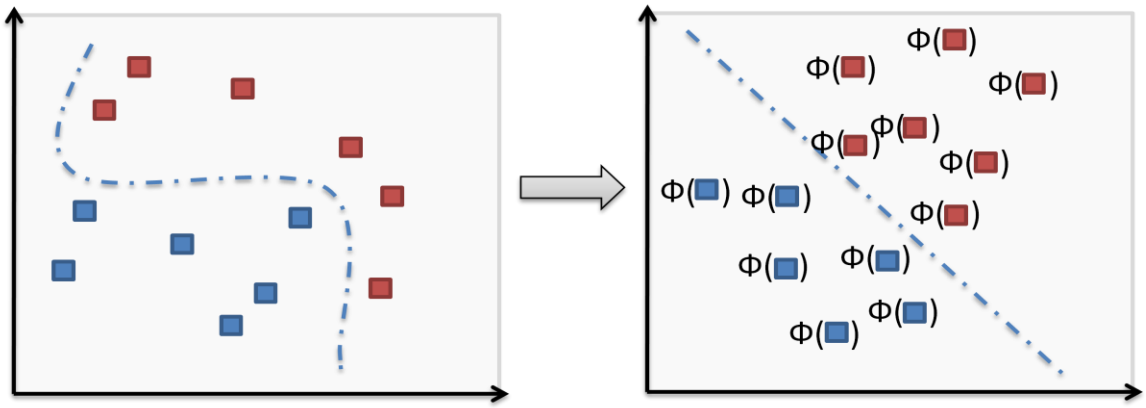


Figura 2: Transformação do espaço de entrada para espaço de características

Essa abordagem pode ser utilizada para casos pequenos sem problemas. Mas quando se tem casos um pouco mais complexos, pode-se ter um aumento inaceitável do uso de memória utilizada para armazenamento durante a classificação e do tempo necessário de processamento, além disso, é difícil de obter uma boa estimação.

Os métodos de kernel resolvem este problema. Suponha que o vetor de peso pode ser expresso pelos exemplos utilizados para o aprendizado:

$$f(x) = \sum_{i=1}^{n} \alpha i . xi^T x + b \quad (3)$$

No espaço de características F fica:

$$f(x) = \sum_{i=1}^{n} \alpha i . \phi(xi)^T \phi(x) + b \quad (4)$$

Esta representação em função de αi é conhecida como dual. Ainda nessa representação, para casos com um número grande de exemplos, é muito difícil de ser implementado. Para isso, é utilizado o chamado Truque Kernel.

Define-se a relação entre a função kernel K e o mapeamento ϕ é:

$$k(x, x') = \phi(x)^T \phi(x') \quad (5)$$

A função f fica:

$$f(x) = \sum_{i=1}^{n} \propto i.k(x, xi) + b \quad (6)$$

Para não termos um custo computacional tão grande, podemos calcular a função kernel para alguns dos casos mais utilizados, com uma formulação mais simplificada.

Para kernel polinomial com grau d, podemos calcular da seguinte maneira:

$$k(x, x') = (x^T x' + 1)^d \quad (7)$$

Todos os polinômios devem ter grau unitário (ou seja, linear) até o valor de d. Caso d seja igual a 1, na equação anterior normalmente a constante adicionada (1) é omitida.

Para kernel gaussiana, a função de kernel é definida como:

$$k(x, x') = \exp(-\gamma. \|x - x'\|^2) \quad (8)$$

Onde $\gamma > 0$ é um parâmetro que controla a largura da Gaussiana.

2.3.4 Margens de Classificação Maximizadas

Definição de margem:

Para um hiperplano, nos denotamos como x_+ o ponto mais próximo do hiperplano dentre os exemplos positivos. Para os exemplos negativos denotamos este ponto como x_-.

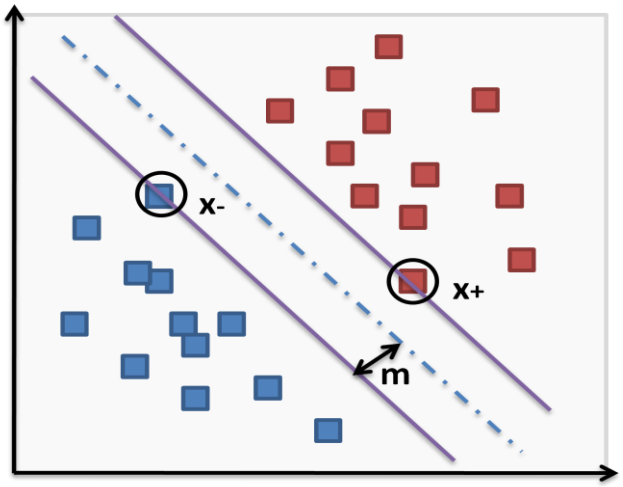


Figura 3: Representação de Margem e dos pontos x+ e x_

A margem do hiperplano f seria:

$$m\,(f) = \frac{1}{2}.\,\widehat{\boldsymbol{w}}^{T}(x_{+} - x_{-}) \quad (9)$$

Sendo que $\widehat{\boldsymbol{w}}$ é o vetor unitário na direção de **w.**

Estamos assumindo que x_+ e x_- são eqüidistantes do limite de decisão, ou seja:

$$f(x_{+}) = \boldsymbol{w}^{T}x_{+} + b = a \quad (10)$$

$$f(x_{-}) = \boldsymbol{w}^{T}x_{-} + b = -a \quad (11)$$

Para uma constante a > 0. Somando as equações anteriores e dividindo por $\|\boldsymbol{w}\|$, temos que:

$$m\,(f) = \frac{1}{2}.\,\widehat{\boldsymbol{w}}^{T}(x_{+} - x_{-}) = \frac{1}{\|\boldsymbol{w}\|} \quad (12)$$

Classificador com margens maximizadas

Para maximizar a margem do classificador é preciso maximizar $1/\|\boldsymbol{w}\|$, que é equivalente a minimizar $\|\boldsymbol{w}\|^2$. O limite da decisão deve classificar todos os pontos corretamente. O problema de otimização se torna:

Minimizar $\frac{1}{2}.\|\boldsymbol{w}\|^2$ (13)

Sujeito a $yi(\boldsymbol{w}^T x_i + b) \geq 1$ (14)

Na prática, muitas das informações não podem ser classificadas linearmente. Para isso utilizamos o método chamado SVM Margem Suave (*Soft-Margin SVM)*. Pode-se atingir um resultado satisfatório, com uma margem maior, para casos em que algumas dados não sejam classificados corretamente, ou seja, casos que apresentem uma pequena porcentagem de erro.

$$yi(\boldsymbol{w}^T x_i + b) \geq 1 - \varepsilon i \quad (15)$$

Onde $\varepsilon i \geq 0$ são variáveis de fola (*slack variables*) que permite que alguns exemplos se encontrem na margem ($0 \leq \varepsilon i \leq 1$) ou sejam classificados erroneamente ($\varepsilon i < 1$).

