UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

João Guilherme de Oliveira Júnior

Avaliação de métodos de aprendizado de máquina supervisionado aplicados a análise de sentimento de voz

Recife

2019

João Guilherme de Oliveira Júnior

Avaliação de métodos de aprendizado de máquina supervisionado aplicados a análise de sentimento de voz

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Estatística e Informática

Curso de Bacharelado em Sistemas de Informação

Orientador: Rodrigo Gabriel Ferreira Soares

Recife

2019

Dedico este trabalho a minha mãe Maria Dalva e ao meu pai João Guilherme de Oliveira.

“..Há três métodos para ganhar sabedoria: primeiro, por reflexão, que é o mais nobre; segundo, por imitação, que é o mais fácil; e terceiro, por experiência, que é o mais amargo.”

(Confucio)

Resumo

Cada vez mais estamos interagindo com agentes virtuais através do uso da voz, porém uma boa comunicação não se resume apenas a compreensão das palavras ditas e pode incluir aspectos não textuais como variações no tom e na intensidade da voz que podem estar relacionadas a elementos presentes no contexto de uma conversação. Assim sendo para melhor eficácia destas tecnologias, um agente virtual precisa ser capaz de entender estes sinais presentes na voz e estimar o estado emocional do usuário a fim de ser capaz de interpretar e responder mais adequadamente aos seus estímulos.

Neste trabalho propomos um modelo baseado em aprendizado de máquina para análise e classificação do estado emocional de um locutor baseada nos sinais presentes na sua voz. Serão comparadas a performance em termos de precisão dos seguintes algoritmos de aprendizado de máquina: Naive Bayes, árvores de decisão, máquinas de vetor suporte, multi layer perceptron e redes convolucionais.

Resultados comparáveis a de jurados especializados puderam ser alcançados com estes modelos onde foi possível distinguir as seguintes emoções: ”alegria”, ”surpresa”, ”desgosto”, ”neutro”, ”medo”, ”raiva”e ”tristeza”.

Palavras-chave: Aprendizado de máquina, reconhecimento de fala, análise de sentimento.

Abstract

Increasingly we are interacting with virtual agents through the use of voice, but good communication is not only a matter of understanding the words and may include non-textual aspects such as variations in the tone and intensity of the voice that may be related to elements present in the context of a conversation. Thus, for a better efficacy of these technologies, a virtual agent needs to be able to understand these signals present in the voice and estimate the user's emotional state in order to be able to interpret and respond more appropriately to their stimuli.

In this work we propose a model based on machine learning to analyze and classify the emotional state of a speaker based on the signals present in his voice. The performance in terms of the accuracy of the following machine learning algorithms will be compared: Naive Bayes, decision trees, support vector machines, multi layer perceptron and convolutional networks.

Comparable results of specialized judges could be achieved with these models where it was possible to distinguish the following emotions: ”happiness”, ”surprise”, ”disgust”, ”neutral”, ”fear”, ”anger” and ”sadness”.

Keywords: Machine learning, speech recognition, sentiment analysis.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AD	Árvore de decisão
ADPCM	Adaptive differential pulse code modulation
AM	Aprendizado de máquina
API	Application program interface
CV	Computer vision
DPCM	Differential pulse code modulation
FP	False positive
GPU	Graphical processing unit
HCI	Human-computer interaction
MFCC	Mel Frequency Cepstral Coefficients
MLP	Multi layer perceptron
MRE	Mínimo risco estrutural
NB	Naive bayes
PCM	Pulse code modulation
SER	Speech emotion recognition
SR	Speech recognition
SVM	Support vector machine
TP	True positive

Sumário

1 Introdução

O reconhecimento de fala (*Speech recognition* ou SR) é um dos campos de estudos mais ativo na área de aprendizado de máquina (AM) (GRAVES; JAITLY, 2014) (NAKAGAWA, 2002) (RUDNICKY; HAUPTMANN; LEE, 1994). O reconhecimento de emoções na locução (*Speech emotion recognition* ou SER) é uma crescente e interessante vertente de pesquisa em reconhecimentos de padrões e interface homem-máquina (*Human-computer interaction* ou HCI), sendo o tema central de muitos artigos e estudos (AYADI; KAMEL; KARRAY, 2011) (ZENG et al., 2008) (WU; FALK; CHAN, 2011).

Devido à crescente adoção e utilização destas tecnologias novos desenvolvimentos devem ocorrer no sentido de tornar esta interface ainda mais natural e eficiente.

Os sistemas atualmente propostos como a Siri[1] e o Google assistant[2] são dimensionados para capturar muito bem as características verbais da fala, ou seja, encontrar as palavras e a sequência correta que ocorrem em um discurso. Mas parte do que é expresso pelo locutor se perde nesse modelo, são as características não-verbais como entonação, intensidade, modulação e velocidade. Informalmente denominadas de forma conjunta como ”tom da voz”[3]. Características estas que podem adicionar sentido e até mudar a compreensão do que é dito por um locutor.

Em marketing e comunicação elementos não-verbais da voz são desenvolvidos[4] na oratória de figuras de liderança como políticos e outros a fim de transimitir empatia, liderança e outras emoções que são relacionadas a um maior êxito em suas atividades.

Em um estudo publicado na revista Evolution and Human Behaviour (COWAN et al., 2016) pesquisadores relacionaram o tom de voz na transmissão de humor.

Procedimentos na área da psicologia utilizam estes elementos não-verbais para diagnóstico do estado mental e emocional de pacientes, como em (CORDIOLI; ZIMMERMANN; KESSLER, 2012).

Um estudo em linguística e prosódia (MILAN; KLUGE, 2017) demonstrou a relação de aspectos como a frequência fundamental na diferenciação de uma afirmação e de uma interrogativa em locutores da língua portuguesa.

1 https://www.apple.com/siri/
2 https://assistant.google.com/
3 https://maisexpressao.info/tom-de-voz-o-que-vem-a-ser-isso/
4 http://rdplanalto.com/noticias/1426/a-importancia-da-voz-no-processo-de-comunicacao

Pode-se perceber pela quantidade e variedade de estudos existentes que existe uma boa parte da comunicação que ocorre por meios não-verbais, assim sendo investigaremos a utilização de aprendizado de máquina na classificação do estado emocional durante a fala com foco nos locutores da língua portuguesa e suas particularidades. De modo que com esta nova informação novas soluções possam permitir maiores e melhores avanços na área de análise de sentimentos (SER).

1.1 Motivação e Justificativa

Cada vez mais estamos interagindo com diversos dispositivos e serviços apenas com o uso da voz. Isto se deve aos recentes avanços na área de reconhecimento de fala que possibilitaram sua utilização de forma mais simples e o com maior precisão.

Serviços como Siri[5] e o agente de voz da google[6] irão se tornar ainda mais presentes na nossa vida, segundo apontam alguns especialistas[7]. As grandes empresas do mercado estão adotando uma estratégia de agregar cada usuário potencial na sua base de clientes fazendo-os consumidores de seu ecossistema de produtos. Com a popularização de smartphones e a recente popularização de agentes virtuais esse processo tem acelerado permitindo a integração de múltiplos serviços de forma natural e transparente aos usuários.

Assim sendo, para interagir de uma forma mais natural com o usuário estes sistemas devem ser capazes de entender cada vez mais elementos presentes na conversação e de modo transparente permitir a personalização da experiência sem maiores esforços[8].

O tom da voz durante uma fala pode adicionar mais significado às palavras nela contida. Em uma conversação, a voz é responsável por representar uma gama extensa de aspectos não-verbais relevantes a conversa, às vezes profundamente conectada ao contexto ou estado do locutor.

Por exemplo, é a partir do tom da voz que podemos estimar o estado emocional do par na conversa (COOK, 2002). Como também é pelo tom de voz que diferenciamos uma afirmação de um questionamento (MILAN; KLUGE, 2017), mesmo se eles forem compostos exatamente das mesmas palavras e na mesma ordem.

Esta informação não-verbal extra é essencial para a correta compreensão do que está sendo dito, como no caso do sarcasmo, em que o que se entende é o oposto do que se fala. Intuitivamente entendemos que existe um ”tom de voz” sarcástico

[5] https://www.apple.com/siri/
[6] https://assistant.google.com/
[7] https://www.computerworld.com/article/3252259/virtual-assistant-battle.html
[8] https://medium.com/syncedreview/augmenting-virtual-assistants-with-personality-and-personalization-f5395707d349

mesmo não sabendo definir exatamente ele, como também sabemos a diferença entre uma pergunta legítima e uma pergunta retórica, aquela que não é seguida de uma resposta.

Portanto seria de muita valia que os agentes virtuais pudessem compreender não apenas os aspectos verbais mais também alguns aspectos não-verbais podendo então fornecer respostas mais adequadas aos estímulos dos seus usuários.

1.2 Objetivos

O objetivo é produzir um sistema que possa identificar o estado emocional do locutor durante sua fala, podendo classificá-la como uma das seguintes emoções: ”alegria”, ”surpresa”, ”desgosto”, ”neutro”, ”medo”, ”raiva”e ”tristeza”. As seis emoções descritas no modelo de (RUSSELL, 1980) e mais um estado neutro foram adicionados ao modelo a fim de incluir um estado de normalidade (tranquilidade).

De modo que se torna necessária a comparação de diversos algoritmos de AM a fim de se concluir qual é o mais adequado para a tarefa. Os seguintes algoritmos de aprendizado de máquina serão avaliados para este fim, a saber: Naive Bayes, redes neurais (Multi-Layer Perceptron), Support Vector Machines (SVM), Árvores de decisão (Decision Tree) e redes convolucionais (ConvNets).

Será realizado um experimento onde um banco de dados, contendo diversos clipes de áudio com emoções distintas serão utilizados para o treinamento e teste de cada algoritmo. Seus resultados serão comparadas aos pares por um teste de hipótese e ao fim estabeleceremos com alguma confiança estatística qual algoritmo teve a melhor performance neste problema.

1.3 Contribuições

A principal contribuição deste trabalho é obter um modelo que possibilite a análise de sentimentos de clipes de áudio classificando-os dentre as as emoções enumeradas na seção anterior, bem como detalhar o pré-processamento das as amostras a partir das gravações.

Uma contribuição menor deste experimento é o estudo utilizando um banco de dados anotados com falas atuadas em português do Brasil, o que vai permitir que uma análise desta natureza seja conduzido na nossa língua.

Por último, este trabalho demonstra o emprego de uma rede neural convolucional (Convnets) na tarefa de classificação de áudio. Previamente as redes convolucionais foram utilizadas com muito sucesso em problemas de visão computacional

(*Computer Vision* ou CV) como em (SIMONYAN; ZISSERMAN, 2014). Estudos mais recentes, como (NASICHUDDIN; ADJI; WIDYAWAN, 2018) e (LI; ZHOU, 2017), apresentam cenários onde redes convolucionais foram empregadas em outros contextos como processamento de texto e de áudio e demonstraram resultados muito promissores o que motivou a sua inclusão neste estudo.

O *layout* escolhido para a rede convolucional é inspirado no *design* da VGGNET (SIMONYAN; ZISSERMAN, 2014) e no trabalho de (ZHANG et al., 2017) e vai contar com no máximo duas camadas convolucionais. Todas as operações de convolução serão realizadas com janelas de kernel de tamanho 3x3 como de costume em redes inspiradas na VGGNET. Na primeira camada convolucional, logo após a operação de convolução será aplicada a operação de *maxpooling*, como usualmente ocorre em redes convolucionais. Na segunda camada convolucional todas as operações de convolução serão aplicadas diretamente uma sobre a outra, de modo a replicar as características descritas no trabalho de (ZHANG et al., 2017), possibilitando que o modelo aprenda representações das variações temporais presentes em cada classe do problema a ser investigado.

1.4 Organização deste trabalho

Este trabalho está estruturado da seguinte forma:

- Uma visão geral do problema pode ser encontrado no Capítulo 2;
- As metodologias adotadas durante o pré-processamento podem ser vistas no Capítulo 3;
- O Capítulo 4 discute os métodos de aprendizado de máquina e os algoritmos que serão avaliados mais profundamente;
- No Capítulo 5 todo o fluxo do experimento é detalhado até a obtenção dos resultados;
- O Capítulo 6 contém a análise dos resultados obtidos;
- No Capítulo 7 são apresentados algumas considerações finais e possíveis trabalhos futuros.

2 Análise de sentimentos da voz

A necessidade de se compreender os diferentes estados emocionais motivou diversos estudos nas áreas ligadas a psicologia e ciências sociais, onde algumas propostas de organização ou estruturação podem ser encontradas, como nos trabalhos de (RUSSELL, 1980) e (SCHERER, 2005).

Uma vez que o objetivo deste estudo é produzir um sistema capaz de categorizar as emoções, a adoção de alguma estrutura para o modelo proposto deve basear-se em uma sólida base científica. Os modelos acima relacionados possuem uma organização de fácil compreensão, porém o funcionamento destes modelos não interessa ao propósito deste estudo, uma vez que novos modelos serão obtidos por aprendizado de máquina.

2.1 Um modelo para classificar emoções

Em (RUSSELL, 1980), um plano é proposto em que dois eixos (energia e valência) descrevem as emoções básicas, este modelo está ilustrado na Figura 1.

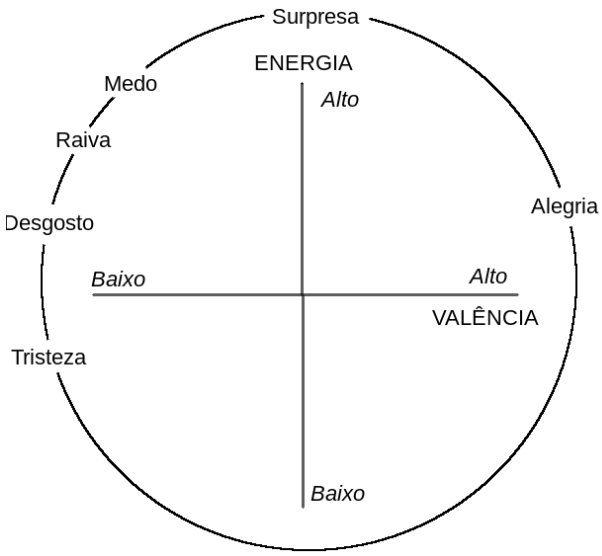


Figura 1 – Modelo proposto por J. Russell em 1980.
Fonte: Adaptado de (RUSSELL, 1980).

Em comparação com o modelo proposto por (SCHERER, 2005), este modelo é consideravelmente mais simples e foi escolhido por ser bem estabelecido contando com diversos estudos que comprovam a sua eficácia, como o trabalho de (EKMAN, 1992).

Pode-se observar que neste modelo existe um conjunto de emoções que estão mais próximas, são elas: ”medo”, ”raiva”, ”desgosto” e ”tristeza”. Percebe-se que existe uma distância considerável entre as emoções ”alegria” e ”tristeza”, assim no modelo proposto optou-se por incluir um rótulo adicional ”neutro” que ocupa um ponto entre estas duas emoções.

Assim cada amostra pertencente ao conjunto de dados anotados devem constar como um dos seguintes rótulos: ”alegria”, ”desgosto”, ”medo”, ”neutro”, ”raiva”, ”surpresa” e ”tristeza”.

2.2 Experimento com jurados

Durante a criação do conjunto de dados anotados adotado neste estudo, foi realizado um experimento por (NETO et al., 2018) afim de validar o conjunto de dados proposto. Para fins de avaliação três jurados especialistas foram consultados para rotular as amostras e demonstraram uma divergência significativa sobre o rótulo de algumas instâncias, como pode ser visualizado na Figura 2. Emoções como surpresa e desgosto mostraram ser as maiores fontes de discordância entre os jurados.

