Universidade de Brasília – UnB
Campus Gama – FGA
Engenharia Eletrônica

Sistema de Reconhecimento de Locutor Utilizando Máquinas de Vetores de Suporte e Classificadores do Tipo *Ensemble*

Bruno Vinícius Senise
e
Evaristo Ramalho Lucchezi da Silva

Orientador: Dr. Cristiano Jacques Miosso

Bruno Vinícius Barbosa Senise
Evaristo Ramalho Lucchezi da Silva

Sistema de Reconhecimento de Locutor Utilizando Máquinas de Vetores de Suporte e Classificadores do Tipo *Ensemble*

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Orientador: Dr. Cristiano Jacques Miosso

Brasília, DF
2017

Brasília/DF, Julho de 2016

Ficha Catalográfica

Bruno Vinícius Barbosa Senise
Evaristo Ramalho Lucchezi da Silva

Sistema de Reconhecimento de Locutor Utilizando Máquinas de Vetores de Suporte e Classificadores do Tipo *Ensemble*.

49p., 210 × 297 mm (FGA/UnB Gama, Engenharia Eletrônica, 2016)

Trabalho de graduação em Engenharia Eletrônica
Universidade de Brasília, Campus Gama – FGA/UnB

1. Reconhecimento de locutor
2. Aprendizagem de Máquina
3. Máquina de vetor de suporte
4. Coeficientes cepstrais
5. Classificadores Ensemble.
I. FGA/UnB

Referência

B. V. B. Senise e E. R. L. Silva (2017). Sistema de Reconhecimento de Locutor Utilizando Máquinas de Vetores de Suporte e Classificadores do Tipo *Ensemble*. Dissertação de graduação em engenharia eletrônica, Universidade de Brasília, Campus Gama, DF, 49p.

Agradecimentos Bruno

Agradeço primeiramente a Deus por me permitir concluir mais esta importante etapa de minha vida. Agradeço também, ao meu colega de pesquisa Evaristo por todo o apoio e trabalho, e ao professor Cristiano, pela excelente didática, ensinamentos e atenção providos no trabalho.

Agradeço também a minha família, meus amigos de faculdade e a minha namorada, por todo o apoio e incentivo necessários para um bom desenvolvimento deste trabalho.

Por fim, agradeço a todos os docentes que me proporcionaram esta formação pela Universidade de Brasília.

Agradecimentos Evaristo

Agradeço a Deus por ter me dado a oportunidade de chegar até aqui. Agradeço também ao orientador Cristiano, que sempre ajudou prontamente em todo o processo de criação do trabalho, e o meu colega Bruno, pelo trabalho e clima amigável na criação deste projeto.

Agradeço também a família e amigos, por sempre me ajudarem quando foi necessário, facilitando bastante a realização do trabalho.

FGA/UnB – Universidade de Brasília, Campus Gama

Sistema de Reconhecimento de Locutor Utilizando Máquinas de Vetores de Suporte e Classificadores do Tipo *Ensemble*

Bruno Senise e Evaristo Ramalho

Monografia submetida ao curso de graduacão em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Aprovada por:

Prof. Cristiano Jacques Miosso, PhD
(Orientador)

Prof. Adson Ferreira da Rocha, PhD (Examinador interno)

Prof. Leandro Xavier Cardoso, PhD (Examinador interno)

Se tornou aparentemente óbvio que nossa tecnologia excedeu nossa humanidade

(**Einstein** *Albert*)

Resumo

Vários sistemas de segurança se utilizam de recursos como senhas, identificação digital e cartões para garantir acesso a materiais ou conteúdos apenas a pessoas previamente autorizadas. Este trabalho propõe desenvolver e avaliar um sistema de liberação de acesso com base no reconhecimento de voz para identificação do locutor, utilizando características vocais previamente aprendidas aplicadas em diferentes classificadores.

A escolha da utilização de uma característica para reconhecimento de um indivíduo, deve ser principalmente atrelada à aplicação. A identificação de um rosto por reconhecimento de imagem, por exemplo, é uma alternativa que apresenta grande em alguns sistemas, mas muitas vezes não é adequada. Um exemplo disso é a identificação de um usuário para inicialização de um computador, onde muitas vezes o usuário se encontra em um ambiente com ausência de luz, inviabilizando o reconhecimento. Outro fator importante para a escolha das características é a efetividade de sua extração para um reconhecimento com baixa taxa de erros. Considerando que a voz apresenta-se como uma boa alternativa para identificação em muitas aplicações (uma porta que não deve autorizar a todos, por exemplo) e que a extração de suas características é eficiente [5], definimos que sua utilização para o reconhecimento de um indivíduo é adequada.

A utilização da voz para controle de determinada aplicação pode apresentar falhas na segurança em alguns sistemas atuais (autorizar usuário não cadastrado, bloquear usuário cadastrado ou até mesmo prover acesso sem requisição do mesmo). Por outro lado, o uso desse recurso apresenta-se como uma alternativa mais conveniente para algumas aplicações em meios de controle de acesso (ou outro tipo de controle).

A diferença entre a utilização do sinal de voz a outras características depende no meio em que o sistema será inserido. Para um sistema de controle residencial (acionamento de lâmpadas, cortinas e outros) a implementação de tal sistema é consideravelmente adequada, tendo em vista que é cômodo, e que qualquer pessoa na residência, geralmente, tem permissão para realizar tal ação. Já no ato do desbloqueio de um *Smartphone*, por exemplo, não é prático que o usuário sempre tenha que fornecer a fala como entrada, tendo em vista que isso ocorrerá frequentemente e, caso o sistema reconheça uma senha em específico, temos de saber tratar as possíveis fraudes (gravações e outros). Deve-se então analisar previamente a aplicação em que o sistema será inserido, para que tenhamos o melhor desempenho em optar por um sistema de reconhecimento de voz ou um que utilize outras características que apresentam suas peculiaridades (senha, biometria, retina e outros).

O projeto do sistema consiste em duas partes: na primeira adquire-se e processa o sinal de voz com o intuito de extrair a informação mais relevante para identificação de locutor. Nesta etapa os recursos para validação do usuário são extraídos e tratados. Em

seguida, devemos implementar o primeiro classificadores propostos (Máquina de Vetor de Suporte,Adaboost,RobustBoost e Bagging). Na segunda parte é projetado um algoritmo usando os valores de um dos classificadores avaliados acima, de maneira a realizar a identificação.

O processamento de tais sinais se faz necessário para uma eficaz análise dos recursos que irão validar o locutor. Tendo em vista que a fala é um som de natureza mais complexa (pois órgãos como os pulmões, a garganta, a boca e o nariz participam do processo de ecoação e caracterização de cada indivíduo), ruídos turbulentos provenientes do ar expelido ou do ambiente externo devem ser devidamente tratados pois podem atrapalhar toda a cadeia de verificação do usuário.

O tratamento e processamento do sinal feito inicialmente, é indispensável para uma verificação eficaz. O que acontece nos atuais sistemas de reconhecimento fonético é que, para uma melhor calibração e eficiência do sistema, um banco de dados de teste relativamente amplo precisa ser implementado e testado de forma a validar o mesmo. Essa tarefa pode ser demasiadamente difícil levando em conta o pré processamento do sinal ou a forma de teste implementada. Outros aspectos que dificultam essa etapa são as diferentes formas que cada indivíduo tem de se comunicar, como dialetos e estilos de fala, dificultando ainda mais o processamento do sinal.

A etapa de reconhecimento de locutor (apresenta maior desafio em eficiência na atualidade) estudada na segunda parte desta proposta utiliza a princípio, máquina de vetor de suporte. Basicamente esse método busca uma taxa de similaridade entre o sinal obtido e o esperado, este último já pré adquirido e processado. Essa taxa de similaridade pode ser obtida através do método convolucional, que lista o grau de sobreposição de um sinal em relação a outro, ou seja, quanto mais próximo um padrão sonoro de outro, maior esse grau.

Por fim, um hardware precisa ser desenvolvido para validação em tempo real de um indivíduo previamente cadastrado. O protótipo em que será implementado tal sistema irá realizar um controle de acesso em um sistema de fechadura. Tal fechadura poderá ser implementada utilizando artifícios eletrônicos/mecânicos como motores de passo e relés, ou uma fechadura previamente fabricada. Tal controle precisa ser cuidadosamente calibrado, pois um usuário já pré cadastrado deverá ter o mínimo de problemas ao acessar seu sistema, e outro não cadastrado nunca poderá ter êxito pois se tratando de sistemas de segurança, a tolerância a falhas deve ser mínima.

Palavras-chave: Reconhecimento de locutor, máquina de vetor de suporte, coeficientes cepstrais, classificadores ensemble.

Sumário

1 Introdução **13**

1.1 Contextualização . . . 13

1.2 Definição do Problema e Proposta de Pesquisa e Desenvolvimento . . . 15

1.3 Objetivos . . . 16

1.3.1 Objetivo Geral . . . 16

1.3.2 Objetivos Específicos . . . 16

1.4 Justificativa . . . 16

1.5 Estrutura da Dissertação . . . 17

2 Fundamentação Teórica e Estado da Arte **19**

2.1 Sinais de Voz . . . 19

2.1.1 O Sistema Vocal Humano . . . 19

2.1.2 Coeficientes Cepstrais . . . 20

2.2 Classificadores . . . 21

2.2.1 Árvores de Decisão . . . 22

2.2.2 Máquina de Vetor de Suporte . . . 23

2.2.3 Classificadores Ensemble . . . 27

2.2.3.1 Boosting . . . 29

2.2.3.1.1 Adaboost . . . 30

2.2.3.1.2 Robustboost . . . 31

2.2.3.2 Bagging . . . 32

2.3 Reconhecimento Automático de Locutor . . . 33

3 Metodologia **36**

3.1 Metodologia de Desenvolvimento . . . 36

3.1.1 Software . . . 36

3.1.2 Hardware . . . 37

3.2 Metodologia Experimental . . . 37

3.3 Metodologia de Análise . . . 39

3.3.1 Métodos de Validação Cruzada . . . 39

3.3.2 Métricas de Desempenho . . . 39

4 Resultados e discussões **42**

5 Conclusão **46**

Lista de Tabelas

4.1 Resultados das métricas dos 8 locutores para o classificador SVM. 42
4.2 Resultados das métricas dos 8 locutores para o classificador Robustboost. 42
4.3 Resultados das métricas dos 8 locutores para o classificador Adaboost. . . 43
4.4 Resultados das métricas dos 8 locutores para o classificador Bagging . . . 43
4.5 Resultados das métricas das médias dos 8 locutores, para cada classificador. 43

Lista de Figuras

2.1 Sistema Vocal Humano [4, p. 1441]. 19
2.2 Diagrama de Blocos da Extração dos Coeficientes Cepstrais. 20
2.3 Exemplo simples de árvore de decisão [27, p. 5] 23
2.4 Exemplo de SVM com 2 características e hiperplano linear em 2 Dimensões 24
2.5 SVM não linearmente separável em 2 dimensões, treinada com o uso de kernel rbf 28
2.6 Esquemático de um Classificador ensemble [27, p. 16] 28
2.7 Algortitmo de boosting genérico [27, p. 24] 29
2.8 Algoritmo do Adaboost [27, p. 25] 31
2.9 Algoritmo Bagging [27, p. 49] 33
2.10 Representação Gráfica Básica de Aprendizagem Supervisionada e Não Supervisionada [2]. 34
2.11 Tipos de Reconhecimento Automático de Locutor [19, p. 19]. 35
4.1 Representação gráfica das métricas para os classificadores. 44

1 Introdução

1.1 Contextualização

A utilização de técnicas automáticas para reconhecimento de locutor não é recorrente no mundo atual apenas pela comodidade e praticidade, mas principalmente pela segurança que o sistema oferece. Técnicas tanto para reconhecimento de locutor como para reconhecimento de palavras ou fonemas são utilizadas há pelo menos 30 anos, sofrendo bruscas evoluções nesse período [20] [22] [4] .

O ser humano possui várias características inerentes e exclusivas, um bom exemplo para entendimento é a impressão digital, ou biometria. Porém, seus sistemas ainda que com uma grande efetividade nos dias atuais, podem apresentar falhas de segurança, ou ainda inacessibilidade para certas pessoas. Quando tratamos de voz, além de inferirmos uma característica exclusiva de cada um, levamos em conta a praticidade e comodidade (tendência recorrente do avanço tecnológico), e também um meio intransferível de identificação. Portanto, em diversas aplicações, é significativamente mais coerente a utilização de tal característica, levando em conta que ela precisa ser tratada de forma extremamente cautelosa para a atenuação de erros ou eventualidades indesejáveis.

O que parece uma tarefa intuitiva e trivial para o ser humano, é um desafio grande para o ramo de tecnologia e processamento de sinais. Reconhecimento de palavras ou de locutor enfrentam problemas que nunca foram totalmente solucionados, mas a melhora de eficiência nesses tempos foi notória. A fala de um ser humano, devido às variações e inflexões, é um sinal um tanto complexo ao ser processado [20]. Para isso, devemos ensinar um computador a extrair características da voz de uma determinada pessoa, e assim tomar uma decisão, que seria basicamente como aquele sinal é classificado na aplicação.