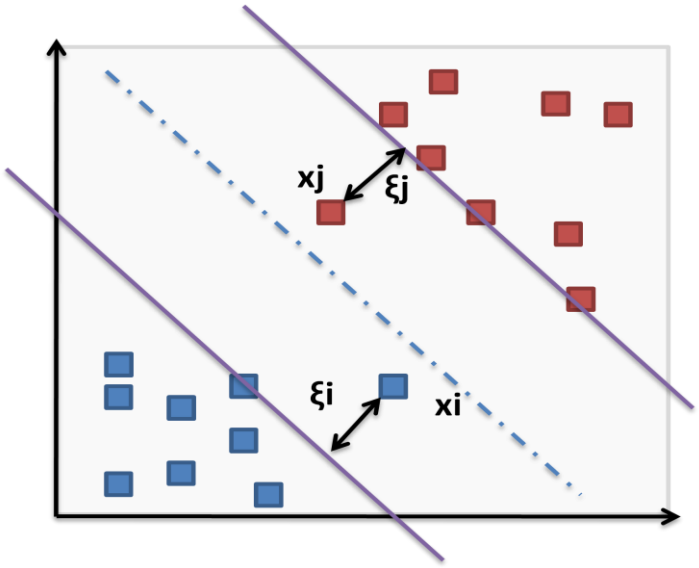


Figura 4: Exemplos de erros de margem

Nesse caso, o problema de otimização fica:

Minimizar $\frac{1}{2}.\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\varepsilon i$ (16)

Sujeito a $yi(\boldsymbol{w}^T x_i + b) \geq 1 - \varepsilon i,\ \varepsilon i \geq 0$ (17)

Onde C > 0 representa a importância de se maximizar a margem relativamente à minimização do erro.

Existe também o método de Multiplicadores de Lagrange, em que podemos transformar o problema para a representação dual.

Maximizar $\sum_{i=1}^{n} \propto i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} yi.yj. \propto i. \propto j.xi^T xj$ (18)

Sujeito a $\propto i \geq 0;\ \sum_{i=1}^{n} \propto i\,.yi = 0$ (19)

Onde $\boldsymbol{w} = \sum_{i=1}^{n} \propto i.yi.xi$ (20)

2.3.5 Entendendo os Efeitos dos Parâmetros de SVM e Kernel

Pelo algoritmo de treinamento/aprendizado do sistema, os parâmetros resultantes são os $\propto i\ e\ b$. Mas para isso, como vimos anteriormente, muitos parâmetro precisam ser tratados para obtermos um bom classificador. A constante de margem suave C e quaisquer outros parâmetros que as funções de kernel podem depender (isso varia conforme modelo).

Primeiramente, analisando C, vemos que para um maior C, teremos uma maior quantidade de erros.

Para maiores valores tanto do grau d, para funções polinomiais e da largura γ, para funções gaussianas, temos uma maior flexibilidade de classificação.

2.3.6 Casos em que temos multi-classes

Os casos que apresentam o problema de classificação múltipa de classes são aqueles em que o conjunto de dados para aprendizado pode pertencer de N diferentes classes, sendo N maior do que dois. E o objetivo que precisa ser atingido é que

qualquer caso depois a ser estudado pelo sistema deve classificar corretamente dentre as N classes.

Existem alguns métodos para esses casos. Existem métodos em que o SVM é transformado para realmente classificar dentre todas as classes desejadas, mas eles apresentam um custo computacional muito alto e é muito complexo. Outro método seria transformar um classificador de duas classes em um classificador de múltiplas classes, método chamado de "*one-vs-all*" (OVA). A vantagem é que ele é mais simples, tanto para a fase de treinamento como de custo computacional. Existem teorias em que OVA apresenta tanta eficácia quanto os outros métodos. Será esse método a ser utilizado pelo nosso projeto.

One-vs-all

Consiste em classificar para uma classe específica contra todas as outras e repetir esse procedimento, fazendo com que todas as N classes sejam estudadas individualmente [7]. Por exemplo: classificar a emoção "triste" em contraste com "alegre", "bravo" e "neutro", e esses três últimos serão considerados como uma mesma classe de classificação. Depois realizar o mesmo procedimento estudando individualmente as classes "alegre", "bravo" e "neutro".

Quando formos estudar um caso a parte do nosso banco de dados no nosso sistema já após o aprendizado, o classificador (alegre, triste, neutro , bravo) que resultar no maior valor (positivo), ou seja, que apresentar maiores assimilações com o caso estudado, será classificado respectivamente como tal.

2.4 TRABALHOS CORRELACIONADOS

2.4.1 Monografia Anterior

No início do trabalho, foi considerado o estudo realizado anteriormente pelos alunos Carlos Eduardo Pinheiro Corrêa e Lucas Bertan Menegassi. Os componentes obtidos

do trabalho foram: um software, uma monografia sobre o mesmo e um banco de dados de vozes.

Estudo da monografia e entendimento do software:

O software foi implementado no Matlab e não apresenta nenhum tipo de interface com o usuário, tendo que ser utilizado diretamente pelo Matlab.

O algoritmo desenvolvido anteriormente basicamente usava um método de aprendizado que usava uma série de amostras para estabelecer padrões para cada uma das quatro emoções básicas: alegre, bravo, neutro e triste.

Banco de Dados de Vozes

O banco de dados contêm um total de 86 faixas em formato *.wav com duração média de 30 segundos. Para alcançar um aprendizado eficiente, uma boa base de dados é necessária para que um padrão seja estabelecido com consistência.

Nesse caso, o banco de dados utilizado foi analisado e percebeu uma má qualidade de gravação o que pode ter comprometido os resultados obtidos. O nível de ruído apresentado foi muito elevado, devido à falta de equipamento para obtenção desse banco de dados. Uma possível solução seria a aplicação de filtros digitais nas amostras.

O banco foi obtido com a ajuda dos atores do GTP – Grupo de Teatro da Poli, interpretando da maneira que lhe foi conveniente, as quatro emoções, lendo textos que lhe foram disponibilizados [1].

Implementação do software e teste:

Uma primeira dificuldade que foi vista na implementação do software foi que para a fase de aprendizado do sistema, a chamada das amostras de vozes teria que ser feito manualmente, tendo que se chamar cada amostra e implementar a função de aprendizado individualmente, gastando-se muito tempo de trabalho. A solução proposta

para isso foi de se automatizar esse procedimento. Essa função criada se encontra detalhadamente no *Anexo A*.

O algoritmo por eles desenvolvido foi testado com o mesmo banco de dados utilizado durante o estudo e os resultados foram semelhantes.

Caso 1: Todo o banco de dados utilizado tanto para aprendizado como para teste. A eficácia global descrita na monografai foi de 62,5%. Pode-se analisar os resultados por emoção na Tabela 1.