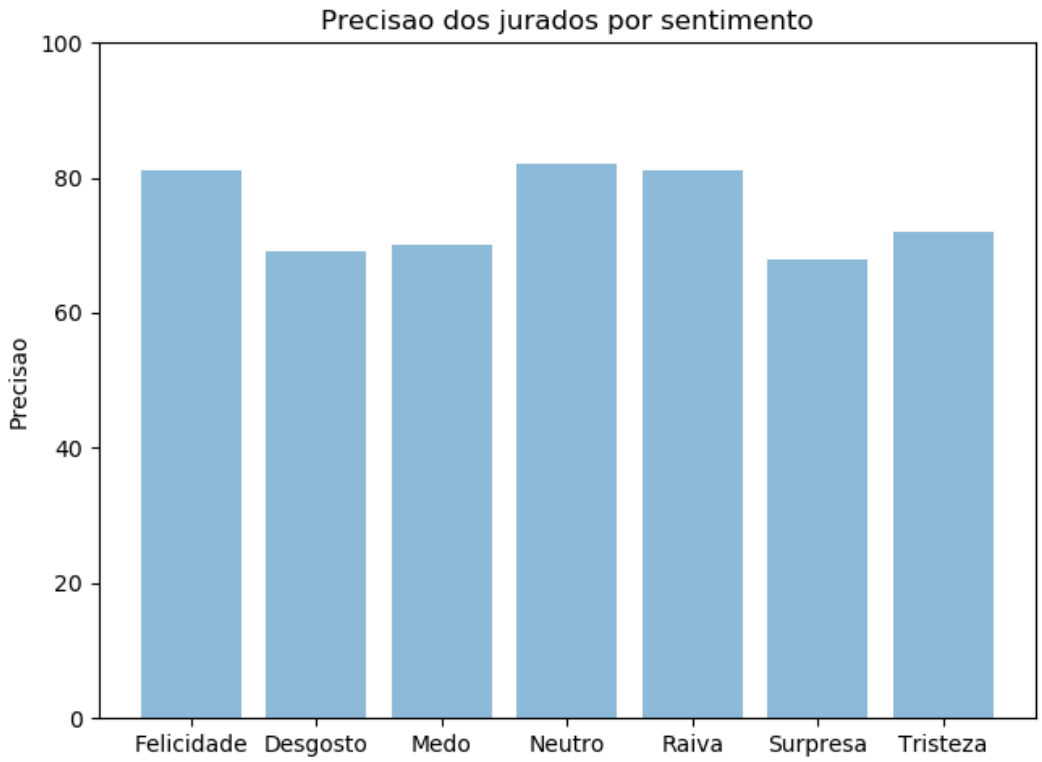
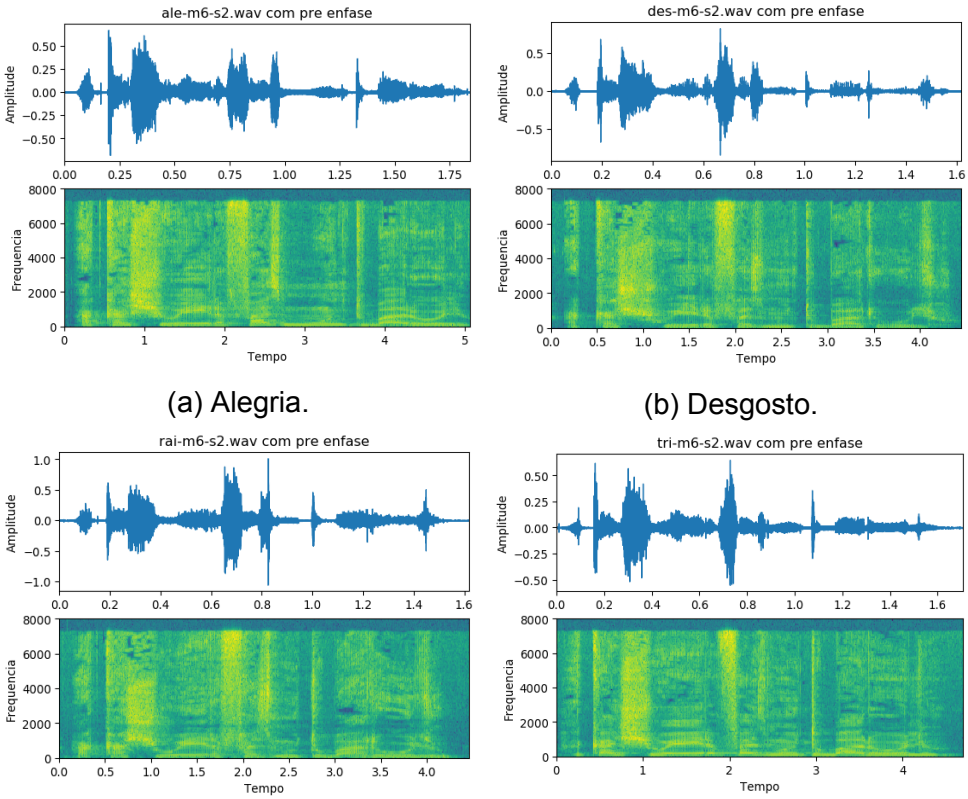


Figura 2 – Resultados dos jurados.
Fonte: Adaptado de (NETO et al., 2018).

Intuitivamente reconhecemos algumas emoções com facilidade, como é o caso de alegria e tristeza, outras vezes algumas expressões podem soar muito parecidas como na alegria e na raiva. Podemos observar estas similaridades e divergências também na representação temporal e no domínio das frequências, o que pode ser visualizado na Figura 3.



(a) Alegria.

(b) Desgosto.

(c) Raiva.

(d) Tristeza.

Figura 3 – Representação do som da mesma sentença em quatro emoções distintas.

Isto ilustra como este é um problema difícil a ponto de confundir até as percepções humanas. O que torna oportuno a investigação com algoritmos de AM, como é o caso do experimento realizado por (WU; FALK; CHAN, 2011) possuindo como base a língua alemã.

3 Metodologia

Neste Capítulo são abordados os métodos utilizados para a obtenção das amostras a partir dos clipes de áudio, detalhando o processo de conversão do som em uma representação discreta e a transformação destes dados para uma representação que permita a extração dos atributos espectrais que ao final serão utilizados pelos algoritmos de aprendizado de máquina.

3.1 Representação matemática do som

O som é um sinal complexo composto de muitos outros sinais mais simples e que variam em volume, tom, duração e etc. A física clássica define o som como ondas que se propagam no espaço que possuem uma freqûencia e uma amplitude, como consta na equação do movimento harmônico simples:

$$x = A \cdot cos(2\pi f t + \theta_0), \tag{3.1}$$

onde x é a posição (nível) do sinal no tempo t, A é a amplitude do sinal, f é a frequência do sinal em hertz e θ_0 é a fase (deslocamento) inicial.

Sinais sonoros mais simples se combinam e podem formar sinais mais complexos, como ilustrado na Figura 4. Sinais senoidais podem ser somados a fim de formar várias formas de onda, inclusive quadrada como pode ser visto na Figura 5. Esta característica também é fundamental para a decomposição de um sinal utilizando a transformada de fourier que será empregada mais a frente, já que qualquer forma de onda pode ser representada como uma composição de infinitas funções seno e cosseno (série de fourier).

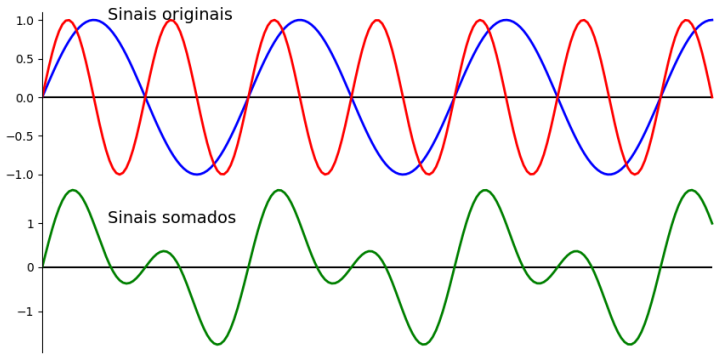


Figura 4 – Composição de um sinal sonoro a partir de sinais mais simples.

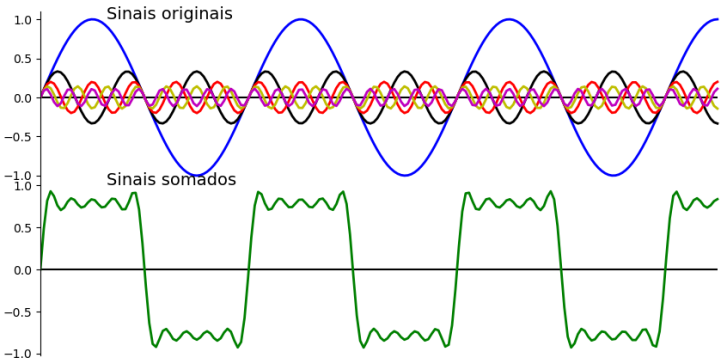


Figura 5 – Composição de um sinal com forma de onda quadrada a partir de senóides.

3.1.1 PCM

Um processo conhecido como PCM (*pulse code modulation*) é aplicado ao som a fim de transformá-lo em uma representação discreta, esta transformação é realizada durante o processo de gravação do som pelo dispositivo gravador. Uma ilustração de como PCM é composto pode ser visto a na Figura 6.

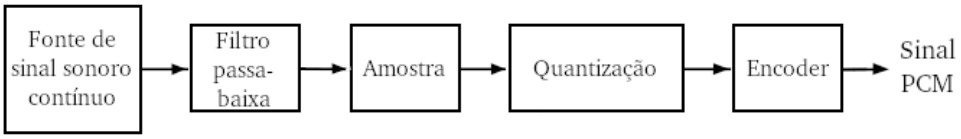


Figura 6 – Diagrama em blocos do PCM.

Pode-se observar que cada operação do PCM é realizada após o término de uma operação anterior e que recebe a saída desta operação como sua entrada. Em uma visão geral o PCM converte o som em uma representação filtrada de pontos discretos tanto no aspecto das amplitudes como no tempo e ao fim codificando-os para a reprodução como um arquivo digital.

Primeiramente o sinal de entrada é filtrado por um filtro passa baixa. Este filtro tem a finalidade de impedir ruídos de alta frequência que atrapalham o processo de amostragem de um sinal de baixa frequência, efeito conhecido como *aliasing*.

Um filtro passa baixa é um circuito eletrônico passivo que tem a finalidade de desviar e anular os componentes de maior frequência de um sinal, empregando basicamente um resistor e capacitor calculados para atuar apenas quando um sinal acima da frequência de corte ocorre, sua aparência pode ser vista na Figura 7. A frequência de corte f_c do filtro é determinada pelo seguinte cálculo:

$$f_c = \frac{1}{2\pi RC}\ , \tag{3.2}$$

onde f_c é a frequência de corte em hertz, R é o valor em ohms do resistor R1 e C é a capacitância em farads do capacitor C1.

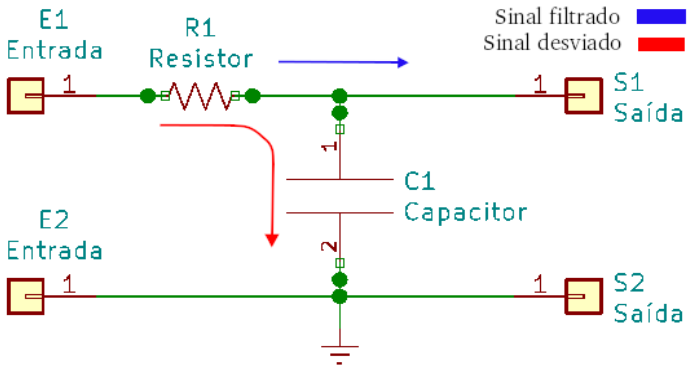


Figura 7 – Diagrama eletrônico de um filtro passa baixa simples.

Na etapa de amostragem uma leitura é periodicamente colhida do sinal filtrado na frequência definida pela taxa de amostragem a fim de criar uma representação discreta em relação ao tempo.

O teorema de Nyquist descreve a relação que a taxa de amostragem possui na representação de sinais de maior frequência. Dado um sinal de frequência f_{sinal}, a taxa de amostragem mínima necessária para sua correta representação é definida por:

$$f_{amostragem} = 2 \times f_{sinal}, \tag{3.3}$$

onde $f_{amostragem}$ e f_{sinal} são frequências em hertz.

Assim de acordo com esse limite qualquer sinal com frequência maior que a metade da taxa de amostragem sofre grande degradação após o processo de amostragem podendo ocasionar em deformação ou perda substancial de informação.

A voz humana ocupa um espectro de frequências relativamente baixa, variando entre cerca de 80 à 8000 hertz. Com isto em mente a taxa de amostragem adotada neste experimento foi de 16000 hertz (ou 16 khz), valor satisfatório para conter todo o espectro da voz humana de acordo com o teorema de Nyquist.

Após a amostragem é realizada a quantização que pode ser entendida como um arredondamento do nível sinal colhido na etapa anterior a fim de torná-lo discreto em relação às amplitudes. Como a maior parte da informação sonora está nos sinais de menor intensidade se utiliza uma quantização não-uniforme também conhecida como u-law, de tal modo que mais bits são utilizados para os níveis mais próximos ao zero, assim resultando em uma melhor resolução em sinais fracos. A obtenção dos níveis de amplitudes discretos pode ser realizada como a seguinte equação:

$$F(x) = sgn(x)\frac{ln(1+\mu|x|)}{ln(1+\mu)}, \tag{3.4}$$

onde $F(x)$ é a amplitude discreta obtida, x é a amplitude contínua do sinal, $sgn(x)$ é a

polaridade do sinal x e μ é o coeficiente de quantização e tem seu valor padronizado em 255.

As etapas de filtro passa-baixa, amostragem e quantização funcionam como um conversor analógico / digital. A etapa final é a de encoder e é responsável pela conversão em uma sequência binária projetada para minimizar a banda utilizada aplicando compressão de dados, esta etapa é específica ao dialeto do PCM adotado. Alguns dialetos incluem o *Differential Pulse Code Modulation* (ou DPCM) e o *Adaptive Differential Pulse Code Modulation* (ADPCM). Os arquivos de áudio no formato wave (com extensão .wav) geralmente são codificados no dialeto ADPCM. Não exploraremos profundamente o dialeto PCM pois seus detalhes são menos interessantes aos objetivos deste trabalho.

A aparência de um sinal senoidal após sua conversão em PCM pode ser visto na Figura 8.

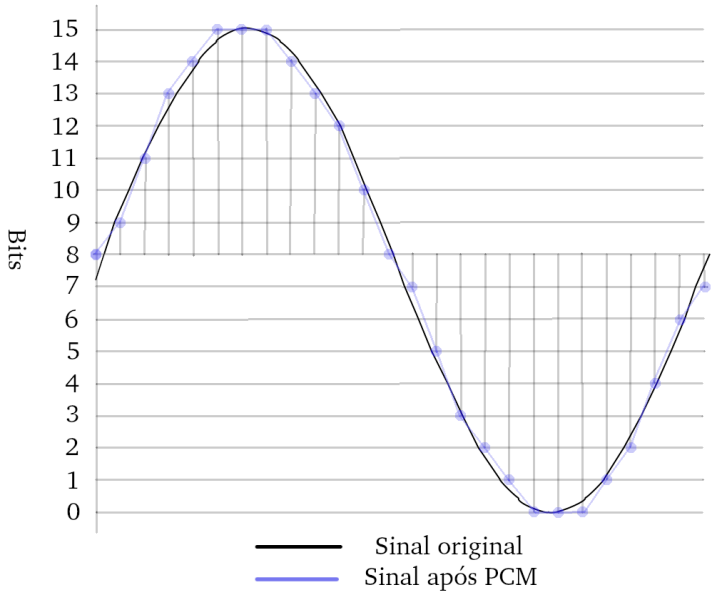


Figura 8 – Funcionamento do PCM.

Mesmo com a obtenção de uma representação discreta de um evento sonoro, a tarefa de obtenção de atributos de alto nível como a emoção transmitida durante o evento de forma sistemática ainda não havia sido solucionada, representando até então um problema em aberto. Fator este que estimula a investigação do problema de reconhecimento de emoções durante a fala com técnicas de AM.

Todo o processo de conversão de um evento sonoro análogico em uma sequência digital de dados tenta preservar as características necessárias para se reproduzir uma representação muito próxima do som original. Mas quando analisada mais atentamente, o conteúdo de um evento gravado muito provavelmente irá conter outros sinais espúrios na forma de ruídos, outros sons indesejados e até distorções introduzidas em

sua conversão digital. Isto acaba dificultando que alguma informação relevante seja separada dos outros sinais que não estão relacionados ao evento que se deseja processar, por exemplo. Assim os sons gravados terão de ser pré processados de modo a focar em alguns aspectos dos mesmos que possam melhor representá-los ainda que resumindo-os.

3.2 Fundamentação teórica do pré-processamento

Na etapa de pré-processamento as amostras são extraídas a partir dos arquivos de áudio e são convertidas em vetores que compõem o conjunto de dados do experimento. O tipo de pré-processamento necessário para um experimento de AM está ligado a natureza dos atributos a serem obtidos. Em aplicações de reconhecimento de áudio processos adicionais são empregados para este fim como filtros, funções de janelas e extração de coeficientes de espectro. O processo de escolher quais aspectos das amostras devem ser utilizados em um experimento de AM é conhecido como seleção de atributos.

3.2.1 Seleção de atributos

Segundo (WU; FALK; CHAN, 2011) seleção de atributos para aplicações em reconhecimento de áudio geralmente se dividem em duas categorias: Atributos prosódicos e atributos espectrais. Atributos prosódicos estão relacionados a características como volume, duração entre outros enquanto os atributos espectrais estão associados a coeficientes distribuídos em um espectro de frequências geralmente menor do que toda banda de sinais presentes na representação original.

Para este experimento optou-se por utilizar os atributos espectrais em favor da combinação com atributos prosódicos, que segundo (MITRA; FRANCO, 2015), resultaria em um melhor classificador. Esta escolha foi no sentido de ajustar o experimento ao orçamento disponível para sua realização. Para capturar a característica sonora presente no espectro foi utilizado a técnica MFCC (*Mel Frequency Cepstral Coefficients*) (MERMELSTEIN, 1976).

3.2.2 Pré-ênfase

A operação de pré-ênfase é comumente presente em aplicações de processamento de áudio e o seu funcionamento é como um filtro que tenta enfatizar as frequências mais altas no sinal da fala. As frequências mais altas estão relacionadas com a maior parte da informação vocal em comparação às frequências mais baixas do espectro da voz. O pré-ênfase se dá pela redução da intensidade das frequências mais baixas, eliminando do seu espectro as que contenham um sinal fraco, possivelmente

resultando em um áudio mais livre de ruídos. A função de pré-ênfase é definida na equação (3.5) e um exemplo de como ela pode afetar a forma de onda de um som pode ser visto na Figura 9.

$$y(t) = x(t) - \alpha x(t-1), \tag{3.5}$$

onde $y(t)$ é a amplitude obtida para o sinal de entrada x no instante t e α é o fator de pré-ênfase a ser aplicado, por exemplo 0.95.

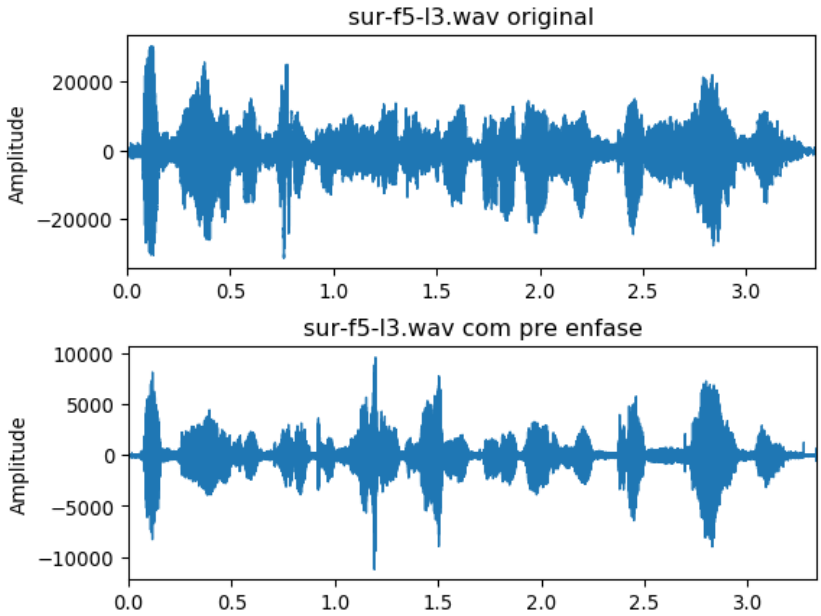


Figura 9 – Exemplo de um áudio com pré-ênfase aplicado.