Um conceito básico e de extrema importância para sistemas inteligentes é o de aprendizagem de máquina. Este consiste em ensinar um sistema ou um computador a encontrar soluções para um determinado problema. De forma mais específica, essa solução é encontrada extraindo padrões de dados de entrada e identificando os mesmos, de forma a tornar possível uma tomada de decisão. Tal conceito apesar de parecer específico,é uma ramificação da inteligência artificial e abrange uma área de conhecimento grande. Diversas aplicações utilizam de recursos oferecidos pela aprendizagem de máquina, dentre elas temos processamento de imagens, sequência de classificação de DNA e reconhecimento de fala [9].

Reconhecimento de palavras ou fonemas podem se dividir em duas vertentes principais: reconhecimento isolado e reconhecimento contínuo. Para sistemas de reconheci-

mento isolado, o usuário oferece como entrada ao sistema palavras isoladas, de forma que o sistema irá relacionar tais entradas com padrões previamente armazenados. Para sistemas de reconhecimento contínuo, o usuário oferece como entrada uma sequência de palavras na forma de sua fala natural, e o sistema deve ser capaz realizar o reconhecimento de todas elas [20]. Esse modelo é sem dúvidas mais complexo, pois antes de tratar de padrões previamente informados ao sistema, este deve inicialmente prover a correta separação das palavras, o que pode ser complexo levando em conta as variações de fala de cada indivíduo, e a diferença de energia que algumas palavras têm em seu início e em seu fim na pronúncia. Por outro lado, o reconhecimento contínuo de fala é o mais desejado, pela própria forma natural de comunicação do ser humano.

Dentre os modelos de reconhecimento contínuos e isolados, temos mais duas vertentes, que tratam de sistemas supervisionados e não supervisionados.Os sistemas supervisionados necessitam de um treinamento, ou seja, uma prévia entrada de dados para que os rótulos sejam definidos e o sistema saiba em quais desses devem classificar as posteriores entradas. Já os sistemas não supervisionados, que são notoriamente de maior complexidade, o sistema deve identificar padrões nas entradas para que uma entrada posterior semelhante a uma já fornecida possa ser identificada [27].

Para realizarmos um adequado reconhecimento, necessitamos então de recursos para treinarmos ou testarmos nosso sistema. Os recursos utilizados no reconhecimento vocal são os coeficientes cepstrais, que consistem no tratamento do sinal de voz em frequência utilizando algumas manipulações matemáticas para extração de características da fala. Tais coeficientes se apresentaram muito eficientes para este tipo de aplicação devido à capacidade de preservação da informação [23] [17]. Esses recursos são utilizados por diferentes sistemas de classificações no reconhecimento, que são basicamente a maneira como o computador ou a máquina irá tomar a decisão (sobre quais regras) [1].

Uma técnica de classificação frequentemente utilizada na área de reconhecimento de fala em seus primórdios é o Modelo Oculto de Markov (HMM), que basicamente trata de maneira estatística e temporal o sinal de voz bem como suas variações (pertencentes ao mesmo indivíduo). Tal técnica possui uma variedade de trabalhos e estudos diversificados, porém com pouca implementação em sistemas embarcados feita [7]. Esse motivo se dá pela necessidade de um grande banco de dados para treino, considerando em sistemas isolados que a taxa de erro nesse modelo é dada pelo número de termos treinados não classificados. Em sistemas contínuos, essa taxa de erro é obtida através dos segmentos treinados que contém erro de reconhecimento [22].

Um classificador eficiente e frequentemente utilizado é o algoritmo de máquina de vetores de suporte (SVM), que consiste em traçar um hiperplano de forma a dividir um conjunto de dadas características de maneira binária. Trabalhando tal classificador, consegue-se traçar vários hiperplanos de maneira a diferenciarmos n classificações desejadas, tomando como entrada os coeficientes cepstrais extraídos [1] [6].

Uma outra área de estudos de aprendizado de máquina são os classificadores ensemble, que realizam uma série de classificações usando classificadores fracos, gerando um classificador robusto, sendo esse classificador binário também, e com ele faz a classificação dos resultados, que permite a avaliação do desempenho.

1.2 Definição do Problema e Proposta de Pesquisa e Desenvolvimento

A segurança da informação ou de bens valiosos sempre é uma pauta a ser tratada, pois nenhum sistema de segurança funciona efetivamente para qualquer caso. Mesmo com inúmeros esforços para a melhoria da confiabilidade dos sistemas utilizando-se de identificação digital do polegar e/ou cartões de acesso com senhas (por mais complexas que sejam nos dias atuais), ainda assim ocorrem falhas e quebras de segurança que permitem acesso indevido de conteúdo ou material, ou então bloqueiam acesso para um usuário que teria a devida permissão. Uma alternativa é o uso de sinais de voz para se realizar o controle de acesso. Sistemas de reconhecimento por voz podem apresentar falhas de identificação, porém é um método que vem a ser cada vez mais cômodo em determinadas aplicações e, caso seja bem desenvolvido e implementado, pode oferecer um nível de segurança considerável e suficiente. Ao contrário de um cartão, senha ou até mesmo a identificação digital que podem ser roubadas, a voz é sempre inerente ao indivíduo. Por outro lado, a voz também não é um método que garante segurança em todos os casos, porque pode-se fazer uso de gravações de voz, tornando o sistema acessível. Uma alternativa para aumentar a eficiência da segurança, seria usar sinais vocais (utilizando métodos e classificadores adequadamente eficientes) em conjunto com algum dos sistemas de seguranças já utilizados.

Analisando o meio tecnológico que o mundo se encontra hoje, é notável ver que a utilização da voz para diversas aplicações tem sido cada vez mais frequente e conveniente (*Smartphones* [15], aplicações veiculares e outros), pois trata-se de um sistema que oferecerá um nível de segurança maior, acessibilidade, comodidade e conforto para quem o utiliza.

É proposto então o desenvolvimento de um sistema que, ao receber um determinado sinal sonoro emitido pelo usuário (fonema ou palavra), fará o processamento de maneira a analisar características extraídas pelo classificador para treinamento e, com isso, possa ser capaz de identificar o usuário ou não, caso ele não esteja no banco de vozes. Para a identificação dos padrões será aplicado fundamentos e técnicas da aprendizagem de máquina, responsável por fazer o tratamento do sinal adquirido previamente, e comparar com o sinal de voz recebido naquele dado instante. Esse aprendizado é feito através de SVM e classificadores ensemble (Adaboost, Robustboost e Bagging) através de um

treinamento supervisionado, no qual serão cadastradas as vozes (treinamento) de todos aqueles que terão o acesso concedido.

1.3 Objetivos

1.3.1 Objetivo Geral

O presente trabalho visa desenvolver um protótipo de controle de acesso para verificação da fala de um determinado locutor cadastrado utilizando diferentes classificadores, comparando o desempenho dos mesmos.

1.3.2 Objetivos Específicos

Para o desenvolvimento geral da proposta, foram listados alguns objetivos específicos:

I. Desenvolver e implementar o algoritmo de extração dos coeficientes cepstrais.

II. Desenvolver e implementar os algoritmos de classificação (supervisionados e independente de texto).

III. Treinar os sistemas a partir de um banco de dados.

IV. Realizar a correta identificação do locutor para os sistemas desenvolvidos.

V. Treinar os sistemas com sinal vocal real.

VI. Verificar a identificação do locutor especificado em todos os casos, ou seja, utilizar métricas (acurácia, precisão e outros) para validar o desempenho de cada classificador.

VII. Comparar o desempenho dos classificadores desenvolvidos.

VIII. Desenvolver o protótipo para controle de acesso.

1.4 Justificativa

A principal limitação em sistemas de seguranças utilizando-se de identificação sonora se dá na dificuldade da máquina, para reconhecer uma palavra, pois o ser humano naturalmente insere variações no seu modo de falar que o diferencia de qualquer outro, seja por idioma, sotaque, ritmo, dialetos ou simplesmente vícios de linguagem. É de fácil percepção que um sinal vocal de uma palavra, ao ser processado por um computador nunca será subsequentemente igual. Dessa forma, técnicas precisam ser desenvolvidas para a criação de um modelo que melhor irá definir as características vocais daquele indivíduo ou

daquele idioma. Além disso, para um ser humano, o sinal sonoro não é a única forma de comunicação em uma conversa. São utilizados diversos recursos não verbais como gestos, linguagem corporal e movimento labial que são grandes auxiliadores no entendimento de uma informação a ser transmitida, da forma como quem transmite deseja, o que é ainda mais difícil de expressar para uma máquina [20].

Levando em conta as limitações existentes, é importante se realizar uma pesquisa de aplicação em segurança para a área de reconhecimento de voz com uso de aprendizado de máquina, que vem tendo um avanço de desenvolvimento principalmente na última década [20]. E com isso, comprovar que pode-se usar sinais sonoros para sistemas de controle de acesso, sem que haja erros de forma a inviabilizar o uso da fala.

1.5 Estrutura da Dissertação

A presente dissertação se desenvolve como será descrito.

Na segunda seção é apresentada a fundamentação teórica acerca dos conceitos básicos a serem trabalhados. Ela descreve inicialmente como ocorre a produção da fala e suas características de um ponto de vista voltado para a fisionomia. As características extraídas que serão utilizadas nesse trabalho é aprofundada nessa seção. A fundamentação teórica e matemática sobre os classificadores a serem utilizados é descrita, bem como a relação e aplicação das características extraídas nesses sistemas. No desfecho da seção, é discutido o conceito geral de reconhecimento de locutor como um sistema. São apresentados os tipos de reconhecimento de locutor e suas necessidades para aplicação. Esse último tópico trata basicamente de como o classificador interage com as características extraídas (rotulações, validações, supervisão e outros) e ainda alguns resultados acerca dos classificadores mais utilizados no segmento de reconhecimento de locutor, incluindo os implementados no presente trabalho.

A terceira seção descreve toda a metodologia utilizada para implementação do sistema. Na primeira parte dessa seção, esta que abrange a metodologia experimental, é descrito em alto nível o sistema do ponto de vista do hardware, ou seja, quais componentes básicos são necessários, suas funções e como eles irão compor o projeto como um todo. Também é apresentada nessa sessão, uma descrição da implementação dos algoritmos classificadores e da extração das características (linguagem, software, bibliotecas e outros). A segunda parte trata da metodologia experimental, a qual descreve todos os testes realizados (além de como foram feitos) que irão permitir reprodutibilidade e falsificabilidade do sistema. Também são apresentados os testes, classificadores e parâmetros trabalhados. Para os testes, é detalhado qual banco de dados foi utilizado e as principais características de tal (número de pessoas, idade, gênero e condições de aquisição). A última parte dessa seção, a metodologia de análise, descreve como os testes realizados irão compor um resultado válido, isto é, com quais métricas (resultados estatísticos) poderemos chegar a uma

conclusão de classificação válida.

Na quarta seção deste trabalho, os resultados coletados e validados são apresentados de fato, como descrito pela metodologia de análise. Possíveis melhorias e limitações do algoritmo implementado também são discutidas. Nesta seção, são apresentadas, de forma gráfica e em tabelas, comparações entre os classificadores escolhidos no que diz respeito às métricas apresentadas na metodologia de análise.

Por fim, a última seção conclui a dissertação com relação aos resultados coletados e todo o sistema si. Esta inclui o cronograma de tarefas e estudos realizados.

2 Fundamentação Teórica e Estado da Arte

2.1 Sinais de Voz

2.1.1 O Sistema Vocal Humano

A produção da fala apresenta basicamente dois componentes principais. O primeiro deles é a fonte de excitação. A fonte de excitação corresponde ao fluxo de ar que ocorre saindo dos pulmões, conduzido pela traqueia, quando desejamos produzir a fala. O segundo componente é o "trato vocal". Este corresponde a todo o sistema que funciona como um filtro (faringe, laringe, epiglote, esôfago, cavidade oral, nasal e outros, mostrados em Figura 2.1), alterando o espectro da onda acústica (domínio da frequência) proveniente do processo de excitação. Todo esse conjunto (e cada um deles em si) carrega informações inerentes e exclusivas de cada indivíduo que serão extraídas para reconhecimento de locutor [4]. Um bom exemplo disso, É a possibilidade de se reconhecer uma pessoa apenas pela sua respiração, pois o que é produzido no processo de excitação carrega características de um determinado ser humano, semelhante ao que é acontece na fala (o que prova que tais características não são apenas resultado do trato vocal, mas em conjunto do mesmo com a fonte de excitação) [17].