Tabela 1: Resultado do software da monografia anterior para o Caso 1

	Resultado Obtido				
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**90,9%**	4,5%	5%	4,5%
	Alegre	25%	**40%**	0%	40%
	Triste	60%	0%	**5%**	405
	Bravo	5%	15%	0%	**85%**
Eficácia	**56,25%**				

Tabela 2: Resultado da Implementação para o Caso 1

	Resultado Obtido				
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**90,4%**	4,8%	4,8%	0
	Alegre	24%	**38%**	0	38%
	Triste	57%	0	**5%**	38%
	Bravo	5%	14%	0	**86%**
Eficácia	**54,975%**				

Caso 2: Uma metade é utilizada para aprendizado e outra para teste. Para acharmos o rendimento médio, dividiremos o caso 2 em duas partes. A primeira usa uma metade para o treinamento e a outra metade para teste e a segunda parte é a situação inversa.

Tabela 3: Resultado do software da monografia para o Caso 2 Primeira Parte

	Resultado Obtido				
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**36,4%**	18,2%%	45,5%	0%
	Alegre	9,1%	**36,4%**	9,1%	45,5%
	Triste	9,1%	18,2%	**36,4%**	36,4%
	Bravo	0%	36,4%	9,1%	**54,5%**
Eficácia	**40,9%**				

Tabela 4: Resultado do software da monografia para o Caso 2 Segunda Parte

	Resultado Obtido				
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**63,6%**	0%	18,2%	18,2%
	Alegre	9,1%	**18,2%**	27,3%	45,5%
	Triste	54,5%	0%	**36,4%**	9,1%
	Bravo	0%	9,1%	18,2%	**72,7%**
Eficácia	**47,7%**				

Eficácia Média resultou em **44,3%.**

Tabela 5: Resultado da Implementação do Caso 2 Primeira Parte

	Resultado Obtido				
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**80%**	10%	10%	0
	Alegre	20%	**40%**	0	40%
	Triste	70%	10%	**0%**	20%
	Bravo	10%	30%	0	**60%**
Eficácia	**45%**				

Tabela 6: Resultado da implementaçào do Caso 2 Segunda Parte

	Resultado Obtido				
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**80%**	10%	10%	0
	Alegre	10%	**30%**	10	50%
	Triste	30%	40%	**20%**	10%
	Bravo	10%	30%	0	**60%**
Eficácia	**47,5%**				

Eficácia Média resultou em **46,25%.**

Este teste foi feito tanto para verificar os resultados do trabalho anterior como também para nos certificar que o banco de dados de vozes estava sendo suficiente para implementações futuras.

3 EXPERIMENTOS

3.1 CORPUS (BANCO DE DADOS)

Após uma primeira análise do banco de dados utilizado no trabalho anterior, que continha um nível de ruído elevado o que dificultava até mesmo um reconhecimento feito por voluntários humanos, foi decidido que seria necessário um novo banco de dados para a realização do estudo. O banco de dados utilizado nesse estudo foi adquirido de um site da Surrey Audio-Visual Experessed Emotion, um braço da Universidade de Surrey no Reino Unido [13]. O banco de dados original consistia em faixas gravadas por quatro atores nativos na língua inglesa em sete diferentes emoções: raiva, nojo, medo, alegria, tristeza, surpresa e neutro. Os textos que foram interpretados pelos atores simulando todas as emoções consistiam em 15 sentenças foneticamente balanceadas. As faixas foram gravadas no formato wav com freqüência de amostragem de 44100 Hz.

Para este trabalho, utilizaram-se apenas as faixas cujas emoções fossem: raiva, alegria, tristeza e neutro.

O banco de dados utilizado no trabalho anterior possuía uma alta relação de ruído, o que pode ter comprometido os resultados obtidos. O nível de ruído apresentado foi muito elevado, devido à falta de equipamento para obtenção desse banco de dados. Uma possível solução seria a aplicação de filtros digitais nas amostras. Tal alternativa não foi necessária já que foi adquirido um novo corpus de melhor qualidade.

3.2 EXPERIMENTO 0

Análise com voluntários humanos: Buscando avaliar de modo quantitativo o banco de dados utilizado no estudo, realizou-se um teste para se determinar um valor de comparação para o qual o sucesso do algoritmo deveria se basear.

Utilizou-se durante o teste, a ajuda de 3 voluntários para testar as 180 amostras. Cada voluntário foi submetido ao teste de um terço do banco de dados utilizado no estudo, sendo que as amostras foram dividas igualmente de acordo com as emoções. Cada usuário recebeu uma parcela do banco de dados com falas nas 4 emoções dos 3 falantes do banco de dados.

Tabela 7: Resultado Voluntário 1

		Resultado Obtido			
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**60%**	0%	27%	13%
	Alegre	13%	**47%**	33%	7%
	Triste	7%	0%	**86%**	7%
	Bravo	14%	7%	14%	**65%**
Eficácia		**65%**			

Tabela 8: Resultado Voluntário 2

		Resultado Obtido			
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**84%**	0%	7%	7%
	Alegre	7%	**7%**	28%	58%
	Triste	7%	0%	**86%**	7%
	Bravo	14%	7%	14%	**65%**
Eficácia		**60%**			

Tabela 9: Resultado Voluntáio 3

		Resultado Obtido			
Resultado		Neutro	Alegre	Triste	Bravo

Desejado	Neutro	**60%**	0%	27%	13%
	Alegre	13%	**47%**	33%	7%
	Triste	33%	0%	**60%**	7%
	Bravo	0%	14%	0%	**86%**
Eficácia	**78,3%**				

No geral, os voluntários, sem nenhum tipo de influência obtiveram um índice de acerto de 67,8%.

3.3 EXPERIMENTO 1

A plataforma utilizada no trabalho foi o programa Matlab®. O Matlab possui uma biblioteca de funções que implementam o SVM para diversas classes.

O primeiro experimento foi analisar a eficácia do SVM para reconhecimento das emoções através da análise dos 30 parâmetros levantados pelo trabalho Desenvolvimento de Algoritmo de Reconhecimento de Emoções por Processamento da Voz [1].

O código da implementação do SVM se encontra no **Anexo B.**

Com base no resultados de alguns testes de implementação do software, utilizando função kernel gaussiana e polinomial, percebe-se que o mais apropriado, para esse caso, é o polinomial.

Além de escolher a função kernel, tem-se que entender qual a influência dos parâmetros dessa função kernel na porcentagem de acerto. Para isso é necessário realizar diversos testes, com diferentes valores de parâmetros kernel. Os parâmetros que resultarem em uma melhor eficácia serão os apropriados para o programa.