Podemos observar no resultado do filtro de pré-ênfase que a maioria das amplitudes no sinal foram reduzidas, isto se dá por consequência da maior parte da energia do sinal da voz está concentrada nas frequências mais baixas. Podemos observar também que algumas porções recebem um ”ganho” pela eliminação dos componentes de baixa frequência, estas porções correspondem aos trechos com sinais de maior frequência.

3.2.3 Transformada de fourier

A transformada de fourier é o processo que consegue mudar um sinal representado no domínio do tempo (como o sinal obtido através do processamento PCM) em um representação no domínio das frequências, ou seja, onde podemos verificar a amplitude de cada componente sonoro decomposto na escala das frequências.

Como já foi mencionado um sinal sonoro é composto por uma infinita soma de sinais senoidais (série de fourier) e através da transformada é obtida em forma de um somatório de funções seno e cosseno os sinais fundamentais que compõem qualquer

sinal composto. De acordo com o ilustrado na equação do movimento harmônico (3.1) estes sinais possuem uma frequência fundamental e uma amplitude de modulação. Na Figura 10 pode-se perceber como sinais senoidais de diferentes frequências são decompostos a partir de um sinal que varia no tempo. A transformada de fourier é definida matematicamente como:

$$F(\alpha) = \int_{-\infty}^{+\infty} f(t) e^{-2\pi i \alpha t} dt, \tag{3.6}$$

onde F é a transformada de fourier da frequência α para a função f que varia no tempo t.

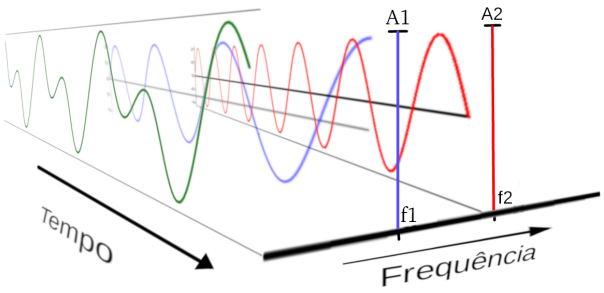


Figura 10 – Transformada de Fourier.

3.2.4 Função hamming

A função hamming tem aplicações em processamento de áudio na tarefa de segmentação das janelas de tempo, a sua equação é apresentada na definição (3.7) e sua aparência é mostrada na Figura 11. Ela é utilizada durante a transformada de fourier e serve a dois propósitos: Primeiro, tornar a computação da transformada mais simples pela segmentação do sinal em um conjunto finito de dados. E segundo, para corrigir um problema que ocorre quando a transformada é realizada em janelas conhecido por *spectral leakage*, que ocorre quando um sinal é recortado em uma janela diferente do seu período, fazendo que apareça ”sobras” de sinais que acabam sendo interpretados como um novo sinal de período mais curto, ou seja, criando um ruído de alta-frequência. A função *hammimg* é aplicada à janela antes da transformada e pela sua forma tende a suavizar essas ”sobras” de sinal que tendem a ocorrer no início e no fim da janela, reduzindo consideravelmente os vazamentos de potência de espectro, tornando assim mais precisa a transformada de fourier.

$$W_{hm}[n] = \left\{ 0.54 + 0.46 \cos \frac{2\pi n}{M-1} \right\}, \ 0 \leq n \leq M-1, \tag{3.7}$$

onde M é a quantidade de amostras na janela W_{hm} e n é a n-ésima amostra da janela.

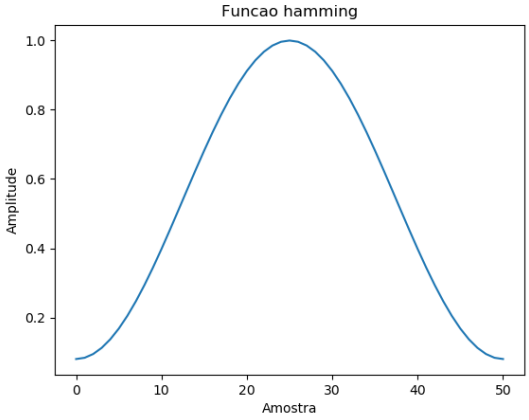


Figura 11 – Gráfico da função Hamming.

3.2.5 Coeficientes MFCC

Os coeficientes MFCC (MERMELSTEIN, 1976) são comumente utilizados em aplicações de reconhecimento de fala por adotar uma escala não-linear conhecida como escala *Mel* (VOLKMANN; STEVENS; NEWMAN, 1937). Esta escala representa satisfatoriamente os sinais característicos da voz e por isso foi escolhida para obter os atributos de espectro para este experimento. Após a revisão da literatura, foi constatado que vetores a partir de 13 coeficientes são suficientes para capturar corretamente as características vocais, porém foi optado por um vetor de 24 coeficientes neste experimento. Os coeficientes MFCCs podem ser obtidos com a seguinte equação:

$$C_n = \sum_{k=1}^{K} log(S_k) cos[n(k - \frac{1}{2})\frac{\pi}{k}], n = 1, ..., L, \tag{3.8}$$

onde C_n é o n-ésimo coeficiente do vetor MFCC, L é a quantidade de coeficientes a serem obtidos, K é uma janela proveniente da função *hamming* com K amostras e $S(k)$ é a saída da transformada de fourier para a k-ésima amostra da janela.

3.3 Representando o som como um vetor de dados

De acordo com (SHRAWANKAR; THAKARE, 2013) as propriedades do som da voz humana são quase estacionárias (*quasi-stationary*), ou seja quase não variam quando se considera uma janela de tempo curta, da ordem de 5 a 100ms. Para capturar a evolução temporal dessas mudanças foi empregada a função *hamming* que visa recortar o clipe em janelas menores de tempo como detalhado na Seção 3.2.4, assim os atributos de cada janela são extraídos resultando um vetor de coeficientes MFCC para cada janela de tempo.

Como a duração de cada gravação (clipe) de áudio naturalmente apresenta uma variação em seus comprimentos, isto implica em uma quantidade variável de vetores MFCC. Uma vez que os modelos de AM empregados necessitam de um vetor de atributos de tamanho fixo, foram aplicados ajustes para que as amostras possuísem uma representação de comprimento fixo para todos os clipes. Assim clipes de menor duração tiveram silêncio adicionado ao seu fim (*padding*) e clipes maiores tiveram que ser truncados (*cropping*).

3.3.1 Ajustes no tamanho das amostras

As amostras obtidas após o pré-processamento são matrizes de tamanho ($24 \times Q$), onde 24 se refere aos coeficientes MFCC e Q é a quantidade de janelas da amostra.

Após a obtenção da matriz de vetores MFCC o tamanho da dimensão das janelas Q é ajustado para que tenha um valor fixo de 500. Este valor foi escolhido de tal forma que a maioria dos clipes de áudio fossem menores que este tamanho e que poucas amostras precisassem ter suas janelas finais descartadas, preservando assim a maior parte da informação. Assim se há janelas após a quinquagésima, então as janelas MFCC excedentes são descartadas e se não há janelas suficientes, então janelas MFCC vazias são adicionadas ao fim de modo a sempre obter um espectrograma de tamanho fixo em (24x500) o que totaliza 12000 parâmetros.

Assim, por causa das restrições de espaço todos os valores sofreram arredondamentos e apenas os 10 primeiros coeficientes MFCC e as 20 primeiras janelas constam nas tabelas a seguir. Na Tabela 1 pode ser visualizado um exemplo de amostra após o pré-processamento pertencente ao rótulo ”surpresa” e na Tabela 2 uma amostra pertencente ao rótulo ”neutro”.

Assim o resultado final do pré processamento é um espectrograma de tamanho fixo para cada amostra. Para fins de armazenamento e para que possa ser utilizado nos diferentes algoritmos de AM a matriz resultante é convertida em um único vetor. Um exemplo de como é a aparência de um espectrograma pode ser visto na Figura 12.

Embora os sons possam ser representados formalmente e substancialmente resumidos ainda assim não podemos estimar diretamente atributos como a emoção transmitida analisando estes valores de maneira absoluta, devida a grande variância encontrada entre diferentes locutores. Contudo, com a aplicação de aprendizado de máquina e um grande volume de dados pode ser possível encontrar estas relações em forma de regras e assim fazer boas previsões sobre o estado emocional de um locutor.

	Coeficientes MFCCs									
1º vetor	**0.33**	**0.02**	**-0.06**	**0.05**	**0.01**	**0.02**	**0.03**	**0.02**	**0.02**	**-0.01**
2º vetor	0.37	-0.04	-0.06	0.06	-0.02	0.01	-0.01	0.04	-0.03	0.01
3º vetor	**0.47**	**-0.14**	**-0.09**	**0.11**	**-0.05**	**-0.01**	**-0.06**	**0.06**	**-0.07**	**0.01**
4º vetor	0.50	-0.16	-0.11	0.12	-0.05	0.01	-0.08	0.06	-0.04	-0.03
5º vetor	**0.44**	**-0.12**	**-0.13**	**0.11**	**-0.03**	**-0.00**	**-0.06**	**0.04**	**-0.03**	**-0.04**
6º vetor	0.43	-0.11	-0.13	0.10	-0.01	-0.03	-0.03	0.08	-0.07	-0.02
7º vetor	**0.45**	**-0.07**	**-0.12**	**0.08**	**0.03**	**-0.03**	**-0.04**	**0.08**	**-0.09**	**-0.04**
8º vetor	0.58	-0.04	-0.22	0.14	0.08	-0.06	-0.08	0.08	-0.06	-0.12
9º vetor	**0.88**	**-0.01**	**-0.33**	**0.35**	**0.11**	**-0.13**	**-0.08**	**0.03**	**-0.07**	**-0.24**
10º vetor	1.32	-0.08	-0.53	0.63	0.14	-0.18	-0.05	-0.06	-0.09	-0.30
11º vetor	**2.13**	**-0.16**	**-0.99**	**1.16**	**0.18**	**-0.39**	**0.11**	**-0.18**	**-0.27**	**-0.33**
12º vetor	3.20	-0.37	-1.67	1.72	0.21	-0.49	0.12	-0.41	-0.18	-0.42
13º vetor	**4.38**	**-0.78**	**-2.59**	**2.46**	**0.06**	**-0.79**	**0.45**	**-0.44**	**-0.10**	**-0.45**
14º vetor	5.57	-0.69	-3.17	3.17	0.18	-1.19	0.61	-0.74	-0.53	-0.51
15º vetor	**8.17**	**-0.91**	**-5.06**	**4.14**	**1.01**	**-1.05**	**-0.07**	**-1.55**	**0.51**	**-0.44**
16º vetor	12.84	-0.91	-8.38	6.14	2.82	-1.41	-0.31	-1.83	1.33	-0.32
17º vetor	**15.62**	**-0.37**	**-9.83**	**7.55**	**4.84**	**-1.80**	**-0.24**	**-0.03**	**1.45**	**-0.63**
18º vetor	18.19	1.49	-6.38	9.15	4.05	1.00	1.44	-0.10	0.92	-0.79
19º vetor	**21.56**	**1.34**	**-6.35**	**12.10**	**4.56**	**-0.76**	**1.15**	**0.29**	**-1.14**	**-1.14**
20º vetor	21.05	2.39	-4.45	11.52	3.87	-1.47	-0.39	0.16	-0.72	-1.33

Tabela 1 – Exemplo de amostra do rótulo surpresa (10 coef. MFCC x 20 janelas).

	Coeficientes MFCCs									
1º vetor	**0.05**	**0.01**	**-0.02**	**0.01**	**0.01**	**-0.01**	**-0.00**	**0.00**	**-0.01**	**0.00**
2º vetor	0.11	0.07	0.02	0.04	0.02	-0.02	-0.02	-0.02	-0.03	-0.02
3º vetor	**0.63**	**0.51**	**0.16**	**0.29**	**0.17**	**-0.13**	**-0.12**	**-0.12**	**-0.22**	**-0.15**
4º vetor	1.50	0.97	0.12	0.82	0.62	-0.33	-0.18	0.08	-0.32	-0.42
5º vetor	**2.22**	**1.50**	**0.27**	**1.44**	**1.14**	**-0.38**	**-0.12**	**0.35**	**-0.39**	**-0.75**
6º vetor	2.55	2.21	1.07	1.82	1.33	-0.09	-0.10	-0.05	-0.84	-1.07
7º vetor	**2.60**	**2.40**	**1.28**	**1.79**	**1.34**	**0.11**	**-0.12**	**-0.26**	**-0.83**	**-0.97**
8º vetor	2.58	2.26	1.14	1.83	1.42	0.06	-0.08	-0.07	-0.70	-0.90
9º vetor	**2.50**	**1.89**	**0.70**	**1.82**	**1.38**	**-0.18**	**0.01**	**0.20**	**-0.58**	**-0.58**
10º vetor	2.71	1.50	-0.06	1.80	1.25	-0.72	0.06	0.47	-0.69	-0.26
11º vetor	**4.08**	**1.40**	**-1.53**	**2.24**	**1.53**	**-2.20**	**-0.23**	**1.24**	**-1.36**	**-0.51**
12º vetor	3.70	1.25	-1.67	1.62	1.55	-1.49	-0.35	0.62	-1.11	-0.03
13º vetor	**1.96**	**0.81**	**-0.80**	**0.60**	**0.89**	**-0.23**	**-0.13**	**-0.08**	**-0.39**	**0.47**
14º vetor	0.83	0.48	-0.18	0.22	0.43	0.12	-0.04	-0.17	-0.15	0.23
15º vetor	**0.36**	**0.25**	**0.02**	**0.17**	**0.19**	**0.02**	**0.00**	**-0.00**	**-0.06**	**0.03**
16º vetor	0.28	0.17	0.00	0.15	0.10	-0.08	-0.02	0.03	-0.07	-0.02
17º vetor	**0.26**	**0.17**	**0.00**	**0.13**	**0.10**	**-0.05**	**-0.02**	**0.01**	**-0.06**	**-0.01**
18º vetor	0.20	0.14	0.03	0.10	0.08	-0.02	-0.01	0.01	-0.03	-0.00
19º vetor	**0.25**	**0.14**	**-0.06**	**0.04**	**0.09**	**0.02**	**-0.01**	**-0.02**	**-0.03**	**0.02**
20º vetor	1.01	0.49	-0.61	-0.34	0.25	0.19	-0.19	-0.40	-0.12	0.30

Tabela 2 – Exemplo de amostra do rótulo neutro (10 coef. MFCC x 20 janelas).

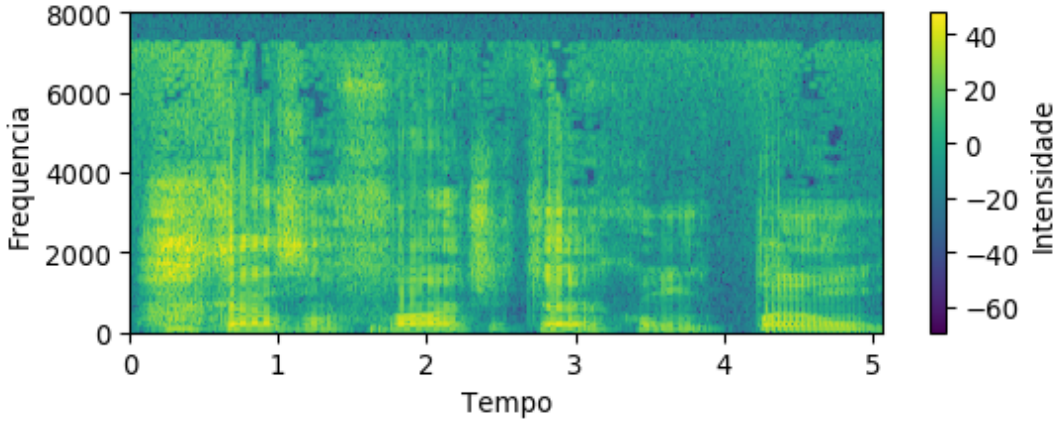


Figura 12 – Aparência de um espectrograma.

4 Métodos de aprendizado de Máquina

Neste estudo aplicamos técnicas de aprendizado de máquina (AM) na tarefa da classificação de emoções presentes na voz, assim sendo precisamos entender alguns de seus conceitos.

Aprendizado de Máquina é uma área dos estudos de inteligência artificial que foca no desenvolvimento de técnicas computacionais a serem empregadas em sistemas capazes de obter novos conhecimentos. AM pode ser entendido como “uma abordagem guiada a dados para a resolução de problemas, na qual padrões ocultos em um conjunto de dados são identificados e utilizados para ajudar na tomada de decisões”, segundo (BRINK; RICHARDS; FETHEROLF, 2017).

A resolução de problemas, como a identificação de qual classe pertence uma amostra, pode ser possível com o auxílio de AM empregando-se técnicas de aprendizado supervisionado ou de aprendizado não supervisionado. Existem outras abordagens menos comuns como o aprendizado semi supervisionado e classificação de multi-classes. Neste estudo, apenas focaremos nas técnicas do aprendizado supervisionado aplicada a um problema de classificação.

4.1 Aprendizado Supervisionado

No aprendizado supervisionado procura-se encontrar a partir de um conjunto de dados os parâmetros para o modelo que gerou o conjunto de dados em primeiro lugar. Assim é necessário que o conjunto de dados reúna diversas amostras cujo o resultado já é conhecido. Este resultado em nosso estudo são as diferentes emoções que um locutor pode transmitir pela voz. Em AM, são conhecidas por classes as diferentes categorias em que se deseja classificar as amostras.