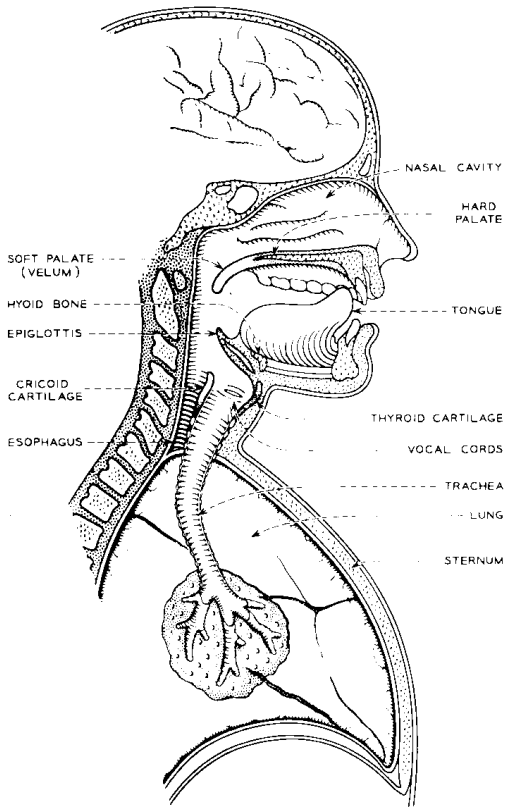

Figura 2.1. Sistema Vocal Humano [4, p. 1441].

Portanto, para todo sinal de voz, considerando ainda as aleatoriedades e inflexões no processo de produção da fala, é possível realizar a extração e processamento de características da mesma e, aplicando à um sistema classificador, podemos diferenciar tal sinal de qualquer outro. Tais características são os coeficientes cepstrais (LPCC).

2.1.2 Coeficientes Cepstrais

O Cepstrum foi inicialmente descrito em 1963 como uma técnica para reconhecimento de frequências ressonantes pelo tempo em sinais de voz [5]. Ele consiste na representação da amplitude da potência de frequências em um espectro de curto período, por considerar um sinal invariante no tempo [17] [1]. A representação do espectro consiste em uma técnica matemática mais aprimorada que permite a extração das frequências formantes de sinal de voz de curto período sem que haja uma perda significativa de informação útil para classificação. A matemática consiste em aplicar a transformada (representação no domínio da frequência) e em seguida aplicarmos as operações de módulo e logaritmo ao sinal, garantindo que o mesmo possa ser separado corretamente dentro da faixa de frequência útil que se deseja trabalhar de forma não-linear. Isto é, a voz humana atinge uma certa faixa de frequência, mas dessa faixa, uma porção restrita representa o sinal do qual podemos extrair um fonema ou até mesmo uma característica de um certo indivíduo.

Após tal processamento, a transformada inversa é realizada para obtenção dos coeficientes. Vale ressaltar que para um eficiente tratamento de sinal, uma amostragem adequada do sinal original deve ser feita, respeitando o teorema da amostragem de Nyquist. O diagrama de blocos de tal processo é representado em Figura 2.2.

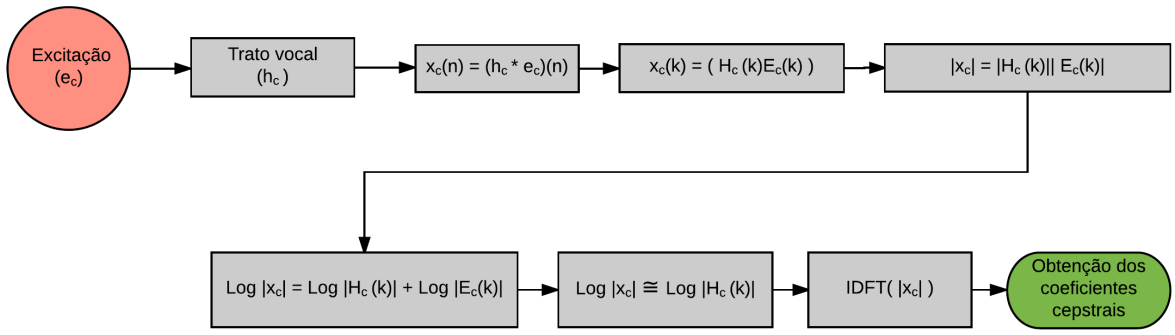

Figura 2.2. Diagrama de Blocos da Extração dos Coeficientes Cepstrais.

O diagrama acima mostra por etapas como é feita a extração dos coeficientes cepstrais. No início do processo da fala, uma excitação ocorre proveniente de fatores altamente estocásticos. Essa excitação é modificada pelo que chamamos de trato vocal. O trato vocal consiste nas características fisiológicas de cada um (caixa torácica, pulmões, diafragma, fisionomia da traqueia, boca e outros), e produz então, um sinal de saída que nos preocuparemos em tratar. Tal sinal corresponde então à uma convolução no domínio do tempo

entre a excitação e o sistema de trato vocal.

Como mostrado no diagrama, aplicamos a transformada de Fourier, o que implica em uma multiplicação no domínio da frequência. O que desejamos fazer, de fato, é aplicar o logaritmo na equação como técnica matemática para que possamos extrair os coeficientes de menores valores sem perdas significativas de informação (manipulando a multiplicação para uma soma) . Antes disso então, devemos tomar o módulo da equação, tendo em vista que queremos tratar das exclusivas eventualidades do logaritmo (números negativos, por exemplo).

Por fim, aplicamos devidamente a função logaritmo, e selecionamos os coeficientes de menores valores, que são suficientes para nossa posterior classificação. A transformada inversa de Fourier então é aplicada, para retornarmos ao domínio do tempo. Um número de amostras adequado experimentalmente definido de amostras (menores) é extraído.

Uma outra maneira de se calcular coeficientes cepstrais é através dos coeficientes *Linear Prediction Coding* ou LPC, esses por sua vez são calculados através de um método usando coeficientes de autocorrelação [16], para se extrair coeficientes cepstrais dos coeficientes LPC, é usada a seguinte recursão

$$c_m = \begin{cases} \ln \sigma^2 & m = 0 \\ a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k} & 1 \leq m \leq p \\ \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k} & m > p \end{cases} \tag{2.1}$$

com a_m sendo os coeficientes LPC, c_m os coeficientes ceptrais e σ^2 é o termo de ganho associado ao modelo LPC, lembrando que (2.1) mostra que um dado número p de coeficientes LPC, pode gerar um maior número Q de coeficientes cepstrais, geralmente algo em torno de

$$Q \simeq \left(\frac{3}{2}\right) p \tag{2.2}$$

os coeficientes ceptrais c_m gerados pela recursão acima são chamados de *Linear Predictive Cepstral Coeficientes* (LPCC).

2.2 Classificadores

Para a classificação de sinais sonoros, é fundamental que seja abordado e aplicado o reconhecimento de padrões, através da construção de bons modelos de um dado conjunto de dados, para o caso deste trabalho, sinais sonoros.

Esse conjunto de dados, em aprendizado de máquina, consiste nos vetores de características (ou treinamento), onde cada um é uma descrição de objeto usando de uma série

de características, por exemplo, os vetores de classificação para sinais sonoros são valores numéricos que descrevem o objeto, sinal sonoro, através de uma série de características (intensidade, trato vocal e etc). O número de características de um conjunto de dados é chamado de dimensão ou dimensionalidade, as características de atributos são os vetores de treinamento de instâncias, e o conjunto de dados são as amostras.

A partir das amostras, pode-se então construir um modelo. Esse processo de geração de modelos a partir de dados é chamado de aprendizado ou treinamento, que vem acompanhado de um dado algoritmo. Existem vários modelos para aprendizado de máquina, sendo que os mais comuns são os já mencionados: supervisionado e não supervisionado [27].

Para o treinamento supervisionado, o principal objetivo é a obtenção de valores com instâncias invisíveis. Esse modelo obtido é chamado de preditor, pois predizemos os valores como sendo "quadrado"ou "cruz". A isso se dá o nome de rótulo, sendo que o preditor deve ser capaz de categorizar uma instância dentro de um dado rótulo, caso seja categórico. Ao processo de aprendizado é dado o nome de classificação e o algoritmo é chamado de classificador e caso o rótulo seja numérico, o processo é uma regressão, e o algoritmo então é uma regressão linear adaptativa. Já no treinamento não-supervisionado, não se tem o uso da informação dos rótulos. É feito um algoritmo de agrupamento, capaz de descobrir padrões para os dados obtidos, classificando-os assim.

Para se definir se um dado modelo de classificação (ou regressão) para treinamento supervisionado funciona de maneira adequada, deve-se avaliar sua generalização, pois generalizar bem indica que o processo de aprendizado está avaliando bem os dados recebidos das instâncias invisíveis, gerando um baixo erro de treinamento ou predição [27].

2.2.1 Árvores de Decisão

Uma árvore de decisão consiste numa estrutura de estrutura de classificador que funciona em com arquitetura dividir para conquistar, sendo que cada nó intermediário corresponde à um teste de características, sendo assim a cada novo dado de treinamento, serão feitas ramificações de acordo com os valores das características, sendo que no nó final de cada ramificação tem-se um rótulo associado. Para se fazer a validação, uma série de testes é feita nos dados de maneira que seja associado o dado da validação com algum dos nós finais da árvore, gerando assim um rótulo pra aquele dado [27], um exemplo simples de árvore de decisão pode ser visto na Figura 2.3.

As árvores de decisão são geralmente processos recursivos, pois o tem-se todo um procedimento para que seja feita de qual ramificação seguir e porque,ou seja, a chave para o bom funcionamento do algoritmo é como selecionar os testes de características.

É um caso comum que, árvores de decisão que possuem uma boa eficácia para amostras de treinamento, não possua uma péssima generalização da informação, enquanto

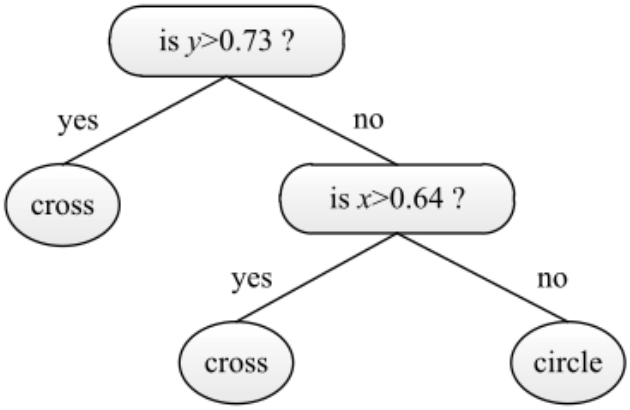

Figura 2.3. Exemplo simples de árvore de decisão [27, p. 5]

.

algumas árvores que não funcionam tão bem para o treinamento acabam validando melhor, gerando o caso de overfitting, isso pode acontecer por conta de ruído presente nas amostras de treinamento que acaba gerando um teste de características ineficiente [27].

Por conta da baixa generalização, árvores de decisão costumam ser bastante usadas como classificador-base para classificadores ensemble, que através do processo de iteração, consegue gerar e combinar várias árvores, criando um processo sem overfit e com alta capacidade de generalização

2.2.2 Máquina de Vetor de Suporte

As máquinas de vetor de suporte (SVM) são classificadores binários que utilizam aprendizado de máquina para separação de um conjunto de vetores de treinamento. A ideia foi oficialmente desenvolvida em meados da década de 90 [6], no entanto, estudos prévios acerca da criação de aprendizado de máquina utilizando de redes neurais para criação de planos lineares [26] e criação de hiperplanos de separação ótima [25] foram feitos, tais que possibilitaram o desenvolvimento do algoritmo proposto por Vapnik.

As SVMs implementam então a seguinte ideia: mapear os vetores de entrada em um espaço multidimensional Z através de um mapeamento não linear escolhido a princípio (*kernel*). Nesse espaço a superfície de decisão linear é construída de forma a garantir a habilidade de classificar corretamente da SVM.

Essa decisão é feita através da criação dos hiperplanos de separação ótima, que consistem na função de decisão linear com a maior distância no espaço entre os vetores de 2 classes. Para se construir os hiperplanos, precisa-se de poucas amostras dos dados treinados, os chamados vetores de suporte, que se localizam nas extremidades do conjunto dos valores no espaço. Calcula-se então a distância mínima entre eles, e no meio dessa distância, é gerado o hiperplano ótimo [6]. É demonstrado na experimentação que, caso

a classificação seja feita corretamente através de um hiperplano de separação ótima, a probabilidade de ocorrência de um erro de classificação é limitada pela razão entre o número de vetores de suporte e o número de vetores de treinamento,

$$E[P(Erro)] = \frac{E[Vs]}{Vt}, \tag{2.3}$$

em que $E[P(Erro)]$ é a Estimativa de probabilidade de Erro, $E[P(Vs)]$ número de vetores de suporte usados para classificar e Vt o número de vetores de treinamento.

Na Figura 2.4 podemos ver um exemplo de SVM simples com apenas 2 classes. Nesse exemplo, pode-se claramente perceber a posição dos vetores de suporte, da distribuição dos dados treinados de uma dada característica e da posição do hiperplano.