Após tentativas empíricas utilizando Funções kernel Gaussianas e Polinomiais, a implementação que obteve mais sucesso foi a que utilizou os seguinte parâmetros de entrada:

```
c = 1000;
lambda = 1e-7;
kerneloption= 1;
verbose = 0;
```

C e lambda são parâmetros utilizados para solucionar o problema de otimização de margem (utilizando Multiplicadores de Lagrange). Kerneloption é um parâmetro que depende do tipo de Função de Kernel a ser utilizada. No caso em que a função de kernel é polinomial, o kerneloption se refere ao grau do polinômio. Verbose é um parâmetro utilizada na biblioteca de SVM, que mostra ou não os resultados no matlab. Colocando-o como zero nenhum display é realizado,

Foram analisados dois casos diferentes neste experimento;

3.3.1 Caso 1: Banco de dados em português

No primeiro caso, visando confirmar a qualidade do banco de dados usado no trabalho anterior, foi utilizado 100% das amostras do mesmo para o aprendizado. Essas amostras foram utilizadas para verificar a eficácia do software e banco de dados. Os resultados obtidos foram os seguintes:

Tabela 10: Resultado caso 1 experimento 1

		Resultado Obtido			
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**66,7%**	19,0%	9,5%	4,8%
	Alegre	0,0%	**95,2%**	0,0%	4,8%
	Triste	0,0%	0,0%	**85,7%**	14,3%
	Bravo	4,8%	4,8%	14,2%	**76,2%**

Eficácia	**80,1%**

A eficácia obtida utilizando SVM foi 42% acima do que a encontrada no trabalho anterior, onde foi usado, para identificação e classificação de emoções, redes neurais.

3.3.2 Caso 2: Banco de dados em inglês

Motivado pelo resultado obtido com o banco de dados em português, foi realizada a mesma metodologia com o banco de dados em inglês. E os resultados foram os seguintes:

Tabela 11: Resultado caso 2

		Resultado Obtido			
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**73,3%**	17,7%	4,5%	4,5%
	Alegre	4,4%	**95,6%**	0,0%	0,0%
	Triste	2,2%	0,0%	**80,0%**	17,8%
	Bravo	4,4%	0,0%	40,0%	**55,6%**
Eficácia		**76,1%**			

A eficácia no caso 2 foi acima da obtida no experimento 0, com voluntários humanos e apenas 5% menor do que a obtida no caso 1.

3.4 EXPERIMENTO 2: ANÁLISE CRUZADA

Com resultados positivos no experimento 1, foi realizado uma análise com banco de dados cruzados. Usando o banco de dados em português para o aprendizado e o banco de dados em inglês para o teste, e vice-versa.

A eficácia obtida foi bem abaixo do esperado, indicando que o reconhecimento de emoções depende do idioma que ser quer analisar.

Tabela 12: Resultado caso 3 - Português / Inglês

		Resultado Obtido			
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**55,6%**	8,9%	22,2%	13,3%
	Alegre	53,3%	**20,0%**	13,3%	13,3%
	Triste	40,0%	0,0%	**60,0%**	0,0%
	Bravo	33,3%	4,4%	62,2%	**0,0%**
Eficácia		**33,9%**			

Tabela 13: Resultado caso 3 - Inglês / Português

		Resultado Obtido			
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**85,7%**	4,8%	4,5%	4,5%
	Alegre	95,2%	**4,8%**	0,0%	0,0%
	Triste	90,5%	0,0%	**4,8%**	4,8%
	Bravo	95,2%	4,8%	0,0%	**0,0%**
Eficácia		**23,8%**			

3.5 EXPERIMENTO 3: EXPERIMENTO FATORIAL

Para melhorar a eficácia média do software, fez-se um experimento fatorial para todos os parâmetros relevantes para a função kernel. Esse experimento fatorial consiste em um código que percorre diversas possibilidades de parâmetros de entrada e reconhece quais seriam as condições que obtêm melhor resultado em termos de eficiência.

Esse experimento fatorial foi feito para ambos os bancos de dados e obtiveram condições ótimas diferentes entre si.

3.5.1 Caso 1: Banco de dados em português

No primeiro caso, o algoritmo analisou os parâmetros de entrada da função de treinamento. Os valores ótimos para os parâmetros de entrada encontrados foram os seguintes:

```
c = 1018;
lambda = 1e-7;
kerneloption= 1;
verbose = 0;
```

Utilizando os 31 parâmetros para análise, foram obtidos os seguintes resultados para o caso em que se faz treinamento com 100% das amostras e teste com 100% das amostras:

Tabela 14: Resultado Caso 1 com 100% e 100%– Experimento 3

	Resultado Obtido				
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**66,7%**	19,0%	9,5%	4,8%
	Alegre	0,0%	**95,2%**	0,0%	4,8%
	Triste	0,0%	0,0%	**90,5%**	9,5%
	Bravo	4,8%	4,8%	14,2%	**76,2%**
Eficácia	**82,1%**				

A eficácia obtida utilizando SVM foi 2,5% acima do que a encontrada no experimento 1.

E para o caso em que se faz treinamento com 50% das amostras e testes com os outros 50% das amostras:

Tabela 15: Resultado Caso 1 com 50% e 50% - Experimento 3

		Resultado Obtido			
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**100,0%**	0,0%	0,0%	0,0%
	Alegre	13,6%	**36,4%**	31,8%	18,2%
	Triste	0,0%	13,9%	**68,2%**	18,2%
	Bravo	0,0%	27,3%	45,5%	**27,3%**
Eficácia		**58,0%**			

A partir dessa etapa, o experimento fatorial foi realizado de maneira a encontrar a melhor combinação dentre os 31 parâmetros. Devido ao número de cálculos que seriam necessários para analisar todos os casos, algo na casa das centenas de milhões, apenas os casos com 1, 2, 3, 27, 28, 29, 30 e 31 parâmetros foram analisados nesse estudo.

Tabela 16: Resultado experimento fatorial no banco de dados em português

Número de parâmetros	Máximo número de acertos
1	46,43%
2	53,57%
3	58,33%
27	84,52%
28	83,33%
29	83,33%
30	82,15%
31	82,15%

O resultado ótimo foi encontrado utilizando os parâmetros: 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30 e 31. Ou seja, sem os parâmetros 1 a 4.

3.5.2 Caso 2: Banco de dados em inglês

Como no primeiro caso, o algoritmo analisou os parâmetros de entrada da função de treinamento. Os valores ótimos para os parâmetros de entrada encontrados foram os seguintes:

```
c = 413;
lambda = 1e-7;
kerneloption= 1;
verbose = 0;
```

Foram obtidos os seguintes resultados para o caso em que se faz treinamento com 100% das amostras e teste com 100% das amostras:

Tabela 17: Resultado Caso 2 com 100% e 100% – Experimento 3

		Resultado Obtido			
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**71,1%**	17,7%	4,5%	6,7%
	Alegre	4,4%	**95,6%**	0,0%	0,0%
	Triste	2,2%	0,0%	**82,2%**	15,6%
	Bravo	2,2%	0,0%	33,3%	**64,5%**
Eficácia		**77,8%**			

A eficácia obtida utilizando SVM foi 3% acima do que a encontrada no experimento 1.