Seja (a_1, a_2, ..., a_n) os atributos de uma amostra *x* e *m* a quantidade de amostras, podemos definir *M* como uma matriz de tamanho $m \times n$ que representa os dados do treinamento. Seja *y* o rótulo de uma amostra *x* e *L* = (y_1, y_2, ..., y_m) um vetor de tamanho *m* com os rótulos dos dados de treinamento, então é possível definir o algoritmo de aprendizado de máquina como uma função *f*, tal que:

$$f : M \rightarrow L, \tag{4.1}$$

Os atributos são características observáveis em cada amostra que será utilizada para diferenciar as classes do problema. O que implica que os atributos bem

escolhidos são essenciais para o sucesso de um sistema de AM e atributos que não ajudam a diferenciar o problema acarreta em resultados erráticos. A suposição é que existe alguma regra escondida nos dados e que queremos encontrá-la de maneira sistemática.

O algoritmo de AM pode aprender estas regras após uma etapa de treinamento e assim adquire uma representação geral do conhecimento existente naqueles dados de treinamento. É chamada de poder de generalização a capacidade de um modelo de classificar um dado inédito de forma correta a partir do modelo aprendido.

Por outro lado outros fatores como poucos dados, treinamento demasiado ou uma sobre-capacidade do modelo pode levar a uma situação de *overfiting*, onde o aprendizado é prejudicado resultando em um baixo poder de generalização do modelo aprendido (mesmo quando o modelo pontua muito bem no conjunto de treinamento).

O modelo que alcançar um bom poder de generalização pode ser utilizado para fazer boas previsões sobre dados mesmos que estes nunca tenham sido observados antes (durante a etapa de treinamento).

A Classificação é “a predição de novos dados em baldes (classes) utilizando um classificador modelado por um algoritmo de AM” (BRINK; RICHARDS; FETHEROLF, 2017). Na classificação a classe (ou rótulo) de uma amostra é a saída do modelo aprendido durante o treinamento para uma dada amostra cujo o rótulo é desconhecido.

Os algoritmos de AM (ou classificadores) possuem por diversas vezes parâmetros que precisam ser definidos para a sua utilização, são os chamados hiperparâmetros. Estes parâmetros têm total influência no seu poder de generalização, sendo o ajuste ótimo outro fator crítico para o sucesso do qualquer experimento em AM.

Ao final é desejado encontrar o classificador que consiga desempenhar o papel da função f, que relaciona a matriz de dados M, que contém os atributos das amostras no banco de dados, ao vetor de rótulos L, contendo os seguintes valores: ”alegria”, ”surpresa”, ”desgosto”, ”neutro”, ”medo”, ”raiva” e ”tristeza”.

4.2 Naive Bayes

O algoritmo Naive Bayes utiliza uma técnica estatística de probabilidade condicional baseada no teorema de Thomas Bayes. Embora seja de simples implementação por não requerer hiperparâmetros, este algoritmo possui um forte viés. É assumido que a ocorrência de um atributo não tem correlação alguma com a ocorrência dos outros atributos (daí o nome *naive* que significa ingênuo).

Uma afirmação como esta pode não se sustentar para eventos do mundo real onde atributos podem estar correlacionados. Mesmo assim este forte viés não invalida

a utilização deste algoritmo para aplicações do mundo real.

O teorema de Bayes diz a probabilidade de dois eventos que ocorram em sequência. Ou seja a probabilidade de A ocorrer sob a condição de B ocorrer e é exposto a seguir:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}, P(B) \neq 0. \tag{4.2}$$

O termo $P(A|B)$ pode ser lido como a probabilidade da classe A ocorrer se ocorrerem as evidências (atributos) B.

O termo $P(B|A)$ é a probabilidade dos atributos independentes ocorrerem se a classe da instância for A e pode ser obtida a partir da contagem dos casos em que cada atributo ocorre para cada classe no conjunto de dados.

O termo $P(A)$ é a probabilidade *a priori* de uma determinada classe ocorrer no conjunto de dados e também é obtida pela contagem de casos relevantes no conjunto de dados.

O termo $P(B)$ é a probabilidade *a priori* que ocorra a evidência no conjunto de dados sendo utilizado apenas para normalizar o resultado. $P(B)$ assume o mesmo valor para cada classe do problema de classificação por não levar em conta a classe e assim pode ser omitido. Sua ausência não fará diferença quando calculada a probabilidade de cada classe na classificação de uma instância, já que é um termo denominador comum entre todas as classes.

O treino da rede desconsidera qualquer instância que possua dados faltosos e consiste em se calcular as frequências relativas de cada atributo em relação a cada classe e as frequências relativas das classes no conjunto de treino.

Para testar se uma amostra pertence a uma determinada classe, o termo P(A) é substituído pela frequência de ocorrência da classe no conjunto de treino e o termo P(B|A) é a probabilidade resultante das frequências relativas de cada atributo observado em relação a classe A no conjunto de treino. Como cada atributo é considerado independente dos outros, o termo P(B|A) é o valor de suas probabilidades individuais multiplicadas para aquela instância.

Como o que se deseja é determinar a classe para uma instância desconhecida o problema se resume então a encontrar o maior produto de P(B|A) com P(A) dentre as classes do problema. A divisão pelo termo constante P(B) não muda a relação entre as probabilidades finais das classes do problema. Assim a equação da probabilidade P(A|B) pode ser simplificada, como pode ser vista a seguir:

$$P(A|B) = P(B|A) \times P(A). \tag{4.3}$$

Ao fim os resultados para cada classe são normalizados dividindo-se cada valor pela soma de todos os valores encontrados. A classe com maior probabilidade é a escolhida para amostra.

Uma de suas principais vantagens é a ausência da fase de treinamento uma vez que essa fase envolve apenas o cálculo de frequências relativas das classes que é realizado quase instantâneamente. Algo que pôde ser comprovado pelo baixo tempo que leva o treinamento neste algoritmo em relação ao de todos os outros.

4.3 Árvores de decisão

Uma árvore de decisão é um modelo de AM que pode ser utilizado em tarefas de classificação e regressão cuja estrutura consiste de um determinado número de nós e folhas organizados como um grafo acíclico.

Este modelo tem uma forma que lembra uma árvore de cabeça para baixo, com a raiz em cima e ramificações que vão crescendo para baixo na forma de novos nós e folhas. Cada nó interno da árvore representa um determinado teste sobre um atributo de uma instância e o resultado do teste realizado é representado pela ligação a outro nó ou a uma folha. As folhas são os nós mais extremos da árvore e são referentes aos rótulos da classificação. Como a classe ”Sim”e ”Não” do exemplo mostrado na Figura 13.

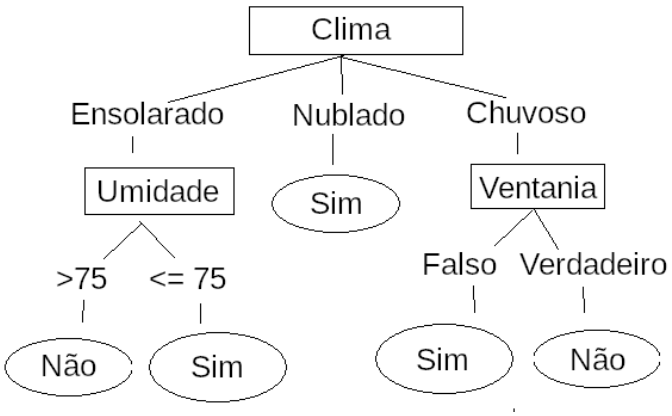


Figura 13 – Árvore de decisão simples.

Ao classificar uma determinada instância, a AD é percorrida recursivamente a partir do nó raiz realizando testes a cada nó visitado. Quando um nó interno é visitado um novo teste é realizado e este processo recursivo ocorre até que se alcance um nó

folha, situação em que o caso base da recursão é satisfeita e resultando na classificação da instância. Em algoritmos recursivos como na árvore de decisão, o caso base consiste na condição que é utilizada para encerrar o processo recursivo.

O objetivo das ADs é subdividir o espaço dos atributos em espaços menores de tal forma que possam definir as classes do problema, ou seja encontrando sub espaços que possam ser associados a apenas uma das classes do problema.

O treino de uma árvore de decisão envolve escolher qual atributo será utilizado a cada nó que melhor separa as classes do problema.

Utilizando recursão, o problema de se separar o espaço de atributo vai sendo dividido em instâncias mais simples, onde uma mesma estratégia pode ser aplicada até que se alcance o caso base. No caso do treino de ADs, o caso base ocorre quando todos o nós folhas contém apenas amostras de uma mesma classe.

A fim de se poder estabelecer um comparativo entre os atributos disponíveis que melhor ofereça esta diferenciação é preciso definir uma métrica que melhor captura essa propriedade.

Podemos definir a utilidade em separar o espaço dos atributos como o ponto que separe o espaço de tal forma que agrupe as instâncias mais semelhantes (homogeneidade) enquanto separa grupos mais diferentes (heterogêneos) possíveis. A métrica utilizada é o *ganho de informação*, que é definida a seguir:

$$G(S, A) = E(S) - \sum_{i=1}^{m} P(A_i)E(S_i), \tag{4.4}$$

onde $G(S, A)$ é o ganho de informação para o conjunto de amostras S em relaçãp ao atributo A que assume m valores distintos A_i, $E(S)$ é a entropia do conjunto S, $P(A_i)$ é a probabilidade de A assumir o valor A_i em S e $E(S_i)$ é a entropia do subconjunto S_i composto de instâncias que possuam o atributo A com valor A_i.

Entropia em teoria da informação pode ser compreendido pela quantidade de bits necessários em média para poder armazenar uma informação e é definida a seguir:

$$E(S) = -\sum_{i=1}^{n} P(C_i)log_2P(C_i). \tag{4.5}$$

Sendo que $E(S)$ é a entropia do conjunto S, n a quantidade de classes do problema e $P(C_i)$ a probabilidade da classe C_i ocorrer .

O seguinte pseudo-código ilustra o algoritmo de obtenção de uma árvore de

decisão:

Algoritmo 1: Cria_AD.

Entrada: Conjunto de atributos A, Matriz de amostras S, número mínimo de amostras $minAmostras$ e número máximo de nodos $maxNodos$.

Saída: Árvore treinada AD.

início
  $AD =$ ÁrvoreVazia()
  `#Checa se é o caso base.`
  se Entropia$(S) = 0$ ***ou*** ContaNodosCriados$() = maxNodos$ **então**
    $folha =$ NovaFolha(S)
    SomaÁrvores$(AD, folha)$
    retorna AD
  fim
  `#Encontra atributo com melhor ganho que agrupe ao menos` $minAmostras$ `em cada subespaço`
  $listaGanhos = [\,]$
  para *cada* $atributo \in A$ **faça**
    $checaSubs = Verdadeiro$
    para *cada* $subespaco \in$ *ParticionaDados*$(S, atributo)$ **faça**
      se ContaAmostras$(subespaco) < minAmostras$ **então**
        $checaSubs = Falso$
      fim
    fim
    se $checaSubs = Verdadeiro$ **então**
      $listaGanhos \leftarrow listaGanhos + Ganho(S, atributo)$
    fim
  fim
  $atributo_{escolhido} =$ EscolheMaior$(listaGanhos)$
  `#Adiciona nodos na árvore`
  $nodo =$ *NovoNodo*$()$
  AdicionaTeste$(nodo, atributo_{escolhido})$
  SomaÁrvores$(AD, nodo)$
  `#Chamada recursiva para cada subespaco`
  para *cada* $subespaco \in$ *ParticionaDados*$(S, atributo_{escolhido})$ **faça**
    $subArvore =$ Cria_AD$(A, subespaco, minAmostras, maxNodos)$
    SomaÁrvores$(AD, subArvore)$
  fim
fim
retorna AD

Árvores de Decisão tem como maior vantagem a fácil compreensão do modelo treinado. Um dendograma pode ser utilizado para visualização de uma árvore de decisão assim possibilitando uma boa análise e melhor compreensão das regras aprendidas.

Na Figura 13 é mostrada uma árvore de decisão baseada no weather dataset disponível em várias ferramentas de AM. Neste exemplo o modelo tentou aprender uma regra para o problema de se podemos ou não fazer alguma atividade ao ar livre baseada no clima, umidade relativa do ar e condições do vento.

4.4 Redes Neurais

Multilayer Perceptron (MLP) é uma classe de algoritmos de AM que utiliza um sistema de computação paralela inspirada no modelo biológico dos neurônios no cérebro. Na Figura 14 temos a representação do neurônio artificial.

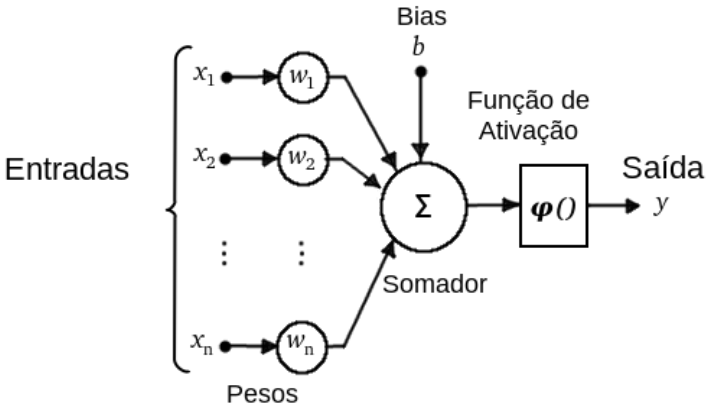


Figura 14 – Modelo matemático do neurônio artificial.
Fonte: Adaptado de (HAYKIN, 1994).

Cada neurônio é modelado como diversas variáveis de entrada que mediante a um estímulo externo tem seus sinais somados e ponderados com pesos individuais. Esse resultado é submetido a uma função de ativação (que é definida por hiperparâmetros), qualquer função de ativação pode ser utilizada desde que seja derivável (contínua), as funções não-lineares são, em geral, escolhidas por melhor modelar os problemas da vida real que geralmente possuem estas não-linearidades como característica.

Considerando um neurônio com duas entradas $x_!$ e x_2, pesos w_1 e w_2, um bias b_1, uma saída y_1 e uma função de ativação φ_1. Podemos definir sua computação da seguinte forma:

$$y_1 = \varphi_1(z_1(x_1, x_2, w_1, w_2, b_1)), \tag{4.6}$$

sendo que,

$$z_1(x_1, x_2, w_1, w_2, b_1) = (x_1 \times w_1) + (x_2 \times w_2) + b_1.$$

A equação de z pode ser reescrita como:

$$z = \sum_i x_i \times w_i + b_i. \tag{4.7}$$

Exemplos de funções de ativação comumente empregadas incluem (mas não se limitam) a sigmoid, a tangente hiperbólica e ReLU (*REctified Linear Unit*). A popular adoção destas se dá por conta de possuírem funções derivadas de simples obtenção (conveniência matemática) o que será bastante útil para o treinamento da rede.

A sigmoid ou logistic é uma função que possui em sua imagem os valores no intervalo de 0 a 1 como pode ser visto na Figura 15. A função sigmoid é definida a seguir:

$$\varphi(x) = \frac{1}{1 + e^{-x}}. \tag{4.8}$$

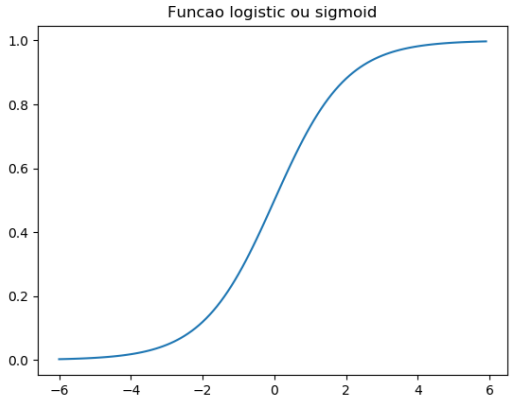


Figura 15 – Função sigmoid ou logistic.

A tangente hiperbólica é similar a sigmoid mas tem a imagem no intervalo entre -1 e 1 como pode ser visto na Figura 16. A função tangente hiperbólica é definida a seguir:

$$\varphi(x) = \frac{2}{1 + e^{-2x}} - 1. \tag{4.9}$$

A função ReLU tem na sua imagem os valores acima de zero e é similar a uma função identidade quando a sua entrada é positiva como pode ser visto na Figura 17. A função ReLU é definida a seguir:

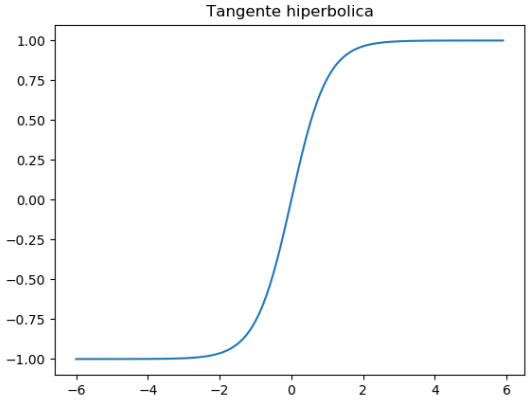


Figura 16 – Função tangente hiperbólica.

$$\varphi(x) = \begin{cases} x, & se\ x \geq 0 \\ 0, & se\ x < 0. \end{cases} \tag{4.10}$$

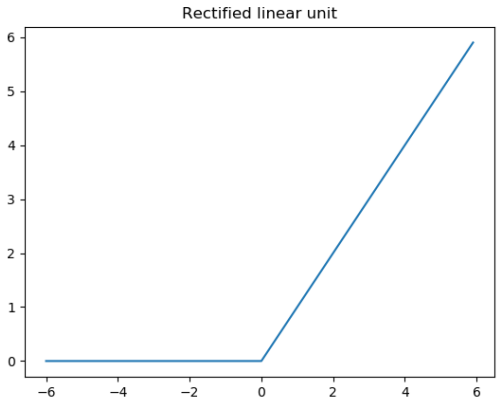


Figura 17 – Função ReLU.

O neurônio então utiliza a função de ativação para propagar ou não o somatório dos estímulos recebidos ponderado pelos pesos associados com cada entrada. Quando o neurônio propaga o estímulo recebido nas suas entradas para a sua única saída é dito que o neurônio foi ativado. O peso aplicado sobre cada entrada do neurônio é o fator que decide se um neurônio propaga ou não o estímulo através da função de ativação (conhecimento da rede).