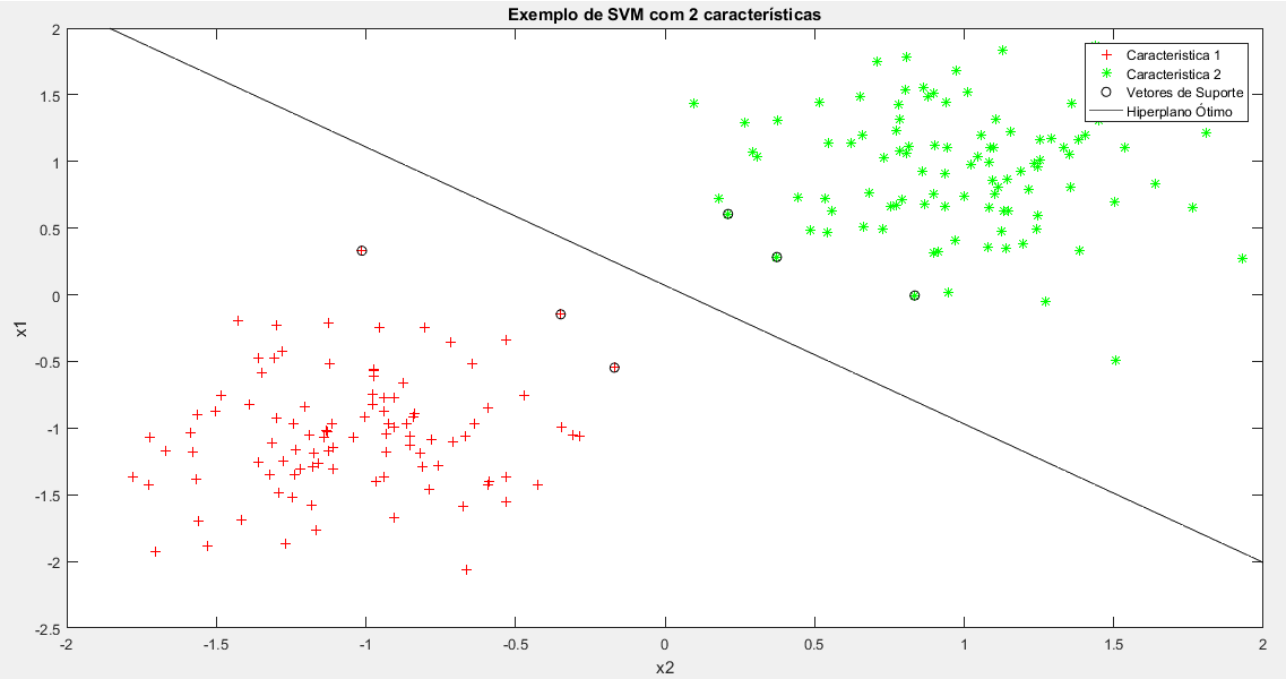

Figura 2.4. Exemplo de SVM com 2 características e hiperplano linear em 2 Dimensões

.

Para se calcular um hiperplano ótimo linear de uma SVM considerando-se um treinamento com 2 rótulos mostrados na figura 2.4, caso exista um vetor $\vec{w}$ e um escalar b, considerando os valores treinados como x_i e os rótulos como y_i, que satisfaçam as seguintes desigualdades,

$$\vec{w} \cdot \vec{x}_i + b \geq 1, y_i = 1 \tag{2.4}$$

$$\vec{w} \cdot \vec{x}_i + b < -1, y_i = -1 \tag{2.5}$$

manipulando (2.4) e (2.5) obtem-se,

$$y(\vec{w} \cdot \vec{x}_i + b) > 1 \tag{2.6}$$

ou

$$y(\vec{w} \cdot \vec{x}_i + b) - 1 = 0 \tag{2.7}$$

para vetores nas extremidades das projeções.

O hiperplano mostrado na figura 2.4 é representado por

$$(w_0 \cdot x + b_0) = 0 \tag{2.8}$$

que faz a separação dos valores treinados com a máxima distância entre os valores de cada rótulo. Calculando a distância entre as projeções dos vetores de treinamento temos

$$d = (S_M - S_m)\frac{w}{|w|} = \frac{2}{|w|} \tag{2.9}$$

onde d é o vetor de distância, $\frac{w}{|w|}$ corresponde ao vetor $\vec{x}$ normalizado. S_M e S_m correspondem aos vetores de suporte nas extremidades de cada região.

Para se maximizar essa distância, são usados multiplicadores de Lagrange α_i, obtém-se então

$$W = \frac{1}{2}\|w\|^2 - \sum_i \alpha(y_i(\vec{w}_i \cdot \vec{x}_i + b) - 1)) \tag{2.10}$$

com W correspondendo à distância maximizada.

Para calcular o valor do vetor $\vec{w}$, faz-se a derivada da distância maximizada com relação ao próprio ($\vec{w}$, obtendo

$$\vec{w} = \sum_i \alpha\vec{x}_i \, y_i \tag{2.11}$$

através desse valor podemos afirmar que $\vec{w}$ é nada mais que a soma de alguns vetores lineares. Calculando agora a derivada de L com relação a b, obtém-se

$$\sum_i \alpha y_i = 0 \tag{2.12}$$

Substituindo agora os valores encontrados nas equações (2.11) e (2.12) acima em (2.10)

$$W = \frac{1}{2}\left(\sum_i \alpha_i\vec{x}_i \, y_i\right)\left(\sum_j \alpha_j y_j\vec{x}_j\right) - \left(\sum_i \alpha y_i\vec{x}_i\right)\left(\sum_j \alpha_j y_j\vec{x}_j\right) - \sum_i \alpha y_i + \sum_i \alpha_i \tag{2.13}$$

reorganizando a equação, temos

$$W = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j \tag{2.14}$$

que é a equação que, de fato, maximiza a distância entre os dois vetores.

Colocando agora essa representação de w em (2.11) na equação (2.5),

$$\sum_i \alpha_i y_i \vec{x}_i \cdot \vec{u} \geq 0 \tag{2.15}$$

observa-se então que a regra de decisão depende apenas do produto interno entre $\vec{x}_i$ e $\vec{u}$.

No entanto, essa maximizacao das distâncias entre os hiperplanos das extremidades muitas vezes não pode ser feita sem que um erro de classificação seja gerado, esse erro corresponde às amostras de treinamento que ficaram entre o hiperplano de uma extremidade e o hiperplano ótimo, nesse caso deseja-se obter uma separação com o mínimo possível de erros, para lidar com essa abordagem introduz-se a função

$$\phi(\xi) = \sum_{i=1}^{l} \xi_i^{\sigma} \tag{2.16}$$

que, para um valor baixo de $\sigma > 0$, sujeito as condições

$$y(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i \tag{2.17}$$

$$\xi_i \geq 0 \tag{2.18}$$

descreve o número de erros de treinamento, sendo que $\xi_i = 0$ para vetores fora das regiões entre hiperplanos, minimizá-la descreve um espaço com o mínimo número de erros de treinamento.

Caso esses dados forem excluídos do conjunto de treinamento, pode-se então gerar um conjunto que não gerará erros de treinamento, podendo assim gerar um hiperplano de separação ideal. Para isso, minimizamos a equação

$$\frac{1}{2} \left\| w^2 \right\| + F(u) C \sum_{i=1}^{l} \xi_i^{\sigma} \tag{2.19}$$

com $F(u)$ sendo uma função monotônica convexa e C uma constante.

Para um C suficientemente grande, σ suficientemente pequeno $\sigma = 1$, o vetor $w0$ e $b0$ que minimizam (2.19), considerando (2.17) e (2.18), é determinado então o hiperplano ótimo que minimiza os erros do conjunto de dados de treinamento e separa os dados restantes maximizando a distância como descrito anteriormente em (2.18), considerando agora os erros de treinamento, a equação fica,

$$W = \sum_i \alpha_i - \frac{1}{2}\left(\sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j + \frac{\alpha_{max}^2}{C}\right) \tag{2.20}$$

sendo α_{max} o valor do maior multiplicador de Lagrange.

Para casos em que um hiperplano linear não consegue fazer a classificação de maneira adequada, deve-se transformar os vetores para outro espaço de N dimensões, no qual é possível fazer a separação linear sem erros. Para se mover ambos os vetores $\vec{x}_i$ e $\vec{u}$ é usada a função *phi*, que transforma uma função de uma dimensão inicial m para uma M, de maneira que nessa dimensão seja possível a separação linear das amostras de treinamento, diferente da dimensão inicial, sendo assim a (2.15) fica

$$\sum_i \alpha_i y_i \phi(\vec{x}_i) \cdot \phi(\vec{u}) \geq 0 \tag{2.21}$$

Esse produto ponto a ponto entre os dois vetores transformados é o *kernel* do sistema, e é definido como

$$K(\vec{x}\,, \vec{u}\,) = \phi(\vec{x}) \cdot \phi(\vec{u}) \tag{2.22}$$

ou seja

$$\sum_i \alpha_i y_i K(\vec{x}\,, \vec{u}) - b \geq 0 \tag{2.23}$$

Existem inúmeros tipos de *kernels* para espaços com mais N dimensões. Na figura 2.5 tem-se um exemplo que é feito o uso de um kernel rbf, que possui o formato,

$$K(u, v) = \exp\left(-\frac{|u - v|}{\sigma}\right) \tag{2.24}$$

2.2.3 Classificadores Ensemble

Os métodos ensemble treinam múltiplos classificadores, ao contrário dos outros métodos de aprendizado de máquina que constroem apenas um classificador contendo todos os dados de treinamento, os métodos ensemble controem uma série de classificadores e os combina criando um classificador robusto, a Figura 2.6 mostra a arquitetura padrão de um classificador ensemble [27].

Um ensemble contém um número de classificadores de base, que são gerados a partir dos dados de treinamento por um algoritmo base de aprendizado de máquina, este podendo ser uma rede neural, árvore de decisão, KNN, ou outro algoritmo pré especificado. Nos classificadores ensemble, o mais normal é usar somente um algoritmo-base para classificação, garantindo homogeneidade dos classificadores, levando também ao uso de ensembles homogêneos [27].

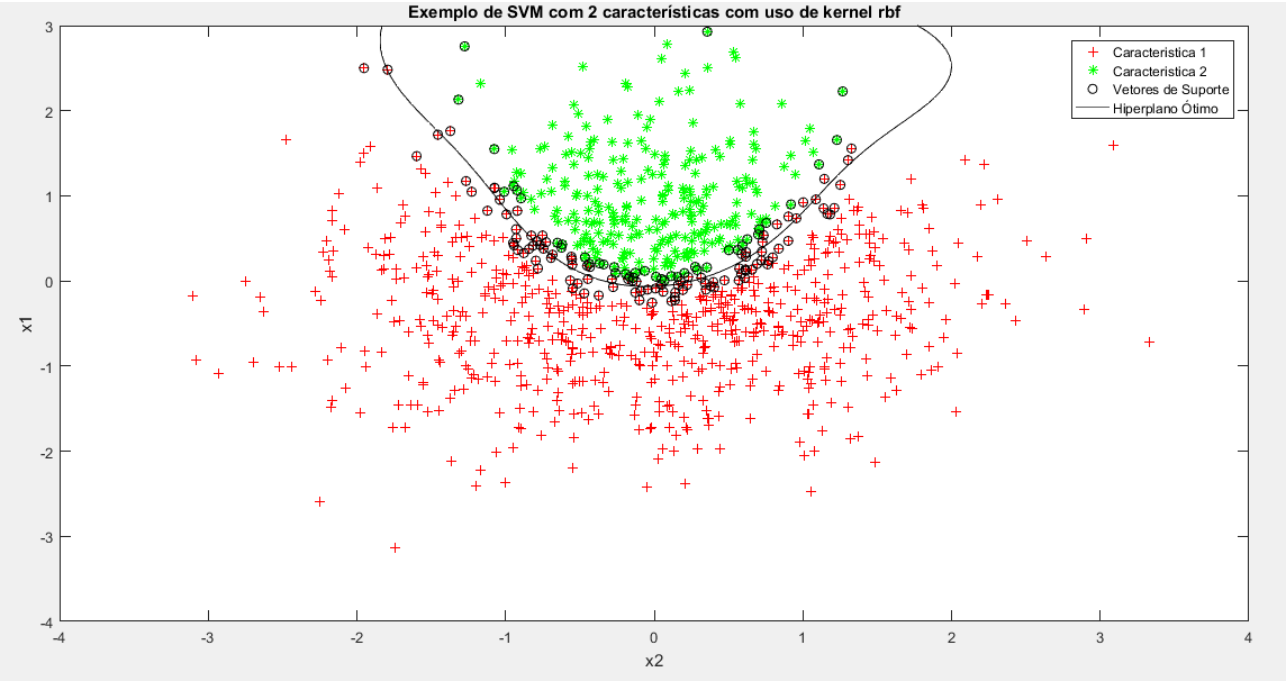

Figura 2.5. SVM não linearmente separável em 2 dimensões, treinada com o uso de kernel rbf

.