Foram obtidos os seguintes resultados para o caso em que se faz treinamento com 50% das amostras e testes com os outros 50% das amostras:

Tabela 18: Resultado Caso 2 com 50% e 50% - Experimento 3

		Resultado Obtido			
Resultado Desejado		Neutro	Alegre	Triste	Bravo
	Neutro	**100,0%**	0,0%	0,0%	0,0%
	Alegre	0,0%	**93,5%**	4,4%	2,2%
	Triste	0,0%	13,5%	**40,0%**	46,4%
	Bravo	0,0%	15,8%	40,0%	**33,1%**
Eficácia		**66,7%**			

A partir dessa etapa, o experimento fatorial foi realizado de maneira a encontrar a melhor combinação dentre os 31 parâmetros. Devido ao número de cálculos que seriam necessários para analisar todos os casos, algo na casa das centenas de milhões, apenas os casos com 1, 2, 3, 27, 28, 29, 30 e 31 parâmetros foram analisados nesse estudo.

Tabela 19: Resultados do experiment0 fatorial no banco de dados em inglês

Número de parâmetros	Máximo número de acertos
1	50,00%
2	54,44%
3	56,38%
27	78,89%
28	78,88%
29	77,78%
30	77,78%
31	77,78%

O resultado ótimo foi encontrado utilizando os parâmetros: 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30 e 31. Ou seja, sem os parâmetros 1 a 4. Coincidentemente, os mesmos parâmetros encontrados para o banco de dados em português.

3.6 EXPERIMENTO 4: IDENTIFICAÇÃO DO NÚMERO DE FALANTES EM UMA AMOSTRA

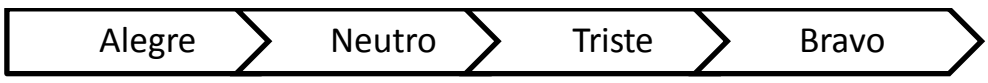
O experimento 2 foi utilizado para conseguir identificar os diferentes falantes dentro de uma amostra. O algoritmo se encontra no **Anexo C.**

A identificação do falante pode ser feita de diferentes maneiras. Através da identificação por freqüência fundamental da voz, através da determinação do momento em que um falante para de falar e o outro começa.

A criação de uma identidade vocal para um indivíduo é algo que vai além do escopo deste trabalho. A abordagem utilizada foi a segunda. Para identificar o momento em que um falante termina sua frase para o outro começar a falar, foram tomadas algumas premissas: um falante só começa a falar após o outro terminar. Ou seja, não foi considerada situação de briga ou discussão com pessoas falando ao mesmo tempo.

A determinação do intervalo de tempo necessário para poder se afirmar que houve uma troca de falante da amostra foi feita através da análise de um período de silêncio. O período utilizado foi de 0,5 segundos. Se durante a amostra ocorrer um intervalo de silêncio maio do que 0,5 segundos, o algoritmo entende que houve uma troca de falante.

A determinação da emoção do pedaço de cada amostra é feito, partindo-se da reconstrução do vetor data de cada falante da amostra.



Apesar de se conseguir identificar bem os falantes, ao se tentar analisar qual é a emoção transmitida em cada trecho, a eficácia do software se torna um pouco comprometida. Pela nossa análise, pode-se supor que se deve ao fato de a lógica do algoritmo estar desconsiderando os períodos de silêncio da amostra, o que compromete a sua classificação por SVM. Os próximos passos desse trabalho seriam de aprofudamento nesse campo de pesquisa.

4 CONCLUSÃO

Ao analisar os resultados dos experimentos realizados, pode-se chegar à conclusão de que o *Support Vector Machine* é um classificador aplicável na identificação de emoções. As taxas de acerto usando SVM foram até 42% acima das obtidas no trabalho anterior.

Um aspecto que ficou evidente e comprovado no experimento 2, foi o de que o aprendizado e acurácia do software depende do idioma que se está estudando. O banco de dados usado para o aprendizado deve ser do mesmo idioma que se quer realizar o reconhecimento. Caso contrário, as taxas de acerto são baixas, aproximadamente 25%.

O experimento piloto realizado nesse trabalho mostra que é uma aplicação possível e pode melhorar ainda mais a relação homem-máquina, com um tratamento customizado.

Para trabalhos futuros pode-se sugerir que se analisem ainda mais as combinações ótimas dos parâmetros levantados nas amostras de áudio e que faça um estudo direcionado para estabelecer como o corpus usado no banco de dados interfere no resultado final do reconhecimento de emoções na voz.

5 BIBLIOGRAFIA

[1] CORRÊA, C.; MENEGASSI, L., Desenvolvimento de Algorítmo de Reconhecimento de Emoções por Processamento da Voz, 2008.

[2] EYBEN, F.; WÖLLMER, M.; SCHULLER, B. openEAR - Introducing the Munich Open-Source Emotion and Affect Recognition Toolkit, Technische Universität München, Institute for Human-Machine Communication, 2009.

[3] VERVERIDIS, D.; KOTROPOULOS, C. A State of the Art Review on Emotional Speech Databases, Artificial Intelligence & Information Analysis Laboratory, Department of Informatics, Aristotle University of Thessaloniki, 2003

[4] BURKHARDT, F.; PAESCHKE, A.; ROLFES, M.; SENDLMEIER, W.; WEISS, B. A Database of German Emotional Speech, T-Systems , 2005.

[5] ROH, Y.W. et al. Novel acoustic features for speech emotion recognition, School of Information and Communication Engineering, Sungkyunkwan University, 2009.

[6] GRIMALDI, M.; CUMMINS, F., Speaker Identification Using Instantaneous Frequencies, IEEE Transaction On Audio, Speech, And Language Processing, v.16, n.6, August 2008.

[7] RIFKIN, R.; KLAUTAU, A., In Defense of One-Vs-All Classification, Journal of Machine Learning Research 5, p. 101-141, 2004.

[8] BEN-HUR, A.; WESTON, J., A User's Guide to Support Vector Machines, Department of Computer Science, Colorado State University, 2010.

[9] PROVOST, F., Machine Learning form Imbalanced Data Sets 101, New York University, 2000.

[10] SCHERER, K.; BÄNZIGER, T; ROESCH, E., Blueprinting for Affective Computing – A sourcebook, Oxford University Press, p. 232-244, 2010.

[11] VOGT, T.; ANDRÉ, E.; BEE, N., EmoVoice – A Framework for Online Recognition of Emotions from Voice, Multimedia Concepts and their Application, University of Augburg, 2008.

[12] HEMAN-ACKAH, Y. D., Physiology of Voice Production: Considerations for the Vocal Performer, Department of Otolaryngology – Head and Neck Sugery, Thomas Jefferson University, 2005.