Em uma rede MLP existem ao menos 3 camadas como pode ser visto na Figura 18.

A primeira camada não realiza computação e é o vetor de entrada. A quantidade de neurônios nesta camada equivale a quantidade de atributos que existem no

problema considerado. A camada seguinte é totalmente conectada a sua camada anterior, ou seja cada saída de um neurônio em uma camada é ligada a uma entrada de cada neurônio que existe na camada seguinte. As camadas intermediárias (também conhecidas como camadas escondidas) tem um número e quantidade de neurônios que é definida por um hiperparâmetro durante a sua inicialização e tem grande influência no poder de generalização do modelo.

A última camada é o vetor de saída e tem tamanho igual ao número de classes que existem no conjunto de dados, a ativação desta última camada é dada pela função softmax para problemas de classificação, que fornece um valor entre 0 e 1 que pode ser entendido como a probabilidade da instância pertencer a uma determinada classe.

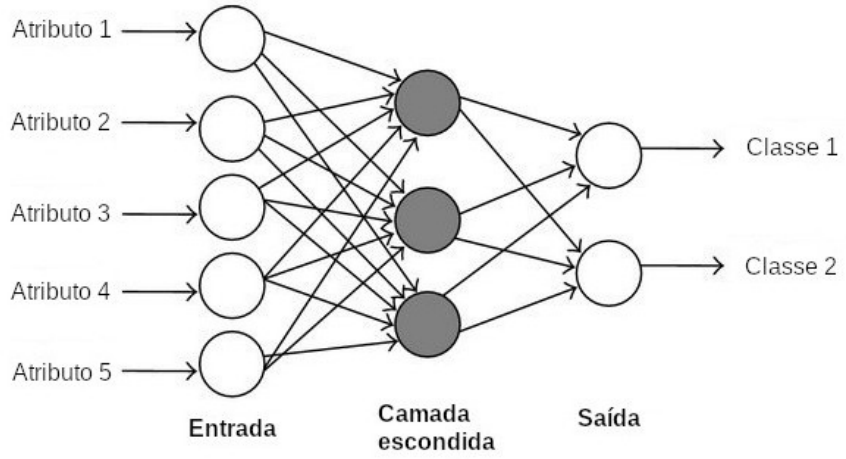


Figura 18 – Um MLP simples.

A rede funciona da seguinte forma quando vai classificar uma amostra: Os estímulos presentes na camada de entrada da rede são propagados para a camada seguinte, logo os neurônios desta camada são todos ativados em paralelo e as suas saídas são as entradas da camada seguinte. Este processo repete em sequência para todas as camadas até que se alcance a camada de saída, assim obtendo o resultado da computação. Pode-se perceber que o fluxo das operações flui da entrada para a saída e este processo recebe o nome de *Forward propagation*.

Com um processo de treinamento a rede consegue aprender novos conhecimentos na forma de uma regra que generalize o problema, logo a tarefa se resume em gradualmente ajustar os pesos de cada neurônio a fim de que os erros de classificação diminuam. Ao início de um um treinamento todos os pesos da rede são inicializados de forma aleatória. Isto é possível com o algoritmo conhecido como *Backpropagation*.

O *Backpropagation* propaga os erros obtidos na classificação das amostras de treino para toda rede e promove estes ajustes para que o erro da rede seja minimizado. Como o nome sugere, estes ajustes têm o fluxo de operações fluindo da saída para

entrada da rede, o motivo do fluxo ser nesse sentido é que só podemos medir o erro a partir da saída do modelo.

O método adotado para corrigir os erros (também conhecido como regra *Delta* (BISHOP, 2001)) calcula o gradiente descendente da função erro E em relação aos pesos (variáveis que contém o conhecimento da rede) como pode ser visto a seguir:

$$-\alpha \Delta E \ , \tag{4.11}$$

sendo que:

$$\alpha \text{ é a } \textit{taxa de aprendizado} \text{ e } \Delta E = \frac{\partial E}{\partial w}.$$

Com o uso da regra da cadeia proveniente do cálculo numérico e da equação (4.6), podemos reescrever ΔE para um peso w_s de um neurônio em uma camada de saída como:

$$\Delta E_{w_s} = \frac{\partial E}{\partial \varphi_s} \times \frac{\partial \varphi_s}{\partial z_s} \times \frac{\partial z_s}{\partial w_s}. \tag{4.12}$$

Então o peso w_s pode sofrer o gradiente descendente propagado:

$$w_{s\ ajustado} = w_s - \alpha \Delta E_{w_s}. \tag{4.13}$$

Este processo é realizado em paralelo para todos os neurônios desta camada. Após todos os pesos da camada de saída receberem ajustes é a vez da camada escondida, onde o processo é repetido de forma praticamente idêntica, se diferenciando apenas no termo $\frac{\partial E}{\partial \varphi_h}$ que passa a ser o somatório dos erros propagados por cada neurônio na camada de saída. Isto é devido ao fato de que a saída de cada neurônio na camada escondida estar conectada a uma entrada de cada neurônio da camada de saída.

Seja w_h o peso de um neurônio na camada escondida assim temos:

$$\Delta E_{w_h} = (\sum_s \frac{\partial E}{\partial \varphi_s} \times \frac{\partial \varphi_s}{\partial z_s} \times \frac{\partial z_s}{\partial \varphi_h}) \times \frac{\partial \varphi_h}{\partial z_h} \times \frac{\partial z_h}{\partial w_h}. \tag{4.14}$$

Para finalmente w_h sofrer o ajuste:

$$w_{h\ ajustado} = w_h - \alpha \Delta E_{w_h}. \tag{4.15}$$

O processo acaba quando são realizadas os ajustes no peso da camada de entrada utilizando o mesmo princípio.

Pode-se dizer que a cada iteração do treino (também chamado de época), o erro obtido é propagado para a rede inteira a partir da camada de saída que tem seus pesos ligeiramente reajustados. A taxa destes ajustes é definido pelo hiperparâmetro *taxa de aprendizado* ou α). Com o avançar das camadas se percebe que os ajustes vão ficando cada vez mais tênues pois uma parcela significativa do erro vai sendo propagado nas camadas anteriores, isto leva a uma desvantagem das redes neurais que ocorre quando existem camadas suficientes para que a parcela de erro vá ficando tão pequena que impede que a rede treine corretamente as suas camadas iniciais, problema este conhecido como *vanishing gradient*.

O gradiente de uma função pode ser entendido como a direção em que os ajustes teriam que ser feitos para que a função cresça, como a função em questão é a função de erro em respeito às ativações e pesos anteriores o que resulta em uma função composta como na equação (4.6) que na verdade desejamos minimizar, logo é adotado a direção oposta (daí o nome descendente) da direção obtida pelo cálculo do gradiente de modo que o erro obtido tende então a diminuir.

Este processo é repetido com todas as amostras do conjunto de treino por diversas épocas até que se perceba que o erro total da rede obtida pela função erro também conhecida como função custo ou (*loss funcition*) convirja para um valor mínimo.

O treino acaba quando se completam todas as épocas previstas ou quando nota-se que o erro total parou de diminuir, caso em que se continuado o treinamento ocorre o aumento dos erros de classificação a cada iteração em um processo de *overfiting*.

O seguinte pseudo-código ilustra o algoritmo de *backpropagation*:

Algoritmo 2: Backpropagation.

Entrada: Quantidade de épocas E, Matriz de amostras M, vetor de rótulos L, taxa de aprendizado T e a topologia de rede N.

Saída: Rede treinada N.

```
início
    IniciaPesos(N)
    epoca = 0
    repita
        para cada amostra, rotulo ∈ (M, L) faça
            previsao = ForwardPropagate(amostra)
            erro = FunçãoCusto(previsao, rotulo)
            para cada camada ∈ N da última para a primeira faça
                para cada peso ∈ camada faça
                    peso = peso − T × RegraDelta(peso, erro)
                fim
            fim
        fim
        epoca = epoca + 1
    enquanto epoca < E;
fim
retorna N
```

O *overfitting* pode ser entendido como um processo onde a rede por ter muita capacidade passa a modelar (aprender) detalhes menos interessantes de amostras como ruídos ou outras excentricidades das mesmas, o que resulta em um modelo injustificavelmente muito mais complexo do que o necessário para uma satisfatória (mais suave) modelagem do problema.

Os efeitos de uma rede que apresenta *overfitting* incluem resultados erráticos e baixo poder de generalização, como pode ser observada na Figura 19. A ocorrência de *overfitting* não é uma particularidade das redes neurais, porém podem ocorrer com extrema facilidade nelas. O outro lado da moeda é o *underfitting* e ocorre quando o modelo é demasiadamente simples para capturar uma regra que represente bem as classes e acaba generalizando muito mal em dados inéditos. A maior diferença entre estas duas situações indesejadas é que no overfitting o erro de teste continua a diminuir quando o erro de validação começa a aumentar, se persistir nessa situação a rede passa generalizar menos o problema atingindo assim piores resultados mesmo quando pontua bem no conjunto de teste.

As redes neurais têm a vantagem de ser mais resistentes a uma má seleção de

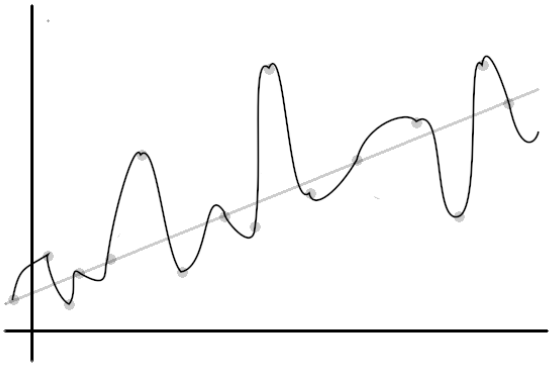


Figura 19 – Exemplo de uma função que apresenta *overfitting*.

atributos e a ruídos nos dados, uma vez que o processo de treinamento vai tender a diminuir os pesos relacionados a estes atributos. Uma das principais desvantagens das redes neurais é devido a forma que ela é treinada, já que tenta minimizar a função erro do modelo e o critério de parada pode ser satisfeito quando atinge um mínimo local, o que não torna possível garantir que o menor erro foi alcançado naquele treinamento, já que ele pode ter sido interrompido antes que os mínimos globais desta função fossem alcançados. Além do já citado problema do *vanishing gradient*.

4.5 SVM

As máquinas de vetor suporte (SVM) são algoritmos de AM fundamentados na teoria do aprendizado estatístico podendo ser entendido como uma implementação do método de minimização do risco estrutural (MRE).

De forma básica, seu funcionamento é baseado no fato de que o erro de classificação de um modelo em dados inéditos (erro de generalização) está ligado a soma do erro de treinamento e um termo que depende da dimensão *Vapnik-Cherovnenkis* (VLADIMIR, 1999) mesmo sem depender de nenhum conhecimento sobre o domínio do problema.

Em problemas mais simples, quando as amostras do problema estão dispostas em um plano, pode-se notar que as instâncias da mesma classe tendem a estar mais próximas entre si, de modo que é possível obter fronteiras de decisão que separam as classes. Para se obter a classe de uma nova instância apenas se consulta as fronteiras já existentes de acordo com o ponto pertencente aquela amostra no plano.

Um problema é dito linearmente separável quando uma fronteira na forma de um vetor pode separar as classes do problema perfeitamente, conforme na Figura 20.

Em casos como esse, um SVM com margens rígidas determina um hiperplano como a superfície de decisão que produza a separação máxima entre as classes, se-

gundo (HAYKIN, 1994). As margens deste hiperplano são os vetores suporte e são colocados sob as amostras mais adequadas como pode ser visto na Figura 20. A fórmula para o hiperplano é dada a seguir:

$$f(x) = w \cdot x + b = 0, \tag{4.16}$$

onde $w \cdot x$ é o produto escalar do vetor normal ao hiperplano *w* com um ponto do plano *x* somado ao coeficiente linear *b*.

Com esta fronteira é possível delimitar as duas classes do problema:

$$w \cdot x_i + b \geq 1 \rightarrow classe\ y = 1$$
$$w \cdot x_i + b \leq -1 \rightarrow classe\ y = -1.$$

Estas condições podem ser reescritas como:

$$y_i(w \cdot x_i + b) \geq 1, \forall\ 1 \leq i \leq n. \tag{4.17}$$

A distância entre as margens do hiperplano podem ser obtidas geometricamente por $\frac{2}{||w||}$, logo para maximizar a margem é preciso minimizar o denominador $||w||$.

A tarefa de encontrar o hiperplano ótimo em um SVM com margens rígidas consiste em encontrar o vetor normal *w* que satisfaça:

$$min\ \frac{1}{2}||w||^2, \tag{4.18}$$

sujeito a,

$$y_i(w \cdot x_i + b) \geq 1.$$

Assim em uma SVM com margens rígidas não se admite que as amostras de uma classe ultrapassem a fronteira de decisão, portanto SVM de margens rígidas não podem ser aplicadas a problemas não-linearmente separáveis.

Com a adição de um termo de folga ($C \sum_{i=1}^{n} \xi_i$) é permitida que uma instância possa estar após a fronteira de decisão. Possibilitando que as SVM com margens suaves possam convergir em uma solução em problemas não-lineares como pode ser visto na Figura 21. O hiperplano ótimo de uma SVM com margens suaves é definido a seguir:

$$min\ \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i, \tag{4.19}$$

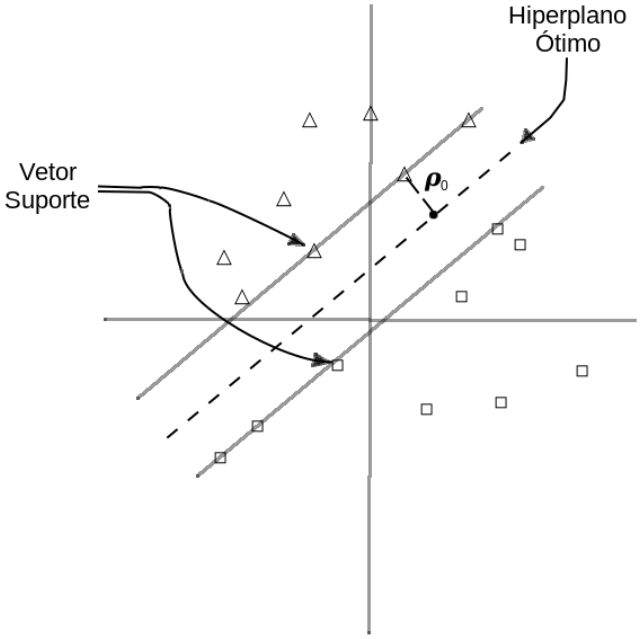


Figura 20 – SVM com margem rígida separando otimamente as classes.
Fonte: Adaptado de (HAYKIN, 1994).

sujeito a,

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0.$$

O termo *C* é a flexibilidade na escolha dos vetores suporte e ξ_i é o erro aceitável para cada instância para que o problema convirja.

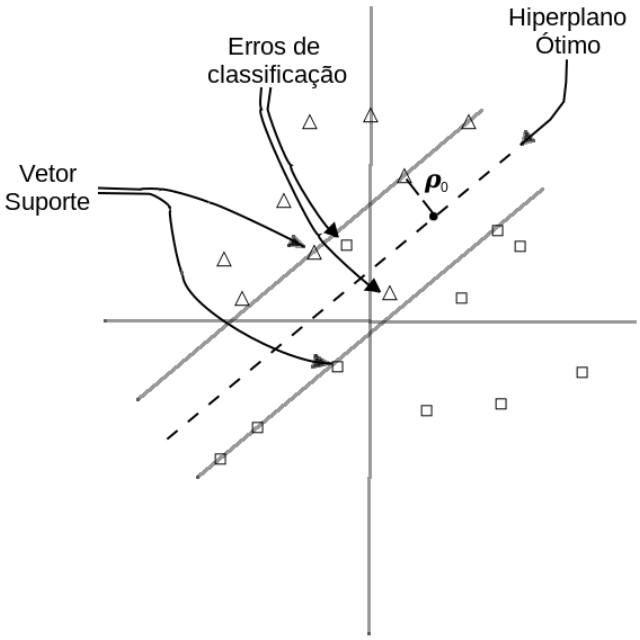


Figura 21 – Exemplo de um SVM com margens suaves.
Fonte: Adaptado de (HAYKIN, 1994).

Para classificar um problema não-linear uma SVM com margens suaves pode não ser suficiente para obter uma generalização satisfatória mesmo utilizando um hiperplano ótimo. Então é utilizada uma técnica conhecida como *truque do kernel* que visa mover o problema não-linearmente separável no seu espaço X para uma nova representação Z em alta dimensão, também chamado de espaço das características, em que muito provavelmente seja possível a sua separação utilizando uma SVM linear segundo (VLADIMIR, 1999).

Seja $\phi : X \rightarrow Z$ um mapeamento do espaço X em um espaço Z de maior dimensão como pode ser vista na Figura 22, então podemos escrever a relação de um *kernel* k que mapeia as amostras x_i e x_j no espaço de entrada X para um produto escalar das suas imagens no espaço de característica Z:

$$k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j). \tag{4.20}$$

O truque do kernel pode mover o problema para um espaço de alta dimensão inclusive onde cada amostra exista em uma dimensão independente das outras, mas geralmente é aplicada aos problemas linearmente separáveis a fim de se obter uma melhor generalização com uma melhor fronteira de decisão. Um simples exemplo de sua operação pode ser vista na Figura 22.