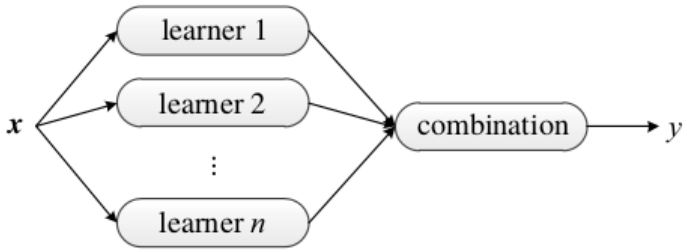

Figura 2.6. Esquemático de um Classificador ensemble [27, p. 16]

.

A Habilidade de generalização de um ensemble é geralmente muito maior do que as dos classificadores de base separadamente. Por isso, uma das aplicações mais comuns para classicadores ensemble é o boost em classificadores fracos, de maneira a obter resultados semelhantes ao de um classificador robusto, ou seja, nessa área os classificadores de base também são conhecidos como os classificadores fracos [27].

Na área de métodos ensemble existem três tipos de abordagem que são comumente utilizadas [27]:

- Combinação de classificadores: Estudado na área de reconhecimento de padrões, trabalha com classificadores robustos e tenta organizar seus resultados e criar regras de combinação para gerar combinações de classificadores robustos.

- Ensembles de classificadores fracos: Estudado na área de aprendizado de máquina,

trabalha usando um classificador fraco e a através de algoritmos robustos, dão potencializam a performance de fraco para robusto. Essa área de estudo gerou deu origem aos classificadores ensemble de Boosting e Bagging como: Adaboost,RobustBoost,Bag.

- Mistura de especialistas: Estudado na área de deep learning, o ensemble tenta aprender uma mistura de modelos paramétricos conjuntamente e usa regras de combinação para obter uma solução média.

2.2.3.1 Boosting

O termo boosting se refere a ideia de transformar um classificador fraco em robusto usando um processo iterativo e sequencial, ou seja cada classificador (exceto o primeiro), usa dos dados da aprendizagem anterior. Intuitivamente, o classificador fraco é melhor para aprendizagem aleatória, enquanto o classificador robusto tem uma performance mais próxima do ideal. [24] provou que é possível que um classificador fraco se tornar robusto, desenvolvendo então o primeiro algoritmo de boosting visto na Figura 2.7.

Input: Sample distribution $\mathcal{D}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:
1. $\mathcal{D}_1 = \mathcal{D}$. % Initialize distribution
2. **for** $t = 1, \ldots, T$:
3. $h_t = \mathfrak{L}(\mathcal{D}_t)$; % Train a weak learner from distribution $\mathcal{D}_t$
4. $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t
5. $\mathcal{D}_{t+1} = Adjust\_Distribution(\mathcal{D}_t, \epsilon_t)$
6. **end**

Output: $H(\boldsymbol{x}) = Combine\_Outputs(\{h_1(\boldsymbol{x}), \ldots, h_t(\boldsymbol{x})\})$

Figura 2.7. Algortitmo de boosting genérico [27, p. 24]

A ideia geral de um algoritmo de boosting é bem simples, suponha uma dada distribuição (D1) com três espaços igualmente distribuídos por exemplo, e seja treinado um classificador(h1) com 50 % de chance de erro(bastante não desejável), ao classificar cada uma dessas 3 distribuições, 2 foram classificadas corretamente (X1 e X2) e 1 falhou(X3), para corrigir esse 1/3 de falha, pode-se por exemplo fazer uma derivação desse espaço, de maneira que o erro visto no primeiro classificador fique mais evidente, treinando e classificando novamente, houve acerto em X1 e X3 mas erro em X2, esse segundo classificador(h2) também é um classificador fraco e possui alta capacidade de falhar,e caso o juntássemos com h1, ainda não teríamos o classificador robusto desejado, pois haveriam algumas falhas nas classificações de X3 e X2. Caso mais uma derivação seja feita, de maneira a acentuar os erros da junção entre h1 e h2, e esse novo classificador treinado (h3) classifique corretamente X2 e X3, aí teria-se o classificador robusto ideal, pois a maioria das instâncias sempre faria a classificação correta[27]. O algoritmo genérico de boosting

que gera o strong learning está mostrado em Figura 2.7, o algoritmo não é real pois existem funções citadas sem o especificações como:Adjust_Distribution e Combine_output, algumas aplicações desse algoritmo são o Adaboost e o Robustboost.

Um classificador fraco normalmente utilizado para realização de classificações ensemble é a árvore de decisão, que consiste num algoritmo que

2.2.3.1.1 Adaboost O algoritmo Adaboost(Adaptive Boosting) desenvolvido em 1995 por Yoah Freund e Robert E. Schapire [13], é o método boosting mais comum atualmente e segue a linha de raciocínio discutida anteriormente, pois tem-se o treinamento usando um classificador fraco (linha 3 do algoritmo de boosting genérico descrito em 2.7 e a estimativa de erro do classificador de distribuição Dt (linha 4), feita através da avaliação de quais as amostras foram corretamente classificadas da distribuição e caso o erro seja acima de 50 %, significa que o classificador não consegue ser melhor do que um classificador aleatório, ou seja, invalida completamente o classificador descartando-o.

A função Adjust_Distribution descrita anteriormente no Adaboost é composta pelo cálculo de coeficientes alpha, estes serão calculados de acordo com o erro do classificador através da equação,

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \tag{2.25}$$

e serão posteriormente usados para ponderar cada um dos classificadores gerados no loop. Além disso deve-se atualizar os pesos para cada amostra,estes serão definidos de acordo com a classificação e o com o valor de alpha, estes valores de peso serão maiores para amostras que foram incorretamente classificadas e menores para as classificações corretas, o valor inicial padrão é 1, e varia a cada treinamento, a formula usada para realizar a ponderação dos pesos para as amostras estão descritas em .

$$D_{t+1} = \frac{D_t \exp(-\alpha_t f(x) h_t(x))}{Z_t} \tag{2.26}$$

com Z_t correspondendo a um fator de normalização.

Depois de várias N repetições do algoritmo, criando uma série de classificadores h_t, realiza-se o procedimento representado pela função Combine_output na seção anterior, que no Adaboost corresponde ao somatório dos alphas multiplicando os valores de uma amostra x de todos os classificadores e avaliando a classe correta através do sinal da resposta, para o caso de um classificador com duas classes, a fórmula que descreve a classificação final para cada amostra é descrita em (2.27), o algoritmo geral do Adaboost está descrito na Figura 2.8

$$H(x) = sign \left(\sum_{t}^{T} \alpha_t h_t(x) \right) \tag{2.27}$$

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:

1. $\mathcal{D}_1(\boldsymbol{x}) = 1/m$. % Initialize the weight distribution
2. **for** $t = 1, \ldots, T$:
3. $h_t = \mathfrak{L}(D, \mathcal{D}_t)$; % Train a classifier h_t from D under distribution $\mathcal{D}_t$
4. $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t
5. **if** $\epsilon_t > 0.5$ **then break**
6. $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$; % Determine the weight of h_t
7. $\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases}$
$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$ % Update the distribution, where
% Z_t is a normalization factor which
% enables $\mathcal{D}_{t+1}$ to be a distribution
8. **end**

Output: $H(\boldsymbol{x}) = \mathtt{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$

Figura 2.8. Algoritmo do Adaboost [27, p. 25]

2.2.3.1.2 Robustboost Desde a invenção do algoritmo do Adaboost, que foi um grande avanço na área de aprendizado de máquina, por ser um algoritmo que conseguia evitar o overfit,ou seja, um treinamento que só classifica corretamente dentro das amostras de treinamento, sem generalização. No entanto o algoritmo apresentava um problema com a inserção de ruído aleatório no rótulos das amostras de treinamento, que acabava por deteriorar completamente os resultados. Friedman propôs um algoritmo para contornar o problema chamada de gentle Adaboost ou Logitboost, que possuia mais tolerância com ruídos, isso se devia ao igual peso dado para todas as amostras que foram mal treinadas pelos algoritmos de apredizado fracos, diferentemente do Adaboost que aumenta o peso, o que acaba gerando uma maior tolerância com ruído.

Foi então criado o algoritmo chamado Robustboost por Yoav Freund em 2009 [12], é uma variação do algoritmo do Brownboost [11], que é baseado no algoritmo *Boost by Majority* (BBM) [13], esse algoritmo é baseado na ideia de que o número de iterações até a criação do classificador forte é definido pelo algoritmo a partir de uma entrada de erro definida pelo usuário. A maioria dos algoritmos de boosting mais comuns como o Adaboost, colocam menos peso em amostras que já foram bastante acertadas por classificadores anteriores, no caso do BBM, ele coloca menos peso em amostras que possuem muitas classificações erradas, colocando-os então nas medições de erros de treinamentos. Uma desvantagem desse algoritmo é que ele não é adaptativo, ou seja, não possui a ideia de colocar pesos em cada um dos classificadores h treinados, para gerar a saída, o Brownboost é justamente o algoritmo BBM adaptativo rodando c vezes,sendo c parâmetro pré computado através do erro.

O algoritmo do Robustboost é bastante similar ao do Brownboost só que, ao invés de minimizar o erro dos classificadores, busca minimizar o númerode amostras para treinamento para os quais as margens normalizadas m sejam menores do que um dado valor

$\theta > 0$.

Sua função de perdas possui a forma

$$\phi(m,t) = 1 - erf\left(\frac{m - \mu(t)}{\sigma(t)}\right) \tag{2.28}$$

e com erf sendo a função de erro

$$erf(a) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{a} \mathrm{e}^{-x^2}\, \mathrm{d}x \tag{2.29}$$

$\mu(t)$ e $\sigma(t)$ são definidos pelas equações

$$\sigma^2(t) = (\sigma_f^2(t) + 1)\mathrm{e}^{2(1-t)} - 1 \tag{2.30}$$

$$\mu(t) = (\theta - 2\rho)\mathrm{e}^{(1-t)} + 2\rho \tag{2.31}$$

derivando (2.28) obtem-se a função de peso para as amostras

$$w(m,t) = \exp\left(-\frac{(m - \mu(t))^2}{2\sigma(t)^2}\right) \tag{2.32}$$

O parâmetro θ é o objetivo, e aumentando esse valor tem-se a diminuição da diferença de performance entre os dados usados no treinamento e de validação. O algoritmo termina de rodar quando t alcança o valor 1, caso o valor de erro seja muito baixo, o algoritmo não termina de rodar, a ideia é achar o mínimo valor para o erro ϵ, este definido previamente, que garante uma quantidade razoável de iterações. Com ϵ e $\sigma_f = 0.1$ para se evitar instabilidade numérica com t perto de 1, calcula-se o valor de ρ, possibilitando o cálculo mostrado em (2.32). Depois de feita as iterações a hipótese final é feita de maneira semelhante ao Adaboost.

2.2.3.2 Bagging

De acordo com a maneira em que são gerados os classificadores de base, exitem dois tipos de paradigmas em métodos ensemble, em um deles esses classificadores são gerados sequencialmente (caso já explicado dos algoritmos de boosting), e em outro método são gerados paralelamente, que é o caso de algoritmos de bagging. A principal motivação no uso de algoritmos bagging é explorar a independência dos classificadores de base, diferentemente dos classificadores boosting que exploram a dependencia (treinamento com pesos nas amostras), e através disso minimizar o erro combinando esses classificadores [27].

Através da desigualdade de Hoeffding, pode-se notar que o erro de generalização reduz exponencialmente de acordo com o número T de classificadores independentes utilizados [27]. No entanto, é praticamente impossível gerar classificadores de base independentes, pois eles são gerados a partir dos mesmos dados de treinamento, uma maneira de

reduzir a dependencia é a escolha aleatória dos dados de treinamento. Uma outra vantagem no uso de métodos de ensemble paralelos é que a velocidade do treinamento pode ser bastante alta, caso estejam-se usando computadores com processador multi-core.

O algoritmo de Bootstrap Aggregating (Bagging) foi desenvolvido em 1996 por Breiman [3] e usa a ideia de classificadores de base independentes discutida anteriormente. Para que sejam gerados diferentes classificadores de base, é aplicado a amostragem bootstrap [8], para obter os subconjuntos de dados a serem usados para treinar esses classificadores. Essa base de dados funciona da seguinte forma, supoe-se uma base de dados de treinamento com um número M de exemplos de treinamento, usando a técnica de amostragem com repetição, será gerada uma amostra com M exemplos de treinamento a partir da base anterior, no entanto alguns exemplos de treinamento serão repetidos enquanto outros serão excluídos, fazendo esse processo T vezes, T amostras de M exemplos de treinamento são geradas, sendo cada uma usada pra treinar um classificador de base.Após feita as iterações, tem-se o classificador robusto

$$H(x) = sign\left(\sum_{i=1}^{T} h_i(x)\right) \tag{2.33}$$

Para se fazer o agrupamento das saídas dos classificadores de base, o bagging adota votação para classificação e cálculo da média para regressão.No caso da classificação, o rótulo que tiver o maior número de votos é dado como a classificação correta,nota-se também que o bagging pode trabalhar perfeitamente com classificadores binários ou multiclasse. O algoritmo de Bagging está mostrado na Figura 2.9.