[13] Site da Surrey Audio-Visual Experessed Emotion (SAVEE) Database. Disponível em: < http://personal.ee.surrey.ac.uk/Personal/P.Jackson/SAVEE/>. Acesso em: junho/2011.

[14] MONTGOMERY, D. C., RUNGER, G. C., Applied Statistics and Probability for Engineers, John wiley & Sons, Inc., p.602-616, 2003.

6 ANEXOS

6.1 ANEXO A

Função importaRandom

```
function [Md,Mdlinha] = importaRandom()

% 1 < aleatorio < 45 [1, 3, 5, 9, 15, 45]
aleatorio=1;

amostra = 4*(21/aleatorio);
Md = zeros(amostra,32);
Mdlinha = zeros(amostra,31);

% amostralinha = 4*45 - amostra;
% Mdlinha = zeros(amostralinha,32);

%Happy
i = 1;
j = 1;
for k=1:21
%     if(rem(k,aleatorio)== 0)
        num = int2str(k);
        if k < 10
```

```
            emoh='a';
        else
            emoh='a';
        end
        filename=strcat(emoh,num);
        [data, fs] = wavread(filename);
        P = calcula_parametros1(data, fs);
        Md(i,1:31) = preenchealegre(P);
        Md(i,32)='A';
        Mdlinha(i,:) = preenchealegre(P);
        i= i+1;
end

%Angry
for k=1:21
%     if(rem(k,aleatorio)== 0)
        num = int2str(k);
        if k < 10
            emoa='b';
        else
            emoa='b';
        end
        filename=strcat(emoa,num);
        [data, fs] = wavread(filename);
        P = calcula_parametros1(data, fs);
        Md(i,1:31) = preenchebravo(P);
        Md(i,32)='B';
        Mdlinha(i,:) = preenchetriste(P);
        i=i+1;
end

%Neutral
for k=1:21
%    if(rem(k,aleatorio)== 0)
        num = int2str(k);
        if k < 10
            emoa='n';
        else
            emoa='n';
        end
        filename=strcat(emoa,num);
        [data, fs] = wavread(filename);
        P = calcula_parametros1(data, fs);
        Md(i,1:31) = preencheneutro(P);
        Md(i,32)='N';
        Mdlinha(i,:) = preenchetriste(P);
        i=i+1;
end


    %Sad
for k=1:21
```

```
%     if(rem(k,aleatorio)== 0)
        num = int2str(k);
        if k < 10
            emoa='t';
        else
            emoa='t';
        end
        filename=strcat(emoa,num);
        [data, fs] = wavread(filename);
        P = calcula_parametros1(data, fs);
        Md(i,1:31) = preenchetriste(P);
        Md(i,32)='T';
        Mdlinha(i,:) = preenchetriste(P);
        i=i+1;
end
```

Função preenchealegre

```
function Mdlinha = preenchealegre (P)
for i=1:31
    Mdlinha(i) = P(i);
end
% Mdlinha(32)= 'A';
end
```

Função Preenchetriste

```
function Mdlinha = preenchetriste (P)
for i=1:31
    Mdlinha(i) = P(i);
end
% Mdlinha(32)= 'T';
end
```

Função Preenchebravo

```
function Mdlinha = preenchebravo (P)
for i=1:31
    Mdlinha(i) = P(i);
end
% Mdlinha(32)= 'B';
end
```

Função Preencheneutro

```
function Mdlinha = preencheneutro (P)
for i=1:31
    Mdlinha(i) = P(i);
end
% Mdlinha(32)= 'N';
end
```

Função Segmentar

```
function S = segmentar(data, fs)

c = fs/2;       % Numero de colunas de S = frequencia de amostragem (44100kHz)
x periodo de discretizacao (0,5s)

l = ceil(length(data)/(fs*0.5));% Calcula o numero de linhas necessarios, a
funcao ceil arredonda para cima
S = zeros(l,c);
for i=1:(l-1);    % ate (l-1) para que na ultima linha o parametro nao exceda
a dimensao de data
    S(i,1:c) = data(1+(i-1)*c:i*c);
end
final =  length(data) - (l-1)*c; %Calcula o numero de elementos a serem
colocados na ultima linha de S
S(l,1:final) = data(((l-1)*c+1):length(data)); %Realiza a segmentacao da
ultima linha de S

end
```

Função Frequencia

```
function F0 = frequencia(data, fs)

c = fs/2;       % Numero de colunas de S = frequencia de amostragem (44100kHz)
x periodo de discretizacao (0,1s)

l = ceil(length(data)/(fs*0.5));% Calcula o numero de linhas necessarios, a
funcao ceil arredonda para cima

f_nyquist = fs/2;
n = length(data)/l;
F0 = zeros(1,l);

for i=1:l
    amostra = data(1+(i-1)*n:i*n);
    Y=2*abs(fft(amostra))/n;
    freq = linspace(0,f_nyquist, length(Y));
    Ycn = abs(fft(amostra))/n;

    [y_max index] = max(Y(2:end));
```

```
    %Look up the frequency that corresponds with the strongest magnitude
    F0(i) = freq(index);
end

end
```

Função Intensidade

```
function RMS = intensidade(S)

[l,c] = size(S); % Obtem as dimensoes da matriz S
for i=1:l
    RMS(i) = (sumsqr(S(i,1:c))/c)^.5;  % Calcula a raiz da media dos quadrados
- RMS
end

end
```

Função Normalizar

```
function F0normal = normalizar(F0)

lowbond = prctile(F0,5);   % Estes limites são adotados com base nos percentil
de 5% e 95%
highbond = prctile(F0,95); % com o objetivo de descartar os 5% inferiores e
superiores da amostra
l = length(F0);            % tornando os dados mais homogeneos
aux = 1;

for i=1:l
    if (F0(i) >= lowbond && F0(i) <= highbond)
        F0normal(aux) = (F0(i) - lowbond)/(highbond - lowbond);    % A
normalização
        aux = aux + 1;
    end
end
```

Função Silencio

```
function flag = silencio(RMS)

l = length(RMS);
flag = zeros(1,l);          % Cria o vetor flag contendo zeros
for i=1:l
```

```
    if (RMS(i) >= 0.05)
        flag(i) = 1;        % Atribui o valor um para RMS >= 0.05
    end

end
```

Função Continuidade

```
function transicao = continuidade(flag)

l = length(flag);
transicao = zeros(1,l);
if (flag(1) == 1)              % Faz a verificacao se a amostra inicia com
silencio ou nao
    transicao(1)=1;
end
for i=2:l
    if ( flag(i-1)==0 && flag(i)==1 )  % Verifica se houve a transicao
silencio-voz
        transicao(i) = 1;
    end
    if ( flag(i-1)==1 && flag(i)==0 ) % Verifica se houve a transicao voz-
silencio
        transicao(i) = -1;
    end
end

end
```