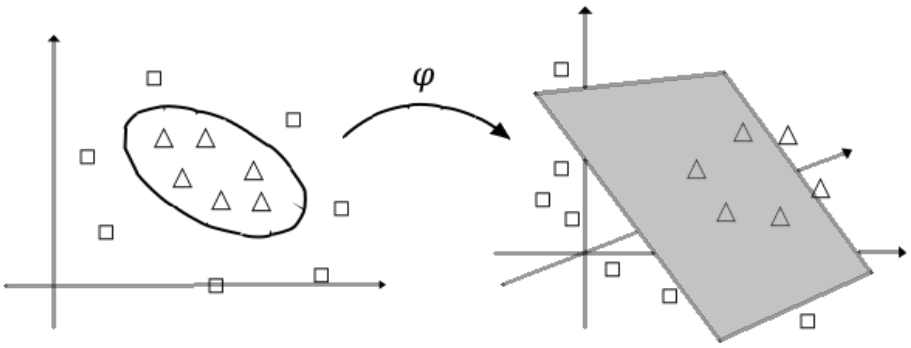


Figura 22 – Exemplo do funcionamento do truque do kernel.

Uma grande vantagem do SVM é que sua utilização não só é possível em vários problemas de classificação e regressão de alta dimensionalidade como os resultados alcançados geralmente são excelentes, isso se deve a função custo adotada ser convexa, ou seja a partir de qualquer par de pontos da função é possível desenhar uma semi-reta que não ultrapassa o espaço interno da função, logo o mínimo global pode ser determinado tratando-a como uma função quadrática.

Uma das desvantagens no uso de SVM é a necessidade de ajustes minuciosos nos seus hiperparâmetros, que podem funcionar bem para alguns problemas e muito ruim pra outros. Como por exemplo, a escolha da função *kernel* e do parâmetro de

flexibilidade *C*, que pode requerer um valor muito particular ao problema que quando mal escolhido pode diminuir o poder de generalização ocasionando *overfitting* (Figura 24) ou *underfitting* (Figura 23).

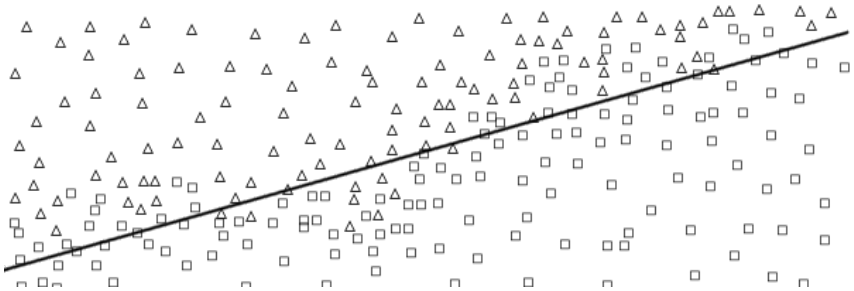


Figura 23 – SVM com *underfitting*.

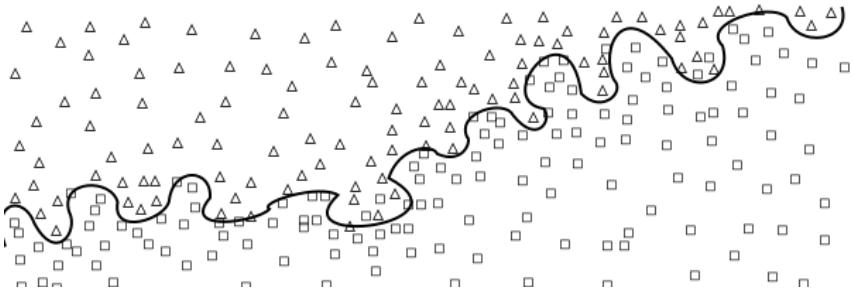


Figura 24 – SVM com *overfitting*.

4.6 Rede neural convolucional

As redes neurais convolucionais são redes neurais especializadas para a tarefa de reconhecimento de padrões em duas dimensões, segundo (HAYKIN, 1994). Suas raízes remontam de estudos sobre o sistema nervoso ligado aos receptores ópticos e na constatação de que estes neurônios se organizam em estruturas complexas que são composições de estruturas mais simples de neurônios.

Nestas redes a sua entrada se dá como uma matriz representando um plano em duas dimensões, como no caso de uma imagem que pode ser compreendida como uma matriz de pixels (vide Figura 25). Isso a difere dos outros algoritmos de AM avaliados neste trabalho que aceitam vetores de entrada. Devido a esta capacidade as redes convolucionais tem muitas aplicações no campo de reconhecimento de imagens.

Experimentos científicos realizados pelo ganhador do prêmio nobel David H. Hubel e Torsten Wiesel (HUBEL; WIESEL, 1959) demonstraram que neurônios situados na periferia do sistema nervoso visual de gatos (córtex visual) apenas disparavam suas sinopses quando recebiam o estímulo provocado por um objeto projetado a partir de um monitor de vídeo de uma certa forma e orientação mas apenas em algum local particular do campo visual (como pode ser visto na Figura 26), se comportando

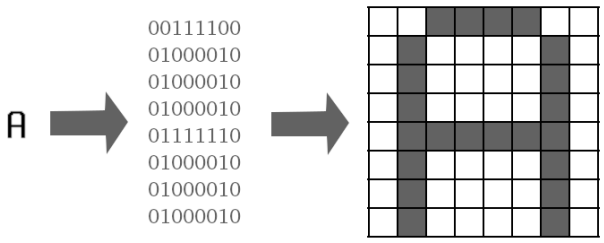


Figura 25 – Bitmap como uma matriz de dados.

de uma forma similar a um filtro para o objeto imóvel. Os neurônios localizados em camadas mais internas pareciam se comportar como combinações dos filtros anteriores. Podendo reconhecer, por exemplo, o mesmo objeto anterior mas apenas se ele estiver se deslocando em uma direção específica, disparando apenas quando ocorria o evento do objeto ter se movido naquela direção. Outra maneira de compreender este comportamento é que esses neurônios funcionavam como um sensor para certos atributos que um objeto pode apresentar.

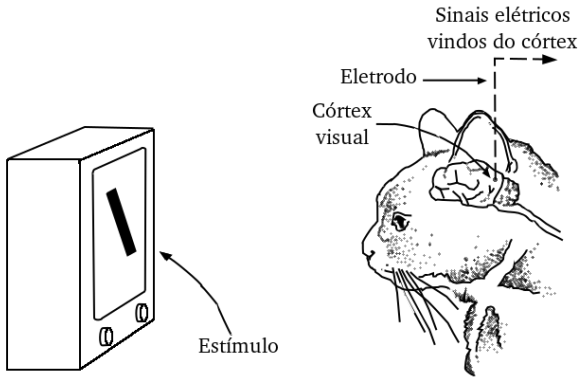


Figura 26 – Experimento de David Hubel e Torsten Wiesel.
Fonte: adaptado de David H. Hubel
https://goodpsychology.wordpress.com/2013/03/13/235/

David Hubel e Torsten Wiesel demonstraram que neurônios presentes no córtex visual de gatos respondiam a atributos específicos de uma imagem como ângulos, linhas, curvas e movimento.

Esta interessante característica foi representada como um layout alternativo para uma rede neural em que filtros podem ser combinados em camadas obtendo assim um comportamento similar. Uma rede convolucional pode ser visualizada na Figura 27.

Na camada convolucional basicamente há dois tipos de operações que podem ocorrer. A primeira operação chama-se de convolução e cria um mapa filtrado da entrada baseada na aplicação sucessiva de seus filtros. Estes filtros possuem duas dimensões e guardam o conhecimento da camada na forma de um padrão aprendido a partir dos dados de treinamento de forma muito similar ao pesos em uma MLP. Os

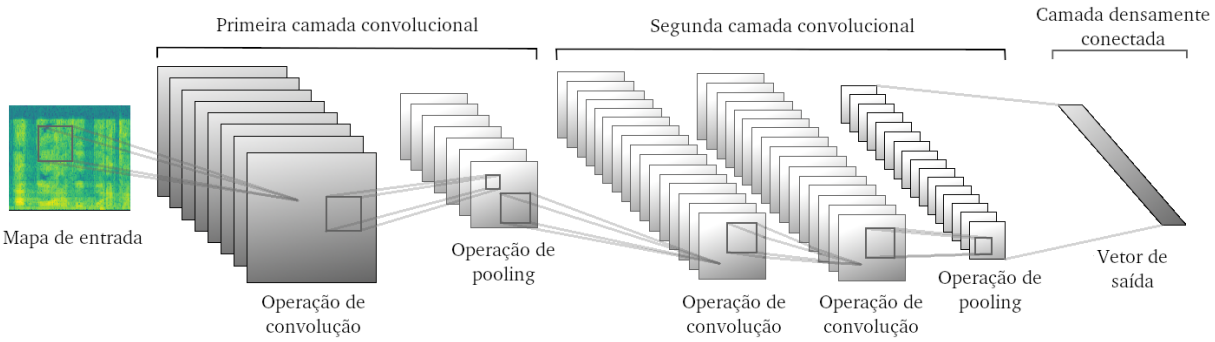


Figura 27 – Exemplo de um rede convolucional.

filtros possuem quantidade (*filters*) e tamanho bidimensional (*kernel size*) definidos por hiperparâmetros.

Inicialmente uma janela do tamanho definido para os filtros é selecionada nas coordenadas iniciais em ambas as dimensões da entrada e esta janela avança lateralmente em passos até que o fim do espaço seja atingido naquele sentido, fazendo com que a janela seja movida ao ponto inicial da primeira dimensão, mas avançando também em passos na segunda dimensão. Repetindo assim todo o processo até que o espaço em ambas as dimensões da entrada seja completamente percorrido.

A cada passo da convolução é aplicada uma operação de produto matricial escalar entre a matriz correspondente a janela selecionada naquele passo e o filtro ativo naquela iteração. Este processo acontece para cada filtro e todos os resultados dos produtos daquele filtro para cada janela do espaço de entrada figuram em um mapa que representa a entrada filtrada como pode ser observado na Figura 28.

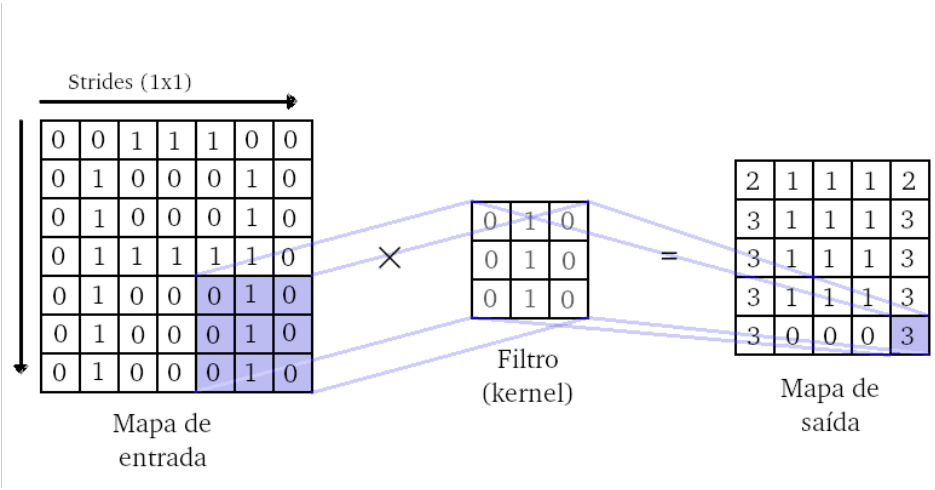


Figura 28 – Operação de convolução.

A segunda operação chama-se *pooling* e trata de diminuir por sub amostragem (*subsampling*) o tamanho do mapa criado pela operação de convolução mas tentando manter as informações mais relevantes. O seu funcionamento é parecido com o da camada convolucional, uma janela de tamanho definido por hiperparâmetro (*pool size*)

percorre todo mapa em passos (que é definido pelo hiperparâmetro *strides*). A sub amostragem se dá de modo que apenas o maior valor de cada quadrante visitado é copiado para o novo mapa (*maxpooling*) resultando em um representação de tamanho bem menor do que da entrada, seu funcionamento pode ser constatado na Figura 29.

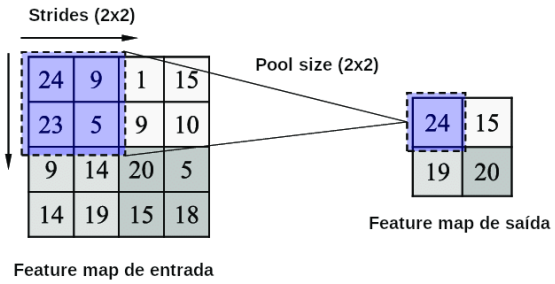


Figura 29 – Operação de maxpooling.

Mais camadas de convolução e pooling podem ser adicionadas na rede aumentando seu poder computacional e de modo a obter uma representação cada vez mais filtrada e resumida da entrada. Ao final de uma rede neural convolucional existe uma ou mais camadas densamente conectadas muito similar as camadas de um MLP sendo que a última camada é responsável pela classificação final das classes através da ativação softmax, que pode ser compreendido como um processo de votação.

Para este experimento foi adotado o seguinte layout para as camadas da ConvNet: Na primeira camada convolucional é aplicada o número de filtros especificado para o experimento, a função de ativação e uma operação de maxpooling. Se o modelo testado possui apenas uma camada convolucional é adicionada a camada densa de saída. Caso contrário, na segunda camada convolucional cada par operação de convolução e função de ativação subsequente é aplicada após a outra e tem sua quantidade de filtros dobrada em relação a operação anterior (não podendo ultrapassar um valor máximo de 256 filtros). Uma última operação de maxpooling é adicionada após a última função de ativação da última operação convolucional empilhada. Finalmente a camada densa com a ativação softmax é adicionada para fornecer a saída do modelo.

Esta rede processa as suas entradas de modo análogo a MLP. A primeira camada processa a entrada da rede e camada seguinte recebe a entrada da anterior em um processo que se repete até que se alcance a camada de saída. Logo as redes convolucionais assim como as MLPs são redes do tipo *feed forward*.

A etapa de *forward propagation* em uma camada convolucional se assemelha com a definição (4.7), substituindo-se a multiplicação por uma convolução, como pode ser visualizado na seguinte equação:

$$Z_{i,j} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \varphi_{anterior_{i+m,j+n}} W_{m,n} + B_{i,j}, \tag{4.21}$$

onde $Z_{x,y}$ é o elemento de índice (x, y) na matriz (mapa) de saída Z de tamanho $P \times Q$ da camada convolucional, $\varphi_{anterior}$ é o mapa de saída da camada anterior (entrada da rede), $W_{m,n}$ é o peso de índice (m, n) pertencente ao filtro convolucional de tamanho $M \times N$ e $B_{x,y}$ é o bias para o índice (x, y). Na operação de convolução a matriz de saída Z tem o mesmo tamanho da matriz de entrada.

A operação de *maxpooling* é bem simples e não realiza computação apenas reduz um espaço $M \times N$ para o maior valor dentro desse espaço. Seja o mapa X de tamanho $P \times Q$ então o *maxpooling*(X) produz um mapa de tamanho $(\frac{P}{M} \times \frac{Q}{N})$.

O treinamento desta rede também é realizada pelo algoritmo backpropagation e gradiente descendente. Como no caso de uma MLP, o backpropagation acontece da saída para a entrada da rede. Como as camadas densas ao final de uma ConvNet são idênticas a de uma MLP, o treino ocorre exatamente da mesma maneira nestas camadas. De tal modo que os pesos de toda rede densa sofrem ajustes com auxílio da regra Delta, descrita na equação (4.11).

Nas operações de *pooling*, por não se realizarem computação (e assim não possuírem pesos), não são realizados ajustes. Porém o neurônio responsável pela propagação do estímulo durante a etapa de *forward propagation* é utilizado para propagar o gradiente do erro proveniente das camadas posteriores, logo seus índices são reutilizados na etapa de treinamento permitindo que o gradiente seja propagado apenas para os neurônios relevantes.

Nas operações convolucionais o algoritmo backpropagation é um pouco diferente: Os filtros convolucionais sofrem ajustes nos seus valores em relação aos neurônios ativados na camada posterior, porém de maneira diferente ao de uma rede densamente conectada que possui pesos individuais para cada entrada, em uma operação de convolução uma região de neurônios próximos são conectados localmente a um neurônio no mapa de saída através do filtro que no processo de convolução desliza por todo o espaço de entrada. Logo, os pesos (elementos) da matriz filtro vão sendo compartilhados entre vários neurônios distribuídos no mapa de entrada, de modo que durante a etapa de treinamento apenas os neurônios que foram afetados por um peso específico é que tem seus erros propagados somados, resultando em um erro total que é utilizado para o cálculo da parcela de ajuste naquele determinado peso. A fórmula para o cálculo do gradiente de um peso em uma operação convolucional pode ser vista abaixo:

Seja $W_{i,j}$ o peso de índice (i, j) de um filtro convolucional de tamanho $M \times N$,

então:

$$\Delta E_{w_{i,j}} = \sum_{p=0}^{P-M} \sum_{q=0}^{Q-N} \frac{\partial E}{\partial \varphi_{p,q}} \times \frac{\partial \varphi_{p,q}}{\partial z_{p,q}} \times \frac{\partial z_{p,q}}{\partial w_{i,j}}. \tag{4.22}$$

O termo $\frac{\partial E}{\partial \varphi_{p,q}}$ é o erro dispersado pela camada posterior e já foi obtido durante seu treinamento. Já o termo $\frac{\partial \varphi_{p,q}}{\partial z_{p,q}}$ pode ser obtido derivando a função de ativação quando sua entrada é $z_{p,q}$. Por último, o termo $\frac{\partial z_{p,q}}{\partial w_{i,j}}$ pode ser obtido derivando a definição (4.21) em relação a variável $w_{m,n}$.