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of base learners T.

Process:
1. **for** $t = 1, \ldots, T$:
2. $h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$ % $\mathcal{D}_{bs}$ is the bootstrap distribution
3. **end**

Output: $H(\boldsymbol{x}) = \underset{y \in \mathcal{Y}}{\arg\max} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$

Figura 2.9. Algoritmo Bagging [27, p. 49]

2.3 Reconhecimento Automático de Locutor

Os sistemas de reconhecimento automático de locutor podem ser desenvolvidos com o objetivo de suprir diferentes requisitos de distintas aplicações. Basicamente, tais sistemas podem se dividir em dependentes ou independentes de texto. Um sistema dependente

de texto, é aquele que requer que um determinado fonema (ou sentença) previamente conhecido seja fornecido como entrada para classificação. Por outro lado, um sistema independente de texto não requer o fornecimento de um determinado fonema ou sentença, pois ele apenas extrai as características e realiza a classificação, de modo que este último necessita de um volume de dados para treinamento consideravelmente maior para uma boa eficácia [4].

Uma característica básica no treinamento de um sistema de reconhecimento automático de locutor, é a supervisão ou não-supervisão do mesmo. Em sistemas supervisionados, são fornecidos ao sistema rótulos (grupos) nos quais as entradas poderão pertencer, de forma que o sistema irá classificar qual desses rótulos (identificações) a entrada fornecida pertence [27]. Em sistemas não-supervisionados, as entradas são fornecidas e mapeadas, de forma que o próprio algoritmo deve realizar a rotulação das características fornecidas de modo a relacionar as entradas com alguma identificação específica (não previamente informada). Uma simples representação desses tipos de treinamento podem ser vista em Figura 2.10.

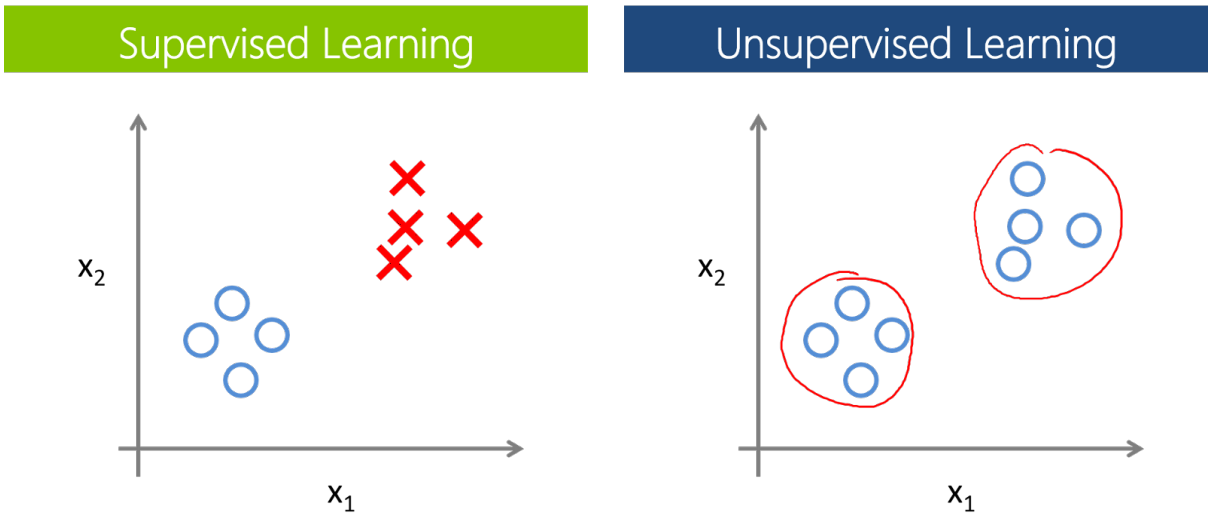

Figura 2.10. Representação Gráfica Básica de Aprendizagem Supervisionada e Não Supervisionada [2].

Outra bifurcação que encontramos nos projetos de tais sistemas, é a identificação e verificação de locutor. Na identificação de locutor, uma entrada de teste é fornecida e o sistema identifica de qual locutor do grupo (já treinado) aquela entrada pertence. A identificação de locutor pode ainda ser um sistema com ou sem rejeição. Nos sistemas com rejeição, o nível de similaridade da característica inserida com o rótulo precisa atingir um certo limiar, ou a classificação será rejeitada. Por outro lado, na verificação de locutor, uma entrada é fornecida e o sistema deve aceitar ou rejeitar a identidade do locutor [18]. A diferença entre sistemas com e sem rejeição consiste no número de decisões a serem tomadas. Enquanto a identificação de locutor precisa tomar um número de decisões

correspondentes ao número de rótulos treinados no sistema, a verificação de locutor toma apenas uma, se a entrada pertence ou não ao rótulo [19].

Nesse trabalho, trataremos de sistemas independentes de texto, supervisionados e que realizam a verificação de locutor. Desconsiderando a característica de supervisão, podemos classificar os sistemas de reconhecimento automático de locutor de acordo com Figura 2.11

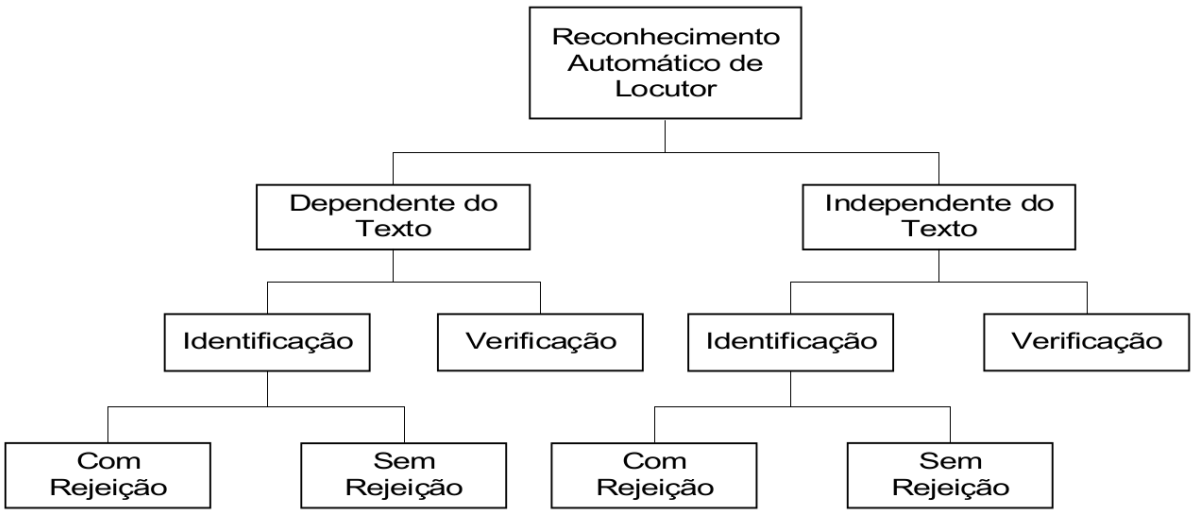

Figura 2.11. Tipos de Reconhecimento Automático de Locutor [19, p. 19].

Com a individualidade nas características que são extraídas dos sinais de voz resultante do trato vocal (diferente para cada ser humano), conseguimos distinguir indivíduos previamente rotulados no sistema. A qualidade e eficiência do sistema que irá realizar o reconhecimento de locutor está diretamente ligada com a correta extração das características a serem classificadas, e com a escolha do classificador em si [15].

Um método que foi utilizado nos primórdios do reconhecimento de fala e de locutor, é o Modelo Oculto de Markov. A implementação de tal técnica para classificação porém não é a mais eficiente nos dias atuais. Utilizando os coeficientes cepstrais em [21], esse sistema apresentou uma taxa de 75% de acerto. Outro método tradicional (até nos dias atuais) de classificação é a Mistura de Gaussianas (GMM), que estima a distribuição adjacente de probabilidade predominante nas características extraídas. Tal classificador, utilizando os coeficientes cepstrais como características, apresentou uma taxa de 70.8% de acurácia em [15].

Avaliando agora os resultados presentes na literatura sobre os classificadores a serem analisados no presente trabalho, verificamos na implementação de uma SVM em [21] uma taxa de acurácia de 84%, ainda utilizando os coeficientes cepstrais como características escolhidas.

3 Metodologia

3.1 Metodologia de Desenvolvimento

3.1.1 Software

O algoritmo utilizado para treinamento da SVM e classificação das características consiste em algumas funções que estão disponíveis no MATLAB e Octave, além de algumas que foram desenvolvidas como:

- sort_samples(), recebe os nomes dos arquivos de som de cada usuário, e divide aleatoriamente 70% dos nomes em um grupo e 30% em outro grupo, que serão os grupos de treinamento e validação respectivamente.
- energy_signal(), recebe o sinal de voz e retorna na saída posições no sinal de voz em que se possui informações de potência relevantes, retirando janelas do sinal que contêm pouca energia (o equivalente ao silêncio em um sinal sonoro).
- calculocepstrum(), adquire o sinal de voz, a informação das posição das janelas de tempo dada pela função energy_signal(), e calcula os coeficientes ceptrais de cada uma dessas janelas, retornando uma matriz com esses valores.
- calculo_ceps_sons(), recebe os várias amostras de som de um locutor, e através da função calculocepstrum() e retorna uma matriz com os coeficientes cepstrais (LPCC) de todos as janelas de todos os arquivos de som.
- matlab2octavesvm(), função que converte a estrutura gerada pela svmtrain, ou o objeto do tipo *ClassificationSVM* gerado pela função *fitcsvm*, para uma estrutura que será lida no Octave instalado em uma raspberry PI.
- svmclassifyoctave(), função que realiza a classificação das caracterísitcas baseada na estrutura de svm gerada na função matlab2octavesvm(), passando no primeiro argumento a estrutura e no segundo a matriz com a quantidade de características.
- training_validation_sound_files(), através dessas matrizes obtidas anteriormente, usa a função *fitcsvm* do MATLAB para fazer o processo de treinamento e o método *predict* do objeto *ClassificationSVM* para a validação da SVM, e usa a função *fitensemble* para fazer o treinamento usando os algoritmos de Adaboost, RobustBoost e bagging usando árvore de decisão como classificador-base, feito o treinamento, é gerado um objeto *ClassificationEnsemble* ou *ClassificationBaggedEnsemble* e usado

o método *predict* para a validação. Calcula-se e retorna a precisão, acurácia, especificidade e sensibilidade para cada classificador, além de gravar uma célula que contém as N svms geradas pela função matlab2octavesvm(), cada uma contendo o treinamento de cada um dos N usuários do sistema contra todos os outros, essa célula e armazena no arquivo 'svm_trained.mat'

- myrbf(), função para gerar uma rbf, de maneira a facilitar a implementação desse *kernel* no Octave.

- realtimeaquisition_sounds, script que roda infinitamente, e quando alguém pressiona um botão faz a gravação do sinal sonoro até que o mesmo botão seja pressionado novamente, com esse arquivo de som captado, são extraídos os LPCCs e feito o teste destes com a svm gravada num arquivo .mat pelo algortimo training_validation_sound_files().

3.1.2 Hardware

O hardware do sistema consistirá basicamente no computador para extração, processamento e classificação do sinal, e no sistema de fechadura.

Inicialmente, o sistema inicializará carregando o software Octave e o script realtimeaquisition_sounds. Quando o usuário apertar um dada tecla será feita a gravação de sua voz até o que mesmo botão seja pressionado novamente. A aquisição desse sinal será feita pelo microfone (sensor primário). O sinal é processado pelo algoritmo classificador escolhido (Raspberry) e, caso o usuário seja verificado (autorizado), na interface da raspberry será imprimida uma mensagem identificando qual é o locutor correspondente, caso o locutor não esteja no sistema, será impressa uma mensagem de usuário não identificado.

3.2 Metodologia Experimental

Para teste do sistema proposto, necessitamos de uma determinada quantidade de dados para treinamento e validação. Sinais de áudio de 8 locutores distintos (Matheus, Patrick, Sarah, Luiza, Lara, Mitsuo, Evaristo e Bruno) foram coletados e processados. Nesse teste, foi proposto a gravação de 6 frases diferentes, de maneira que fosse possível a extração dos coeficientes cepstrais para treinamento dos classificadores. Para cada locutor, foi solicitado 20 gravações de cada uma das 6 frases propostas, de forma que fosse possível treinar o sistema a verificar diferenças nas consecutivas pronuncias da mesma frase, considerando a variância no tempo do sinal produzido no trato vocal e no fenômeno de excitação que o precede. As locuções escolhidas para os testes foram: "Liga", "desliga", "aumente o volume", "diminua o volume", "abra a porta"e "tranque a porta". A escolha dessas locuções também está relacionada ao tempo que o ser humano leva na pronúncia

de cada uma, se tornando necessário treinar o sistema com locuções mais curtas (como "liga") e com mais demoradas (como "aumente o volume"), para uma maior efetividade em classificar sons com tais variâncias.