Função Descarta

```
function [Slinha, datalinha] = descarta(S, data)

[l, c] = size(S);
aux = 0;
i = 0;

rmss = intensidade(S);   % Cálcula a intensidade para poder discernir o
silencio do som
flagg = silencio(rmss);  % Atribui 0 para o silencio e 1 para o som - critério
> 0.05

while (aux == 0)   % Retorna o indice i do flag onde começa a amostra
    i = i + 1;
    if (flagg(i) == 1)
        aux = 1;
    end
end
```

```
aux = 0;
j = l + 1;

while (aux == 0)    % Retorna o indice j do flag onde termina a amostra
    j = j - 1;
    if (flagg(j) == 1)
        aux = 1;
    end
end

Slinha = S(i:j,:);   % Retorna a nova matriz S desconsiderando os periodos de
silêncio iniciais e finais
datalinha = data((i-1)*800+1:j*800); % idem acima para o vetor data

end
```

Função Parametros

```
function P = parametros(F0, RMS, flag, transicao)

l = length(F0);
P = zeros(1,31);

% A) Parametros relacionados a frequencia fundamental

F0aux = F0.*flag; % Produto de cada valor entre os vetores com objetivo de
zerar as partes silenciosas

j =1;
for i=1:l
    if (F0aux(i) > 0)        % Cria o vetor de frequencia fundamental
        f0(j) = F0aux(i);     % considerando apenas as partes de voz ativa
        j = j + 1;
    end
end
F0Normal = normalizar(f0);

P(1) = mean(f0); % Retorna a media de f0
P(2) = median(f0); % Retorna a mediana de f0
P(3) = max(F0);    % Retorna o máximo de F0
P(4) = min(F0);    % Retorna o mínimo de F0
P(5) = prctile(f0,95); % Retorna o percentil de f0 para 95%
P(6) = prctile(f0,5); % Retorna o percentil de f0 para 5%
P(7) = P(3) - P(4);  % Retorna a amplitude de F0
P(8) = P(5)-P(6); % Retorna a amplitude entre 5%-95% dos percentil
P(9) = std(f0); % Retorna o desvio padrao de f0
```

```
% B) Parametros relacionados a intensidade

RMSaux = RMS.*flag; % Produto de cada valor entre os vetores com objetivo de
zerar as partes silenciosas

j =1;
for i=1:l
    if (RMSaux(i) > 0)          % Cria o vetor de intensidade
        rms(j) = RMSaux(i);       % considerando apenas as partes de voz ativa
        j = j + 1;
    end
end

P(10) = mean(rms); % Retorna a media de rms
P(11) = median(rms); % Retorna a mediana de rms
P(12) = max(rms); % Retorna o maximo de rms
P(13) = prctile(rms,95); % Retorna o perctile de 95% de rms
% P(14) = prctile(rms,5); % Retorna o perctile de 5% de rms
P(15) = P(13) - P(14); % Retorna a amplitude entre os percentile 5%-95% de rms
P(16) = std(rms); % Retorna o desvio padrao de rms
% P(17) = mean(F0Normal); % Retorna a media de f0 normalizada
% P(18) = median(F0Normal); % Retorna a mediana de f0 normalizada

% C) Parametros relativos a duracao do som
 j=1;
 barulho(j)=0;
 fim=1; % Inicializa o silencio no primeiro segmento
 for i=1:l
    if (transicao(i) == 1)
        inicio = i;
        quieto(j) = inicio - fim; % Vetor que armazena a duracao do silencio
    end
    if (transicao(i) == -1)
        fim = i;
        barulho(j) = fim - inicio;  % Vetor que armazena a duracao dos
segmentos continuos de voz
        j = j + 1;
    end
 end

 if(inicio > fim)
     barulho(j) = l + 1 - inicio;
 end
 if(fim > inicio)
     quieto(j) = l + 1 - fim;
 end

 P(17) = mean(barulho); % Duração média de um segmento contínuo de voz
 P(18) = mean(quieto); % Duração média das pausas
 P(19) = std(quieto); % Desvio padrão das pausas
 P(20) = sum(barulho)/sum(quieto); % Razão voz/silencio
```

```
% D) Parametros relacionadaos a dinamica da frequencia fundamental

 j=1;
 F0variacao=0;
 for i=1:1
     if (transicao(i) == 1)
         F0inicio = F0(i);
     end
     if (transicao(i) == -1)
         F0final = F0(i-1);
         F0variacao(j) = F0final - F0inicio; % Armazena a variacao da
frequencia fundamental
         j = j + 1;                          % em um segmento continuo de voz
     end
 end

 j = 1;
 k = 1;
 for i=1:length(F0variacao)
     if (F0variacao(i) >= 0)
         F0UP(j) = F0variacao(i);
         j = j +1;
     end
     if (F0variacao(i) < 0)
         F0DOWN(k) = F0variacao(i);
         k = k +1;
     end
 end

 if (j == 1)
     F0UP = 0;  % No caso de não haver incrementos em F0
 end
 if (k == 1)
     F0DOWN = 0;   % No caso de não haver decrementos em F0
 end
 P(21)=mean(F0UP); % Incremento médio de um segmento contínuo de voz
 P(22)=max(F0UP); % Incremento máximo de um segmento contínuo de voz
 P(23)=std(F0UP); % Desvio padrão do incremento
 P(24)=mean(F0DOWN); % Decremento médio de um segmento contínuo de voz
 P(25)=min(F0DOWN); % Decremento máximo de um segmento contínuo de voz
(INVERTE-SE AS FUNÇÕES MIN MAX POR CONTA DO SINAL)
 P(26)=std(F0DOWN); % Desvio padrão do dencremento
 P(27)=length(F0UP)/length(F0DOWN); % Razão entre incremento e decremento

% E) Parametros relacionados a frequenncia fundamental após normailzação

 P(28) = mean(F0Normal); % Retorna a media do vetor normalizado
 P(29) = median(F0Normal); % Retorna a mediana do vetor normalizado
 P(30) = std(F0Normal); % Retorna o desvio-padro do vetor normalizado
```

```
% F) Parametro de esforço vocal - Considera a frequencia e intensidade no
mesmo instante

% P(31) = mean(F0Normal.*rms)/(mean(F0Normal)*mean(rms));

end
```

Função Calcula_Parametros

```
function P = calcula_parametros(data, fs)

S = segmentar(data, fs);  % Funcao que segmenta o vetor data em uma matriz,
                          % onde os parametros de intensidade sao avaliados
para cada linha da matriz

%[S, data] = descarta(S, data);
% Descarta os periodos iniciais e finais de silencio de cada amostra e retorna
o novo vetor data e mariz S

F0 = frequencia(data, fs); % Acha a frequencia fundamental utilizando a
transformada de fourier - Período de observação de 2s

RMS = intensidade(S); % Calcula a intensidade da amostra para cada TD (0,1seg)

flag = silencio(RMS); % Identifica os segmentos silenciosos - RMS < 0,05

transicao = continuidade(flag); % Identifica os segmentos continuos de voz

P = parametros(F0, RMS, flag, transicao); % Calcula todos os parametros
acusticos definidos para a amostra

% clear F0 RMS S data flag fs transicao;  % Limpa as variaveis desnecessarias

end % Fim do algoritmo
```