Assim o peso $W_{i,j}$ pode sofrer um ajuste com gradiente descendente:

$$w_{i,j\ ajustado} = w_{i,j} - \alpha \Delta E_{w_{i,j}}. \tag{4.23}$$

Por último é calculado o erro propagado em relação ao mapa de entrada da camada convolucional para que suas camadas restantes recebam uma parcela do gradiente. O erro $\frac{\partial E}{\partial \varphi_{anterior}}$ é definido por:

$$\frac{\partial E}{\partial \varphi_{anterior_{i,j}}} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \frac{\partial E}{\partial z_{i-m,j-n}} \times \frac{\partial z_{i-m,j-n}}{\partial \varphi_{anterior_{i,j}}}. \tag{4.24}$$

O termo $\frac{\partial z_{i-m,j-n}}{\partial \varphi_{anterior_{i,j}}}$ pode ser resolvido derivando-se a equação (4.21) em relação a variável $\varphi_{anterior_{i,j}}$, obtendo-se $w_{m,n}$, logo:

$$\frac{\partial E}{\partial \varphi_{anterior_{i,j}}} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \frac{\partial E}{\partial z_{i-m,j-n}} w_{m,n}. \tag{4.25}$$

Assim o algoritmo que treina uma rede convolucional é em parte a mesma de uma rede neural apenas se diferenciando nas camadas convolucionais, o seguinte

pseudocódigo descreve o treinamento de uma rede convolucional:

Algoritmo 3: Backpropagation em redes convolucionais.

Entrada: Quantidade de épocas E, Matriz de amostras M, vetor de rótulos L, taxa de aprendizado T e a topologia de rede N.

Saída: Rede treinada N.

```
início
    IniciaPesos(N)
    epoca = 0
    repita
        para cada amostra, rotulo ∈ (M, L) faça
            previsao = ForwardPropagate(amostra)
            erro = FunçãoCusto(previsao, rotulo)
            para cada camada ∈ N da última para a primeira faça
                #Checa se a camada é convolucinal
                se CamadaConvolucinal(camada) = Verdadeiro então
                    para cada filtro ∈ camada faça
                        para cada peso ∈ filtro faça
                            peso = peso − T × RegraDeltaConv(peso, erro)
                        fim
                    fim
                senão
                    #A camada é densa
                    para cada peso ∈ camada faça
                        peso = peso − T × RegraDelta(peso, erro)
                    fim
                fim
            fim
        fim
        epoca = epoca + 1
    enquanto epoca < E;
fim
retorna N
```

Em uma rede convolucional os pesos existentes nos filtros são inicializados ao aleatórios e devido a esta característica tendem a aprender diferentes padrões interessantes existentes nos dados de treino.

Os hiperparâmetros (quantidade de épocas, quantidade de filtros, tamanho da janela de convolução, tamanho da janela de pooling, strides entre outros) devem ser cuidadosamente ajustados para cada aplicação e possuem grande influência no seu

poder de generalização.

Assim sendo pode-se afirmar que as redes neurais convolucionais exploram as correlações espaciais presentes (relação de vizinhança) nos dados de entrada e é capaz de criar representações robustas a distorções diversas como deslocamentos. Agindo como filtros para os atributos mais relevantes ao problema que são aprendidos durante o treinamento.

Uma vantagem percebida é que as redes convolucionais treinam relativamente bem mais rápido (com menos épocas) que as redes neurais, sendo o treinamento com parada prematura uma boa maneira de evitar baixo poder de generalização ocasionado por um treino muito longo.

4.7 Medidas de desempenho

A métrica utilizada para a avaliação de todos os modelos foi a precisão que pode ser entendida como a proporção de acertos do classificador na tarefa de classificação em instâncias novas (inéditas).

A precisão é definida em função dos acertos na classificação do rótulo de uma instância (verdadeiros positivos ou tp) e dos erros na classificação do rótulo (falsos positivos ou fp) das instâncias ou seja as vezes que uma amostra foi classificada como sendo de um rótulo mas deveria ser um outro. Uma outra maneira de pensar essa métrica é como a proporção dos acertos sob todas as instâncias classificadas como de um tal rótulo. Sua fórmula pode ser visualizada a seguir:

$$P = \frac{tp}{tp + fp}. \tag{4.26}$$

No contexto deste trabalho, quando o modelo apresentar uma boa precisão, isto significará que a razão de vezes que as amostras detectadas como sendo de uma certa emoção quando realmente pertencem àquela classe é bem mais alta do que as vezes que são classificadas como sendo de outra emoção em amostras inéditas. O que ajuda a medir a qualidade da previsão em termos de relevância, evidenciando as situações em que o modelo não generaliza bem. A precisão é adequada para problemas onde as classes estão desbalanceadas nos dados de treino permitindo assim medir a relevância das previsões para cada emoção.

4.8 Função de erro

A função de erro utilizada neste experimento é o *cross-entropy* também conhecido como *log loss* que tenta demonstrar as perdas na classificação quando a probabi-

lidade encontrada para uma amostra de pertencer a sua classe correta (que deve ser um valor entre 0 e 1) se difere muito do valor correto esperado para aquela amostra, que no caso seria 1 para a classe correta. Assim quanto maior o valor do cross-entropy maior a distância com o rótulo desejado como pode ser observado na Figura 30. Esta é a métrica mais adequada para problemas de classificação multi-classes e quando aplicada nesse cenário tem os valores para cada classe somados. Observe que o valor de log loss aumenta quando o valor calculado para a classe se afasta de 1.

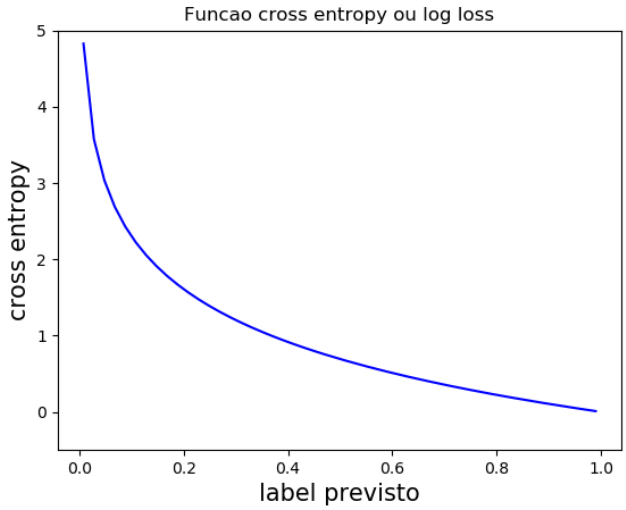


Figura 30 – Aparência da função cross entropy ou log loss.

A equação do cross-entropy é apresentada a seguir:

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c},), \tag{4.27}$$

onde M é a quantidade de classes do problema de classificação, $y_{o,c}$ assume o valor 1 se a amostra O pertencer a classe C e 0 caso contrário e $P_{o,c}$ é a probabilidade calculada da amostra O pertencer a classe C.

4.9 Considerações sobre o volume de dados

A relação da performance obtida por algoritmos de AM com fatores como o tamanho da amostra disponível para o treino já é bem conhecida, contando com alguns estudos que discutem suas implicâncias. Segundo (RAUDYS; JAIN, 1991) o planejamento da quantidade de atributos, o tamanho da amostra, a complexidade do algoritmo de treinamento e a complexidade inerente ao problema são os fatores principais que impactam diretamente na performance de um modelo AM.

A maior dificuldade vem do fato que diversas vezes o volume de dados anotados disponível para estudo é limitado e de difícil obtenção, assim as condições de treinamento com dados limitados é bem frequente e em geral induzem modelos que não generalizam bem.

O efeito conhecido como *curse of dimensionality* (RAUDYS; JAIN, 1991) pode ser visualizado através do estudo da relação da performance obtida com um modelo em relação a quantidade de amostras de treino, também chamada de curva de aprendizagem. A aparência de uma curva de aprendizagem genérica para algoritmos de AM pode ser visualizada na Figura 31.

Figura 31 – Curvas de aprendizado genéricas de algoritmos de AM.

Uma das consequências sobre os efeitos do *curse of dimensionality* é a conclusão contra intuitiva de que dado um problema de classificação com dados de treino finitos quanto mais atributos são adicionados para diferenciar as classes maior a performance, até o ponto que a quantidade de dados disponível se torna o fator limitante. Pois a cada nova dimensão de dados adicionada ao problema a distância relativa de cada amostra vai se tornando maior até alcançar o ponto que o classificador passa a capturar os detalhes menos importantes destas amostras esparsas, ocasionando *overfiting*. O volume de dados teriam que aumentar exponencialmente em quantidade apenas para manter a densidade de distribuição e garantir ganho de performance com aquela nova dimensão.

Para o treino de algoritmos de AM mais intensivos como as redes convolucionais (pertencendo a classe *deep learning*) é vital um grande volume de dados devido a quantidade maior de dimensões a que estes problemas geralmente são submetidos. Assim é introduzida a técnica de *data augmentation* a fim de se aumentar o tamanho do banco de dados com novas instâncias geradas a partir das instâncias existentes nele.

4.10 Uma técnica de data augmentation para aplicações de SR

A técnica de *Data Augmentation* aplicada foi inspirada no algorítmo SpecAugment da google (PARK et al., 2019), técnica que foi capaz de resultados de estado da arte em aplicações com processamento de fala.

O SpecAugment foi desenvolvido para aplicações que utilizam classificação de áudio baseadas em espectrograma e introduz distorções na forma de atributos espectrais ou janelas de tempo que são ao aleatório substituído por valores médios. Estas novas instâncias ainda são das mesmas classes das instâncias originárias mas são ligeiramente diferentes. Estas alterações permitem que o modelo se torne também mais resistente a dados incompletos e outros ruídos que podem ocorrer em instâncias reais.

Um exemplo de um espectrograma antes e após a aplicação dos métodos de *Data Augmentation* pode ser visto na Figura 32. Pode-se notar que nas regiões em cinza os valores originais foram removidos forçando o classificador a encontrar diferentes representações para a nova amostra.

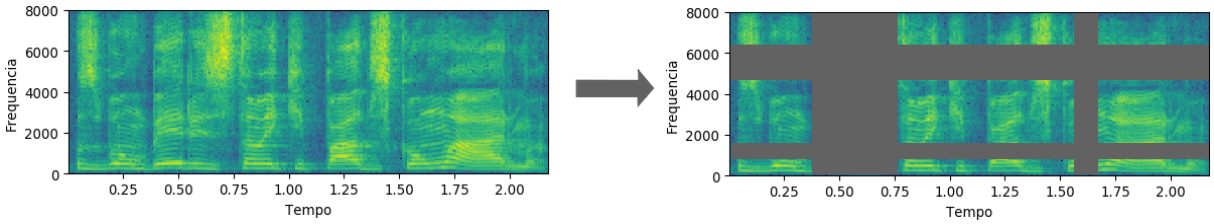


Figura 32 – Espectrograma antes e após a aplicação do SpecAugment.

4.11 Avaliação dos modelos

Após a definição de uma métrica será calculado um resultado para cada algoritmo em um processo de validação cruzada, ao fim é obtido uma média e o desvio padrão destes resultados que é armazenado para cada experimento.

Como cada experimento é com um classificador diferente e que não se tem acesso a toda população mas sim a uma pequena parte, será suposto que as populações do problema são de distribuições diferentes e desconhecidas. Como os resultados vêm de populações diferentes eles não podem ser comparadas diretamente. Assim todos os resultados têm de ser submetidos a um teste de hipótese t-student aos pares, possibilitando que médias às vezes muito próximas possam ser comparadas com alguma relevância estatística.

4.12 Trabalhos relacionados

Após uma revisão da literatura foram encontrados alguns trabalhos que utilizam uma abordagem semelhante à que é proposta aqui e que serviram como base para alguns parâmetros deste experimento.

Em (MITRA; FRANCO, 2015) é apresentada uma abordagem comparativa utilizando redes neurais densas e redes convolucionais onde atributos de espectro da frequência quanto do domínio do tempo foram empregados e se observou uma melhor capacidade da rede convolucional em criar representações invariantes a partir da voz de cada locutor. Minimizando os efeitos que fatores como as diferenças corporais e do aparelho vocal causado por diferentes locutores impactam na performance final do classificador, permitindo uma maior robustez e por consequência melhor performance do classificador em amostras do mundo real onde há diversos locutores. O modelo deste trabalho aplica uma metodologia similar divergindo no layout adotado por não possuir um rede neural densa nas camadas de saída e apenas considerar os atributos do espectro de frequência por motivo de orçamento (custo computacional) disponível para o presente trabalho.

Em (ZHANG et al., 2017), é proposto uma rede convolucional capaz de criar representações temporais a partir de mapas de atributos. Sua performance foi comparada a redes neurais recorrentes na tarefa de reconhecimento de sentenças (sequências de caracteres) em uma imagem. A capacidade de uma rede convolucional de modelar sequências temporais era dada como incerta até então, mas com o experimento ficou claro a capacidade destas redes em problemas que necessitam de alguma capacidade de memória de curto prazo. No presente trabalho o modelo adotado tem uma camada convolucional com operações de convolução sobrepostas de modo a replicar esta característica, contudo o modelo adotado omite as camadas densas antes da camada de softmax por motivos de orçamento.

Duas redes convolucionais inspirada na VGGNet (SIMONYAN; ZISSERMAN, 2014) são demonstradas e atingem resultados de estado da arte na tarefa de reconhecimento de eventos sonoros em (TAKAHASHI et al., 2016). Este artigo inclui um argumento de como com mais camadas convolucionais com tamanho de kernel fixo em 3x3 pode-se contribuir para uma menor quantidade de parâmetros para a rede densa e maior não linearidade mantendo a mesma capacidade de filtros maiores em uma única camada. Para este estudo o tamanho do kernel de 3x3 junto com os tamanhos da janela de pooling (1x2, 2x1 e 2x2) foram adotadas seguindo o modelo proposto por (TAKAHASHI et al., 2016). O método de *data augmentation* empregado em (TAKAHASHI et al., 2016) que consiste em se misturar duas amostras da mesma classe na composição de uma nova amostra não pôde ser empregada no modelo proposto neste estudo por causa da natureza dos problemas que são essencialmente diferentes.

A estratégia de se misturar duas amostras resultaria em dois discursos sobrepostos que tornaria a instância de difícil compreensão, impedindo sua utilização.

Em (WU; FALK; CHAN, 2011) máquinas de vetor suporte são treinadas com diferentes métodos de extração de atributos e seus resultados são comparados na tarefa de classificação de emoções na fala, embora não tenha obtido os melhores índices em todas as avaliações, atributos espectrais como MFCC foram incluídos na avaliação e obtiveram resultados significantes sendo o segundo melhor conjunto de atributo na maioria dos experimentos relatados. O presente trabalho adota uma metodologia muito similar a deste estudo: as amostras do banco possuem 7 rótulos possíveis, os dados foram apreciados por um comitê de jurados e a avaliação dos modelos utiliza um processo de validação cruzada com 10 folds. Algumas diferenças incluem os rótulos adotados pelo Berlin Emotion Database (BURKHARDT et al., 2005) que lista 6 das 7 emoções utilizados no presente trabalho, mas inclui ”boredom” no lugar que seria de ”surprise”, logo os seus resultados não são totalmente comparáveis. A métrica adotada é conhecida como *Fisher discriminant ratio* e foi escolhida por melhor discriminar o conjunto de atributos mais ruidosos, de acordo com os objetivos da avaliação de performance dos atributos em (WU; FALK; CHAN, 2011).

5 Experimentos

Nesta seção será descrito cada etapa do experimento em detalhes até a obtenção dos resultados. Para a construção dos experimentos foi utilizada a linguagem de programação Python[1] e as bibliotecas scikit-learn[2], Keras[3] e TensorFlow[4].

5.1 Bibliotecas de software

O scikit-learn (PEDREGOSA et al., 2011) é uma biblioteca em python que implementa a maioria dos algoritmos de AM e diversas outras ferramentas para tornar mais fácil a pesquisa e a resolução de problemas utilizando técnicas de AM. Possuindo funcionalidades que lidam com as atividades relacionada à validação cruzada, métricas, funções de erro, pré-processamento entre outras. Atividades estas que às vezes se tornam muito volumosas ou não-triviais já que são adjacentes a tarefa de classificação de um problema.

Embora muito completo e no geral muito eficiente, o scikit-learn não possui suporte a redes convolucionais ou aceleração de hardware, condições que inviabilizariam este experimento em sua totalidade, assim será utilizado em conjunto uma outra biblioteca em python que implementa uma API muito similar a do scikit-learn e com a possibilidade de criação de modelos compatíveis com as suas ferramentas: O Keras. O Keras (CHOLLET et al., 2015) é uma *framework* que descreve em alto nível de abstração modelos de AM mas não implementa as operações de baixo nível como as operações matriciais agindo então como um *wrapper* para outras bibliotecas (*backends*) que implementam estas operações de baixo nível, como é o caso do Tensorflow (ABADI et al., 2015).

O Tensorflow é um projeto que foi aberto em 2015 para comunidade opensource mas que foi desenvolvido originalmente para uso interno pela equipe responsável pelo Google Brain[5], tem uma API complexa mas que permite maior flexibilidade no controle de cada aspecto dos modelos e suporte a aceleração em várias arquiteturas como em GPUs (Graphics processing units), mais popularmente conhecidas como placas de vídeo com aceleração 3D voltadas à jogos.

A escolha de se combinar a facilidade proporcionada pelo ferramental do scikit-

[1] https://www.python.org/
[2] https://scikit-learn.org/stable/
[3] https://keras.io/
[4] https://www.tensorflow.org/
[5] https://ai.google/research/teams/brain/magenta/

Biblioteca	Versão	Descrição	Aplicação
libROSA	0.6.2	Manipulação de áudio	Pré-processamento.
scikit-learn	0.20.3	Aprendizado de máquina	Seleção de modelos, métricas e modelos de AM (AD, NB, MLP, e SVM).
Keras	2.2.4	Prototipagem rápida de redes neurais profundas	Modelo de aprendizado de máquina profundo (ConvNet).
TensorFlow	1.12.1	Implementação de baixo nível para computação de grafos	Utilizado pelo Keras para implementação da ConvNet.
CUDA	10	Implementação de baixo nível para computação em GPU	Utilizado pelo TensorFlow para treino da rede com processamento paralelo.

Tabela 3 – Lista de API utilizadas neste experimento e suas aplicações.

learn com modelos acelerados por GPU obtidas do uso do Keras com o TensorFlow se mostrou muito proveitosa e serviu para que todos os experimentos fossem realizados em um processo idêntico, quer dizer com um mesmo fluxo de operações independente se o modelo era um classificador keras ou um classificador do próprio scikit-learn. Na Tabela 3 é listada as versões das bibliotecas utilizadas e um resumo das suas aplicações neste experimento.