Para um treinamento efetivo, variar o gênero do locutor é interessante. Por questões fisiológicas, mulheres têm, no geral, características que possuem uma similaridade maior entre si do que comparado à uma voz masculina. O recíproco vale ao comparar duas vozes masculinas. Para o treinamento do sistema proposto, foram utilizadas locuções de 5 homens e 4 mulheres (com idade média de 24 anos), tornando possível agora uma boa distinção entre vozes masculinas e femininas entre si.

Para avaliar fatores como precisão e acurácia de nosso sistema, precisamos de subsequentes testes de validação. Tais testes devem ser feito de forma a não utilizar os mesmos dados de áudio para treinamento e classificação, pois geraria o mesmo resultado de indicadores como acurácia. Sabendo disso, a técnica utilizada para validação de nosso sistema é o *Holdout* com reamostragem aleatória. Tal teste propõe separar todos os dados em dois subconjuntos, sendo um para validação, e outro para teste. No caso do nosso sistema, para uma eventual validação em seu uso regular, poucos dados serão utilizados para validação, tendo em vista que o usuário irá dar um simples e curto comando de voz para que o sistema o reconheça. A ideia consiste então, em utilizar um conjunto de dados para treinamento maior do que o conjunto para validação.

Para um indicador de desempenho de melhor qualidade do nosso sistema, o *Holdout* deve ser feito várias vezes, com o objetivo de utilizar a maior variedade de combinações para treinamento e validação, gerando convergência para um resultado de acurácia. Para uma estimativa adequada de acurácia, precisão, sensibilidade e especificidade de nosso sistema, utilizamos o método *Holdout* com 500 iterações, um processamento que levou em média 4 horas de execução (O *K-Fold*, caso implementado, poderia oferecer o mesmo nível de resultado em menos tempo).

Cada iteração do método utilizado, nos retorna um valor de cada uma das métricas listadas para cada locutor. Sendo assim, uma média é tirada de cada locutor levando em conta cada uma das métricas. Dessa forma, temos uma estimativa de acurácia, precisão, sensibilidade e especificidade para cada um dos locutores, isso do primeiro classificador. Então, o mesmo teste é feito para os outros classificadores, as médias são calculadas, e no final temos as estimativas de todas as métricas, para cada classificador, considerando os locutores de forma separada. Para uma generalização do resultado final das métricas, é calculada uma média juntando todas as métricas dos locutores, tendo ao final apenas uma estimativa de cada uma das métricas para cada classificador (que está, ao certo, submetida a um erro um pouco maior).

3.3 Metodologia de Análise

3.3.1 Métodos de Validação Cruzada

Para verificar a efetividade de nosso sistema, necessitamos avaliar diferentes métricas de resultados. Antes disso, para avaliação dos resultados, testes (treinamento e classificação) subsequentes precisam ser feitos de forma a estimar melhor tais métricas. Existem algumas técnicas para tais testes serem realizados, trazendo diferentes estimativa da acurácia. As principais técnicas são o *Holdout* e o *K-Fold Cross Validation* [14].

O método *Holdout* consiste em separar uma maior parte das amostras (ou dados) para treinamento, e uma menor parte para classificação (geralmente 2/3 e 1/3, respectivamente [14]). Após uma iteração feita, temos um valor de acurácia que, contudo, é uma estimativa com uma taxa de erro significativa. Para diminuir tal erro, várias iterações são realizadas, trocando quais amostras serão usadas para treinamento e classificação de forma aleatória, sendo esta última ação chamada de reamostragem aleatória. Definimos então a acurácia final como sendo a média de todas as acurácias nesses subsequentes testes. Para uma boa estimativa utilizando esse método, devemos realizar um número de iterações significativamente alto, provocando um efeito de *trade-off* entre custo e tempo de processamento. Além disso, para este método, não garantimos que todas as amostras serão utilizadas para treinamento do classificador ou modelo, o que é um desperdício de amostras que poderiam trazer maior acurácia para o sistema.

O método *K-Fold Cross Validation* consiste em inicialmente separar todo o espaço amostral S em K conjuntos de tamanho S/K. Após isso, utilizaremos o primeiro conjunto para classificação, e o restante para treinamento. Realizamos então, o mesmo treinamento K vezes, sempre alternando o conjunto utilizado para classificação. Portanto, eliminamos o *trade off* entre custo e tempo de processamento, sendo possível realizar uma boa estimativa com menos iterações que o *Holdout* com reamostragem aleatória nos oferece. Além disso, garantimos que todas as amostras estão sendo utilizadas para treinamento [14].

Para uma boa estimativa das métricas de desempenho do nosso sistema, então, devemos utilizar o *K-Fold Cross Validation* ou o *Holdout* com um número alto de iterações.

3.3.2 Métricas de Desempenho

Os resultados provenientes de classificadores podem ter diferentes instâncias. No caso de reconhecimento de voz para controle de acesso, onde apenas uma pessoa é autorizada, teremos duas possíveis instâncias (a pessoa será autorizada, ou não). Para alguns outros modelos de classificação, podemos ter várias instâncias, como no caso de um sistema de reconhecimento facial que faz a leitura de um indivíduo, e diz se ele pertence ou não a um grupo, e caso pertença, qual o seu rótulo. Para todos esses casos, cada rotulação

que o sistema faz, podemos possuir quatro instância de avaliação dessas estimativas com a realidade. São elas os verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos [10].

Um rótulo verdadeiro positivo é aquele em que o classificador reconhece como autorizado alguém que deveria realmente ser autorizado, como no caso do controle de acesso. Da mesma forma, um rótulo verdadeiro negativo é quando a pessoa é classificada como não autorizada, sendo que ela realmente não é. Rótulos verdadeiros consistem então, nas classificações bem sucedidas de nosso sistema, ou seja, quando ele rotula corretamente pessoas não autorizadas e autorizadas, no exemplo explicado.

Utilizando ainda o mesmo caso do controle de acesso, um falso negativo é aquele em que um indivíduo é rotulado como não autorizado, enquanto na verdade ele deveria ser autorizado. Tal é o primeiro dos casos de erro do nosso classificador, sendo assim indesejável. Tal instância de rotulação, apesar de indesejável, pode ser tolerada em algumas aplicações. No exemplo dado, o indivíduo que foi classificado como não autorizado, enquanto deveria ser, necessita apenas realizar uma nova tentativa de classificação, e se o sistema tiver uma boa acurácia, ele conseguirá o acesso. Vale ressaltar que para determinadas aplicações, esse tipo de instância da rotulação pode ser totalmente indesejável.

O último tipo de rotulação, são os falsos positivos. Eles consistem no tipo de rotulação em que o indivíduo é classificado como autorizado, quando na realidade não deveria ser. No nosso caso utilizado como exemplo (reconhecimento de voz para controle de acesso), tal instância de rotulação é totalmente indesejável e deve ser reduzida ao mínimo, pois pode representar uma séria falha de segurança do sistema. Esses então, são as instâncias de rotulação em que nosso sistema erra a classificação, sendo que alguns casos são toleráveis pela aplicação, e em outros não.

Levando em consideração tais avaliações dos resultados do classificador, são estabelecidas métricas para avaliação dos resultados, que são: Precisão, acurácia, sensibilidade e sensitividade.

A precisão avalia quantas vezes o sistema realizou classificações positivas corretamente. Sendo assim, ela é definida pela seguinte fórmula [10]:

$$Precisao = \frac{VerdadeirosPositivos}{VerdadeirosPositivos + FalsosPositivos} \tag{3.1}$$

A acurácia avalia quantas vezes o sistema realizou qualquer classificação correta dentre os testes, seja ela positiva ou negativa. Sendo assim, ela é definida pela seguinte fórmula [10]:

$$Acuracia = \frac{VerdadeirosPositivos + VerdadeirosNegativos}{TotalPositivos + TotalNegativos} \tag{3.2}$$

A sensibilidade, avalia quantas classificações positivas corretas foram feitas dentre todas as que deveriam ser realmente classificadas como positiva. Sendo assim, ela é

definida pela seguinte fórmula [10]:

$$Sensibilidade = \frac{VerdadeirosPositivos}{TotalPositivos} \tag{3.3}$$

A especificidade é, para a aplicação proposta, a métrica de avaliação mais importante, pois leva em conta o que queremos evitar, os falsos positivos. Basicamente, ela avalia quantas classificações negativas foram feitas corretamente, dentro de um conjunto que também leva em conta os negativos avaliados incorretamente. A especificidade é então, o oposto da taxa de falsos positivos (deve ser a menor possível para uma alta especificidade). Definimos a taxa de falsos positivos como [10]:

$$TaxaFalsosPositivos = \frac{FalsosPositivos}{TotalNegativos} \tag{3.4}$$

Sendo assim, a especificidade é definida pela seguinte fórmula [10]:

$$Especificidade = \frac{VerdadeirosNegativos}{FalsosPositivos + VerdadeirosNegativos} = 1 - TaxaFalsosPositivos \tag{3.5}$$

4 Resultados e discussões

Foi feita a extração de 20 coeficientes cepstrais (LPCC), pois é um valor aceitável para caracterização de usuário [16], para cada uma das janelas de 300ms de cada arquivo de som gravado.

Para a coleta dos dados de validação do sistema, o método *Holdout* foi aplicado 500 vezes, um número arbitrário mas suficientemente grande para obter uma boa aproximação das métricas de acurácia, precisão, especificidade e sensibilidade (assim como poderia ser feito com menos iterações aplicando o método *k-fold*).

Após a coleta dos dados, foi tirada a média das 500 medições de cada métrica. Cada tabela de resultados é referente a um classificador, onde para cada um é mostrado o resultado de reconhecimento de 8 locutores. O primeiro classificador a ser analisado foi a SVM com *kernel* RBF com $\sigma = 1$. Suas medidas estão na tabela 4.1 :

(%)	**Acurácia**	**Precisão**	**Sensibilidade**	**Especificidade**
Locutor 1	99,22	99,98	95,69	99,99
Locutor 2	99,21	92,26	96,96	99,37
Locutor 3	99,53	99,91	96,51	99,98
Locutor 4	99,82	99,77	99,42	99,93
Locutor 5	95,13	90,91	70,1	98,9
Locutor 6	93,72	79,4	58,81	98,05
Locutor 7	97,42	87,73	80,59	98,94
Locutor 8	92,94	77,26	16,69	99,51

Tabela 4.1. Resultados das métricas dos 8 locutores para o classificador SVM.

As medidas finais de cada locutor do classificador Robustboost são mostradas na tabela 4.2 considerando $\epsilon = 0.1$, $\sigma_f = 0.1$ e $\theta = 0$ com 100 iterações pré-definidas e árvore de decisão como classificador fraco.

(%)	**Acurácia**	**Precisão**	**Sensibilidade**	**Especificidade**
Locutor 1	97,49	93,03	94,25	98,2
Locutor 2	93,1	0	0	100
Locutor 3	91,66	66,37	77,86	93,74
Locutor 4	98,86	97,33	97,51	99,24
Locutor 5	90,29	78,91	37,92	98,19
Locutor 6	90,45	0	33,86	97,47
Locutor 7	91,72	0	0	100
Locutor 8	92,07	0	0	100

Tabela 4.2. Resultados das métricas dos 8 locutores para o classificador Robustboost.

Obs: Os valores 0 na tabela foram colocados, pois o classificador não conseguiu gerar

Verdadeiros Positivos e Falsos Positivos, gerando assim divisões de 0 por 0, invalidando as medidas de precisão e sensibilidade para alguns locutores no Robustboost

As medidas finais de cada locutor do classificador Adaboost com 100 iterações e árvore de decisão como classificador fraco, são mostradas na tabela 4.3 os resultados obtidos :

(%)	Acurácia	Precisão	Sensibilidade	Especificidade
Locutor 1	99,52	99,7	97,64	99,93
Locutor 2	98,95	96,11	88,62	99,72
Locutor 3	99,17	98	95,69	99,7
Locutor 4	99,41	98,46	98,86	99,57
Locutor 5	94,23	82,67	71,24	97,7
Locutor 6	94,36	76,77	70,91	97,27
Locutor 7	98,25	92,23	86,4	99,32
Locutor 8	93,4	63,99	40,43	97,96

Tabela 4.3. Resultados das métricas dos 8 locutores para o classificador Adaboost.