6.2 ANEXO B

Função treinamentosvm

```
function [xsup,w,b,nbsv,kernel,kerneloption] = treinamentosvm(Md,c, lambda,
verbose)

[l,coluna] = size(Md);
% ---------------------------------------------------
% Seta o numero de classes
```

```
% -------------------------------------------------------
nbclass = 4;
xapp = Md(:,1:31);

% -------------------------------------------------------
% Seta os resultados para o treinamento
% 1o. quarto das amostras - Alegre
% 2o. quarto das amostras - Bravo
% 3o. quarto das amostras - Neutra
% 4o. quarto das amostras - Triste
% -------------------------------------------------------
yapp = zeros(l,1);
linha = l/4;
for i = 1 : linha
    yapp(i,1) = 1;
end
for i = linha+1 : 2*linha
    yapp(i,1) = 2;
end
for i = 2*linha+1 : 3*linha
    yapp(i,1) = 3;
end
for i = 3*linha+1 : 4*linha
    yapp(i,1) = 4;
end

% -------------------------------------------------------
% Seta os parametros do Kernel
% -------------------------------------------------------
c = 418;
lambda = 1e-7;
kerneloption= 1;
kernel='poly';
verbose = 0;

% -------------------------------------------------------
% Treinamento da matriz
% -------------------------------------------------------
[xsup,w,b,nbsv,pos,obj]=svmmulticlassoneagainstall(xapp,yapp,nbclass,c,lambda,
kernel,kerneloption,verbose);
end
```

Função análise

```
function [mestat,estatistica] = analise(ypred)

clear mestat;
clear estatistica;
```

```
[l,c] = size(ypred);

amostra = l/4;
mestat = zeros(4,4);

i = 2;
for k = 1: 4*amostra
    if(k <= amostra)
        i = 2;
    elseif(k > amostra && k <= (2*amostra))
        i = 4;
    elseif(k > (2*amostra) && k <=(3*amostra))
        i = 1;
    else
        i = 3;
    end
    if(ypred(k) == 1)
       mestat(i,2) = mestat(i,2) + 1;
    elseif(ypred(k) == 2)
       mestat(i,4) = mestat(i,4) + 1;
    elseif(ypred(k) == 3)
       mestat(i,1) = mestat(i,1) + 1;
    elseif(ypred(k) == 4)
       mestat(i,3) = mestat(i,3) + 1;
    end
end

mestat = (mestat/amostra)*100;

estatistica = (mestat(1,1) + mestat(2,2) + mestat(3,3) + mestat(4,4))/4;

end
```

6.3 ANEXO C

Função testesvm

```
function [ypred, maxi] = testesvm(Md,xsup,w,b,nbsv,kernel,kerneloption)
xtest = Md(:,1:31);
[ypred,maxi] = svmmultival(xtest,xsup,w,b,nbsv,kernel,kerneloption);

end
```

Função svm

```
clear
```

```
clc

load Md;
[xsup,w,b,nbsv,kernel,kerneloption] = treinamentosvm(Md);

[data,fs]=wavread('triste_alegre'); %  nome do arquivo
S = segmentar(data,fs);
% [S,data]=descarta1(S,data);
F0 = frequencia(data,fs);
RMS = intensidade(S);
flag = silencio(RMS);

% variáveis pertinentes ao problema
troc=1;          %troc se refere a quantidade de Falantes
troca=0;         %troca se refere a quantidade de 0,1s de silêncio entre
falantes
n=length(F0);
media=0;
% fs=44100;
amostragem=0.5;
localdata=1;


for i=1:n

    % frequencia é diferente de zero para todos os instantes
    % em que não está considerado silêncio
    % flag = 1 para barulho
    % flah = 0 para silêncio
    verifica=F0(i)*flag(i);

    % para o caso em que o verifica = 0, ou seja,
    % ocorre o silêncio
    if verifica==0
        troca=troca+1;
        % troca é aumentado
    else
        if troca <3
            % a frequencia e dados são guardados é guardada para mais tarde
ser calculada
            falante(troc,i)=verifica;
            troca=0;
        end
    end
    % quando a troca é percebida
        if troca>2 && verifica~=0
        P(troc,:) = calcula_parametros(data(localdata:i*amostragem*fs), fs);
        localdata = 1+(i-1)*amostragem*fs;
        troc = troc+1;
        falante(troc,i)=verifica;
        troca=0;
```

```
    end
end
P(troc,:) = calcula_parametros(data(localdata:end), fs);


fprintf('\nResultado\n');
for k=1:troc

    freqmedia = sum(falante(k,:))/length(falante(k,:));
    fprintf('Frequencia Individio %d: %f \n',k,freqmedia);

    [ypred, maxi] = testesvm(P(troc,:),xsup,w,b,nbsv,kernel,kerneloption);


    if ypred==1
        fprintf('Emoção: Alegre\n\n');
    else if ypred==2
            fprintf('Emoção: Bravo\n\n');
        else if ypred==3
                fprintf('Emoção: Neutro\n\n');
            else if ypred==4
                    fprintf('Emoção: Triste\n\n');
                else
                    fprintf('Erro\n\n');
                end
            end
        end
    end
end
```

Função Experimento Fatorial

```
load MdAntigo;

% Vetor que enumera todos os parâmetros
n = 1:1:31;
% parametro que conta números de casos
k = 1;
% loop que roda a quantidade de parâmetros a ser utilizada
% param representa a quantidade de parametros que vai ser utilizada!!
for param = 1, 2, 3, 4, 27, 28, 29, 30, 31
    % cria matriz com as possibilidades para a determinada quantidade de
    % parametros a estar se estudada: 2, 3, 4, 5, 6 ... 31
    clear q;
    q = nchoosek(n,param);
    [l,c] = size(q);
    %para cada caso de teste - representam as linhas de q
   for i=1:l
       clear Mdlinha;
       %preenche as colunas de Mdlinha com os parametros do respectivo caso
       for j=1:c
```

```
            e = q(i,j);
            Mdlinha(:,j) = Md(:,e);
        end
        % treinamento do svm
        [xsup,w,b,nbsv,kernel,kerneloption] = treinamentosvm1(Mdlinha);
        % teste do svm
        xtest = Mdlinha;
        [ypred,maxi] = svmmultival(xtest,xsup,w,b,nbsv,kernel,kerneloption);
        % analise dos resultados e gravação da matriz Resultados
        [mestat,estatistica] = analise(ypred);
        m=2;
        for j=1:c
            Resultado(i,1) = estatistica;
            Resultado(k,m) = q(i,j);
            m = m + 1;
        end
        k
        k = k+1;
    end

end
```