As medidas finais de cada locutor do classificador Bagging em modo de classificação usando árvore de decisão como classificador fraco e 100 classificações independentes, são mostradas na tabela 4.4 :

(%)	Acurácia	Precisão	Sensibilidade	Especificidade
Locutor 1	99,39	99,42	97,16	99,87
Locutor 2	97,79	99,73	68,17	99,99
Locutor 3	98,38	97,11	90,42	99,58
Locutor 4	99,41	98,95	98,35	99,7
Locutor 5	95,45	93,29	70,47	99,22
Locutor 6	95,24	87,75	66,53	98,8
Locutor 7	98,07	96,94	79,4	99,76
Locutor 8	94,13	82,68	33,48	99,35

Tabela 4.4. Resultados das métricas dos 8 locutores para o classificador Bagging

Por fim, foi feita uma tabela com as médias de todos os locutores (analisando as métricas citadas) para cada classificador, mostrada na tabela 4.5:

(%)	SVM	Robustboost	Adaboost	Bag
Acurácia	97,12	93,20	97,16	97,23
Precisão	90,90	41,95	88,49	94,48
Sensibilidade	76,84	42,67	81,22	75,49
Especificidade	99,33	98,35	98,89	99,53

Tabela 4.5. Resultados das métricas das médias dos 8 locutores, para cada classificador.

Podemos então, analisar graficamente os dados gerais (considerando todos os classificadores e todas as métricas), na figura Figura 4.1.

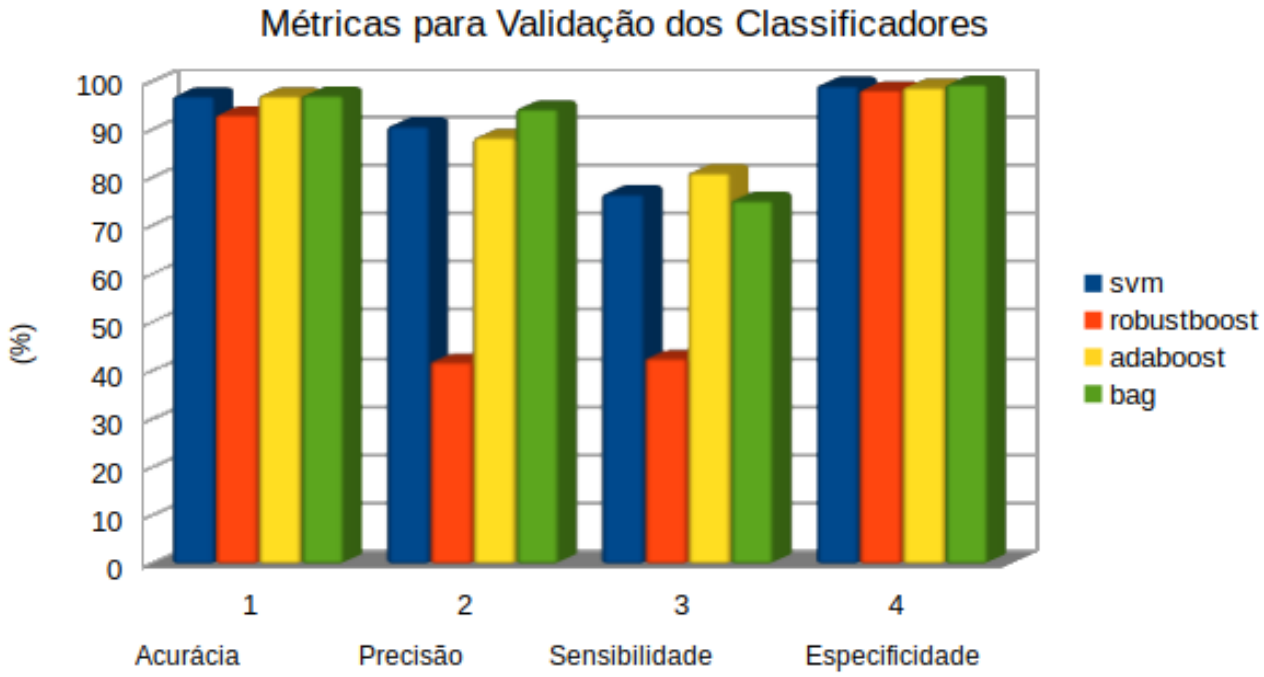

Figura 4.1. Representação gráfica das métricas para os classificadores.

Analisando os resultados, temos quatro métrica a serem avaliadas. Para escolhermos o classificador com melhor desempenho, devemos primeiramente entender os requisitos da aplicação. Para o caso de uma aplicação voltado a um sistema de segurança, desejamos primordialmente que uma pessoa não autorizada nunca acesse o sistema. Esse requisito se torna mais importante comparado a uma falha quando o indivíduo autorizado não consegue acessar o sistema, pois nesse último caso, a validação deve ser feita novamente. Uma analogia semelhante pode ser vista no reconhecimento biométrico para desbloqueio do teclado de um *Smartphone* nos dias atuais, pois muitas vezes o sistema não consegue reconhecer a digital do indivíduo autorizado, tendo este que realizar novamente a tentativa de desbloqueio, ao passo que uma digital não autorizada tem uma chance de conseguir desbloquear o aparelho extremamente baixa.

Tomando isso como pressuposto, desejamos que a nossa taxa de falsos positivos (indivíduo não autorizado classificado como autorizado) seja a menor possível, para que a segurança seja a mais efetiva possível. Sendo assim, a métrica de desempenho que melhor avalia isso é a especificidade. Em um sistema hipotético, onde a taxa de falsos positivos é zero, a especificidade será 100%.

Comparando os dados mostrados pelo gráfico Figura 4.1, vemos que todos os classificadores apresentados demonstraram um resultado satisfatório de especificidade, medida mais importante para a aplicação avaliada, sendo que nenhum ficou abaixo de 98% (na média). Os classificadores que demonstraram maior desempenho nessa métrica foram o Bagging (com 99,53%) e a SVM (99,33%).

Outra métrica importante a ser avaliada (não mais que a especificidade) é a acurácia. Ela mede o quanto que o sistema acertou, positivamente ou negativamente, dentre todos os testes realizados. Analisando o gráfico, temos resultados de acurácia muito próximos para a SVM(97,12%), o Adaboost (97,16%) e o Bagging (97,23%), ao passo que o Robustboost apresentou uma taxa menor de acurácia de 93,20%. Isso significa que, por algum motivo, tal classificador apresentou uma taxa de erro maior nas classificações feitas, porém, essa taxa não foi toda resultante de falsos positivos, pois temos um valor satisfatório de especificidade. Um possível motivo para esse valor reduzido de acurácia, é a baixa sensibilidade do sistema, como pode ser observado no gráfico Figura 4.1. Essa falta de sensibilidade não causa apenas um aumento da taxa de falsos positivos, mas sim, de falsos negativos, o que torna o sistema lento, podendo levar o usuário a realizar a validação repetidas vezes.

Um ponto importante a ser citado, é que um locutor foi removido dos resultados do presente trabalho. Foi solicitado a irmã de uma das locutoras (Clara) realizasse as gravações para teste, e os classificadores apresentaram resultados ruins analisando as métricas apresentadas. Isso se deve ao fato das duas vozes serem extremamente parecidas, causando uma dificultade de distinção do sistema. As gravações dessa locutora foram removidas para o não comprometimento dos resultados. Entende-se, que nem o próprio ser humano consegue distinguir com clareza as vozes dessas locutoras (foi solicitado que a própria irmã reconhecesse, e ela não conseguiu com clareza).

Após essa análise, foi usada uma célula com 8 svms gravadas pelo MATLAB, em um sistema de reconhecimento de locutor no programa Octave da Raspberry Pi 3, para criação de um simples protótipo aplicando um modelo de aprendizagem de máquina.

5 Conclusão

De acordo com os resultados obtidos, conclui-se que como na grande maioria dos casos da engenharia, devemos analisar primeiramente a aplicação do sistema a ser implantado. Para o caso de uma aplicação em segurança, o que realmente nos interessa é que a taxa de falsos positivos seja a menor possível, sendo as outras métricas também importantes, mas não comprometem o real objetivo considerando a aplicação. Sendo assim, analisamos a especificidade, que teve um resultado bom para todos os classificadores.

Assim, analisamos as outras métricas para escolha de nosso classificador. Conclui-se então que os que apresentaram melhores resultados considerando o requisito principal, foram a SVM e o Bagging. Dentre eles o Bagging apresentou resultados um pouco melhores.

Apesar de a especificidade do sistema ser alta, e a taxa de falsos positivos baixa, ela ainda existe. Uma forma de contornar isso, é exatamente a que vemos nos dias atuais, como na aplicação do *Smartphone* mencionada. Quando um indivíduo realiza três tentativas de acesso, onde ele é não autorizado, o sistema bloqueia a tentativa de acesso por um tempo. Isso reduz consideravelmente a chance de um indivíduo não autorizado acessar o sistema. Contudo, deve-se avaliar também métricas como a sensibilidade do sistema, e a taxa de falsos negativos, pois um indivíduo autorizado pode realizar bloqueia indesejado do sistema após ser classificado como não autorizado repetidas vezes.

Como já dito anteriormente, os resultados obtidos com as classificações foram satisfatórios, no entanto esses resultados foram obtidos utilizando somente as vozes captadas, os testes utilizando vozes de pessoas não usadas no treinamento mostrou que os classificadores não apresentam uma boa especificidade para vozes não usadas no treinamento, já que nos testes feitos com vozes diferentes das treinadas, ocorreu do classificador rotular erroneamente uma voz fora do sistema, como alguem inserido nele, o que inviabiliza o uso do sistema proposto na área de controle de acesso, no entanto, ainda pode-se fazer uso dele para uma aplicação que necessite apenas do reconhecimento de voz dentro de um grupo pré-cadastrado.

Um possível avanço desse trabalho seria a combinação da classificação de locutor já desenvolvida com uma classificação de locução, criando um sistema que além de identificar o locutor, em paralelo faz a classificação da locução utilizada.

Referências Bibliográficas

[1] G. Aggarwal e L. Singh. Characterization between child and adult voice using machine learning algorithm. In *International Conference on Computing, Communication Automation*, páginas 246–250, May 2015.

[2] Pedro Barros. Aprendizagem de máquina: Supervisionada ou não supervisionada?, 2016.

[3] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] J. P. Campbell. Speaker recognition: a tutorial. *Proceedings of the IEEE*, 85(9):1437–1462, Sep 1997.

[5] D. G. Childers, D. P. Skinner, e R. C. Kemerait. The cepstrum: A guide to processing. *Proceedings of the IEEE*, 65(10):1428–1443, Oct 1977.

[6] C. Cortes e V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[7] M. Marufo da Silva, D. A. Evin, e S. Verrastro. Speaker-independent embedded speech recognition using hidden markov models. In *2016 IEEE Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI)*, páginas 1–6, Nov 2016.

[8] B Efron e R Tibshirani. An Introduction to the Bootstrap. 1993.

[9] S. D. Essinger e G. L. Rosen. An introduction to machine learning for students in secondary education. In *2011 Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE)*, páginas 243–248, Jan 2011.

[10] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[11] Yoav Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3):293–318, 2001.

[12] Yoav Freund. A more robust boosting algorithm. 2009.

[13] Yoav Freund e Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. páginas 23–37, 1995.

[14] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, páginas 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[15] L. Lei e K. She. Speaker recognition on mobile phone: Using wavelet, cepstral coefficients and probabilisitc neural network. In *2016 13th International Conference on Embedded Software and Systems (ICESS)*, páginas 137–142, Aug 2016.

[16] Jinyu Li, Li Deng, Reinhold Haeb-Umbach, e Yifan Gong. *Fundamentals of speech recognition*. 1993.

[17] L. Lu, L. Liu, M. J. Hussain, e Y. Liu. I sense you by breath: Speaker recognition via breath biometrics. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1, 2017.

[18] Marcos Paulo Barros Oliveira. Verificação automática de locutor, dependente do texto, utilizando sistemas híbridos mlp/hmm. páginas 18–19, 2001.

[19] Evandro David Silva Paranaguá.

[20] F. E. Petry. Speech recognition: A current perspective: In spite of limitations, areas of application are growing, and voice communication with computers may well be commonplace by the 21st century. *IEEE Potentials*, 2(Spring):18–20, First 1983.

[21] S. Raghavan, G. Lazarou, e J. Picone. Speaker verification using support vector machines. In *Proceedings of the IEEE SoutheastCon 2006*, páginas 188–191, March 2006.

[22] A. V. Rao e K. Rose. Deterministically annealed design of hidden markov model speech recognizers. *IEEE Transactions on Speech and Audio Processing*, 9(2):111–126, Feb 2001.

[23] R. M. Santos, L. N. Matos, H. T. Macedo, e J. Montalvão. Speech recognition in noisy environments with convolutional neural networks. In *2015 Brazilian Conference on Intelligent Systems (BRACIS)*, páginas 175–179, Nov 2015.

[24] Robert E Schapire. The Strength of Weak Learnability (Extended Abstract). *Machine learning*, 227(October):28–33, 1989.

[25] S. Kotz V. Vapnik. *Estimation of dependences based on empirical data*. Springer series in statistics. Springer-Verlag, 1982.

[26] C Van Der Malsburg. *Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, páginas 245–248. Springer Berlin Heidelberg, Berlin, Heidelberg, 1986.

[27] Zhi-hua Zhou. *Ensemble Methods*. 2014.