5.2 Conjunto de dados

Foi utilizado o VERBO Database[6] neste trabalho que é uma base de dados que contém 520 clips de áudio e seus respectivos rótulos. Este banco foi desenvolvido para pesquisas na área de AM provendo também um artigo (NETO et al., 2018) de onde as classes puderam ser obtidas.

A quantidade de amostras em cada classe no banco de dados pode ser visualizada através do histograma presente na Figura 33.

5.3 Pré-processamento

O pré-processamento dos clipes de áudios foi realizado na seguinte ordem:

1. Carregamento dos arquivos referidos usando API do librosa[7];
2. Aplicação de um filtro pré-ênfase;

[6] https://sites.google.com/view/verbodatabase/
[7] https://librosa.github.io/librosa/

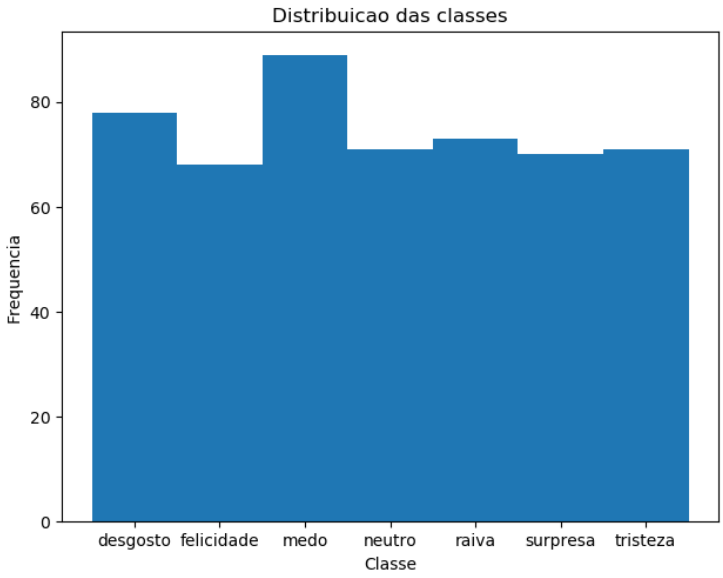


Figura 33 – Distribuição das classes no banco VERBO.

3. Criação de janelas de processamento (através da aplicação da função Hamming disponível no librosa);
4. Transformada de fourier (realizada automaticamente[8] pela biblioteca librosa);
5. Extração de atributos (função mfcc disponível no librosa) para cada janela;
6. Ajustes do tamanho final do vetor de atributo.

Uma versão em pseudocódigo da rotina que realizou o pré processamento pode

[8] embora realize automaticamente a transformada de fourier, foi optado por se fazer manualmente esta etapa pois ela é realizada em conjunto com as definições da função de enjanelamento no librosa.

ser vista a seguir:

Algoritmo 4: Pré-processamento.

Entrada: Amostras a serem pré-processadas A e quantidade de amostras Q.
Saída: Matriz de atributos $M(Qx12000)$

início
  $M = [\,]$
  para *cada* $amostra \in A$ **faça**
    $m = [\,]$
    $p =$ Pré-ênfase$(amostra, 0.95)$
    para *cada* $janela \in$ Hamming(p) **faça**
      $t =$ Transformada$(janela)$
      $coeficientes = 24$
      $m \leftarrow m +$ MFCC$(t, coeficientes)$
    fim
    se Tamanho$(m) < 500$ **então**
      $m = m +$ ***JanelasVazias***$(500 - m)$
    fim
    se Tamanho$(m) > 500$ **então**
      $m =$ ***DescartaJanelas***$(m - 500)$
    fim
    $M \leftarrow M +$ ConverteEmVetor(m)
  fim
fim
retorna M

Após o carregamento do arquivo, Primeiramente é aplicado um filtro de pré-ênfase ajustado para α = 0.95. Durante a revisão da literatura foram sugeridos os valores de 0.95 e 0.97.

Em seguida, a função hamming é aplicada como um parâmetro do librosa[9] para a definição da segmentação do som em janelas, mais uma vez a literatura sugere que o tamanho da janela esteja entre 10 a 100 ms (milissegundos) com overlap (ou sobreposição) de cerca de 20% entre as janelas. O valor da janela escolhido foi de 25 ms com sobreposição de 20% (5 ms).

A transformada de fourier é o processo utilizado para mudar a representação de um sinal do domínio do tempo para sua representação no domínio das frequências, esta operação é realizada pelo módulo *core.stft*[10] *(short term fourier transform)* do librosa.

[9] https://librosa.github.io/librosa/
[10] https://librosa.github.io/librosa/generated/librosa.core.stft.html

Por fim as amostras são extraídos para cada janela de tempo com auxílio da função *feature.mfcc*[11] disponível no librosa e ao final todos os valores são retornados como um uma matriz: Em um eixo são os coeficientes MFCCs e no outro eixo se refere a janela temporal que o vetor foi extraído.

5.4 Data Augmentation

O fato de que no banco VERBO só existem 520 amostras para as 7 classes possíveis torna o problema mais difícil para a extração de um modelo que generaliza bem, este efeito foi percebido na baixa performance inicial de todos os modelos avaliados com este banco (como pode ser visto na Tabela 8. Uma possível solução para este quadro é relatado no experimento de (TAKAHASHI et al., 2016), na ausência de um banco de dados maior, a técnica de *data augmentation* pode ser empregada para artificialmente criar novas instâncias do problema, assim aumentando drasticamente o tamanho dos dados disponíveis. Com uma maior quantidade de dados se espera uma melhora na performance dos algoritmos de AM aplicados ao problema em questão.

O problema de classificação deste estudo possui 12000 parâmetros (na forma de uma matriz 24x500) e um banco com 520 instâncias, após este procedimento foi possível a duplicação do tamanho do banco disponível para o treino e avaliação dos modelos e assim, a princípio, possibilitando melhores resultados em relação ao experimento com apenas as instâncias originais do problema, devido a melhor relação amostras / atributos obtida.

Da proposta original do SpecAugment (PARK et al., 2019) apenas os módulos de *frequency mask* e *time mask* foram implementados, sendo o módulo *time warp* não utilizado devido a sua distorção que implicava que em um ponto arbitrário no tempo toda a porção inicial até aquele ponto fosse trocada com a porção restante. O que resultaria em um evento fora de ordem e possivelmente em uma instância muito artificial e sem sentido para o problema.

Antes da fase de treinamento cada atributo é normalizado com a ajuda da função StandartScaler[12] do scikit-learn, que tem o objetivo de normalizar os dados para que a média seja zero (0) com desvio padrão um (1).

[11] https://librosa.github.io/librosa/generated/librosa.feature.mfcc.html
[12] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

5.5 Seleção e avaliação dos modelos aninhada

A seleção de modelos foi auxiliada pela utilização da técnica de procura em grade implementada no scikit-learn pela função GridSearchCV[13], que visa incluir no processo de treinamento do modelo os seus hiperparâmetros.

Na busca em grade, o treinamento dos modelos se dá de maneira consecutiva com combinações de hiperparâmetros de modo exaustivo dentro das permutações possíveis. Nas Tabelas 4, 5, 6 e 7 é possível verificar os hiperparâmetros pesquisados em grade por cada classificador. Também são informadas as partições nos dados a serem utilizados na validação cruzada. Em seguida, a combinação de hiperparâmetros que obtêm a pontuação mais baixa quando aplicada a função de erro entre as saídas esperadas e as obtidas no conjunto de validação é a combinação selecionada por ter melhor resolvido o problema.

O treinamento é interrompido ao fim de todas as épocas ou quando o critério de parada específico do algoritmo for alcançado, podendo ainda ser interrompido prematuramente quando a função erro parar de convergir, situação conhecida como *early stopping*, suportada no scikit-learn como um hiperparâmetro e no keras como um call-back (uma função sobre-escrita que é executada ao final de cada época). A parada prematura é uma eficiente técnica para evitar o *overfitting* do modelo.

Para os splits foi utilizado o StratifiedKfold[14] com 10 splits. Assim cada partição de dados vai preservar distribuição das classes do problema em cada etapa de teste.

O melhor classificador com os hiperparâmetro já selecionados é obtido para cada um dos algoritmos no conjunto de treino e validação. O próximo passo é realizar um novo processo de validação cruzada no conjunto de teste a fim de encontrar o erro de generalização destes melhores modelos. Esta rodada da validação cruzada aninhada é realizada pelo loop de validação mais externa e tem uma partição de dados estratificados do mesmo modo do utilizado na pesquisa em grade mas utiliza a métrica da precisão para a pontuação final. Este processo de realizar duas validações cruzadas aninhadas também é conhecida como *nested cross-validation* e pode ser visualizado na Figura 34.

5.6 Resultados

Esta seção traz as informações de como os modelos pontuaram de acordo com a sua precisão na classificação do conjunto de dados de teste correspondendo ao seu poder de generalização. Os valores a seguir são calculados dos resultados das

[13] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[14] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

Hiperparâmetro	**Valores**				
critério de avaliação do atributo	entropia				
quantidade mínima de amostras	2	5	10	25	50
quantidade máxima de nodos	2	5	10	25	50

Tabela 4 – Hiperparâmetros para as árvores de decisão.

Hiperparâmetro	**Valores**				
função de ativação	tanh		relu		
algoritmo de treino	gradiente descendente (*sgd*)				
taxa de aprendizado	0.001	0.01	0.1	0.5	
número de neurônios na camada escondida	10	20	40	80	100
épocas	100		500		
taxa de inércia (*momentum*)	0.8				
parada prematura	Sim		Não		

Tabela 5 – Hiperparâmetros para o MLP.

Hiperparâmetro	**Valores**			
função kernel	linear	poly		rbf
gamma	10**-4	10**-2	1	10**2
flexibilidade (*C*)	10**-2	1	10**2	10**4
degree	2		3	

Tabela 6 – Hiperparâmetros para o SVM.

Hiperparâmetro	**Valores**				
épocas	500				
tamanho do lote (*batch_size*)	20				
número de operações convolucionais	1	2	3	4	5
quantidade de filtros	8	16	32	64	
tamanho da janela de kernel	(3,3)				
tamanho da janela de *pooling*	(1,2)	(2,1)		(2,2)	
função de ativação	relu	tanh	sigmoid		
taxa de aprendizado	0.001		0.01	0.1	0.5
taxa de inércia (*momentum*)	0.8				

Tabela 7 – Hiperparâmetros para a ConvNet.

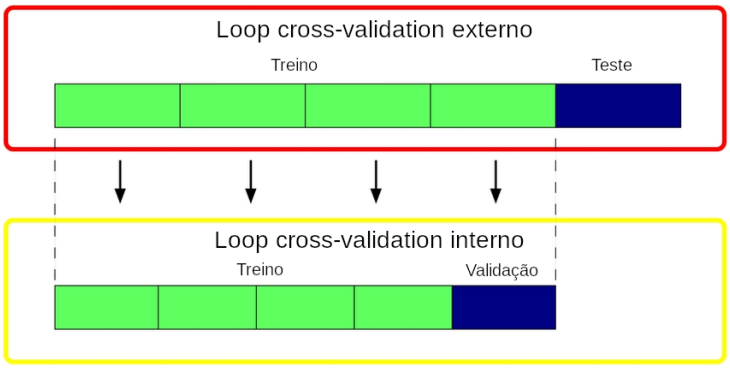


Figura 34 – Validação cruzada aninhada.

Classificador	Média	Desvio padrão
Naive Bayes	0.20	0.05
Árvore de decisão	0.23	0.05
MLP	0.29	0.05
SVM	0.28	0.04
ConvNet	0.31	0.06

Tabela 8 – Resultados do experimento 1 (sem *data augmentation*).

Classificador	Média	Desvio padrão
Naive Bayes	0.41	0.02
Árvore de decisão	0.56	0.06
MLP	0.94	0.04
SVM	0.93	0.03
ConvNet	0.86	0.04

Tabela 9 – Resultados do experimento 2 (com *data augmentation*).

rodadas de validação cruzada externa (as precisões obtidas para cada *fold* de dados) de onde foi calculada a sua média e desvio padrão.

O classificador Naive Bayes obteve uma precisão de apenas 20% no experimento sem *data augmentation* mas pontuou 41% no segundo experimento (com *data augmentation*) como pode ser visto na Tabela 9 e foi o pior resultado entre os algoritmos comparados.

A melhor árvore de decisão encontrada alcançou uma precisão de cerca de 23% na primeira rodada (sem *data augmentation*) e alcançou aproximadamente 56% na segunda (com *data augmentation*) como pode ser visto nas Tabelas 8 e 9.

A rede neural (MLP) obteve uma precisão de 29% no experimento sem *data augmentation* (Tabela 8) o que é consideravelmente baixo, mas no experimento com *data augmentation* conseguiu classificar aproximadamente 94% das instâncias corretamente, sendo o melhor resultado entre os classificadores tradicionais, como pode

Hiperparâmetro	**Valores**
critério de avaliação do atributo	entropia
quantidade mínima de amostras	25
quantidade máxima de nodos	25

Tabela 10 – Hiperparâmetros obtidos para a árvore de decisão (sem *data augmentation*).

Hiperparâmetro	**Valores**
função de ativação	logistic
algoritmo de treino	gradiente descendente (*sgd*)
taxa de aprendizado	0.5
hidden_layer_sizes	40
épocas	500
taxa de inércia (*momentum*)	0.8
parada prematura	Sim

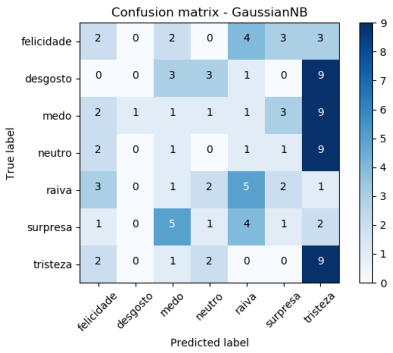
Tabela 11 – Hiperparâmetros obtidos para o multi layer perceptron (sem *data augmentation*).

Hiperparâmetro	**Valores**
função kernel	rbf
gamma	0.0001
flexibilidade (*C*)	100
degree	2

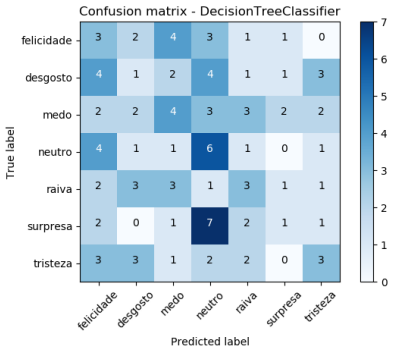
Tabela 12 – Hiperparâmetros obtidos para a máquina de vetor suporte (sem *data augmentation*).

Hiperparâmetro	**Valores**
épocas	500
tamanho do lote (*batch_size*)	20
número de operações convolucionais	3
quantidade de filtros	64
tamanho da janela de kernel	(3,3)
tamanho da janela de *pooling*	(1,2)
função de ativação	tanh
taxa de aprendizado	0.01
taxa de inércia (*momentum*)	0.8

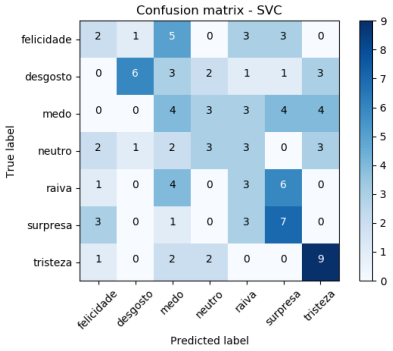
Tabela 13 – Hiperparâmetros obtidos para a rede convolucional (sem *data augmentation*).



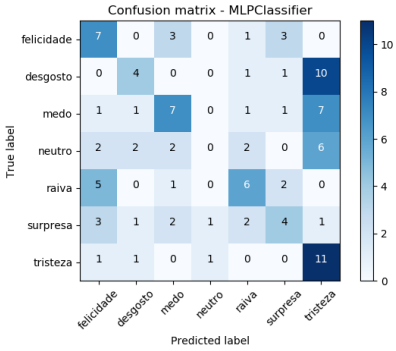
(a) Naive Bayes.



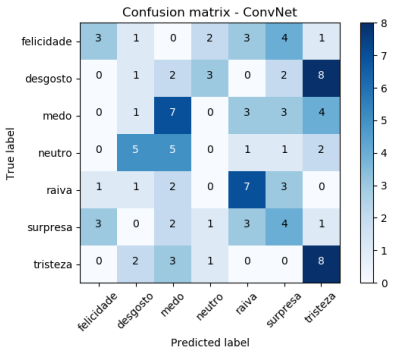
(b) Árvore de decisão.



(c) Máquina de vetor suporte.



(d) Multi layer perceptron.



(e) Rede convolucional.

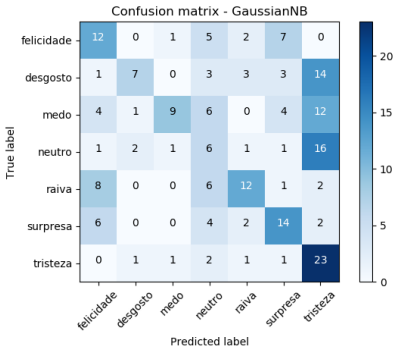
Figura 35 – Matrizes de confusão para os experimentos sem *data augmentation*.

ser visto na Tabela 9.

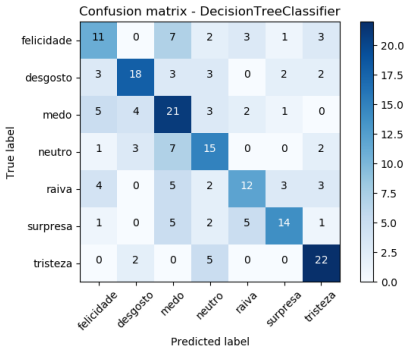
A máquina de vetor suporte praticamente pontuou o mesmo que a MLP em ambos experimentos, sendo que no experimento com *data augmentation* obteve 93% de acertos, como pode ser vista na Tabela 9, contudo podemos observar que a sua variância é consideravelmente menor, o que indica boa generalização em casos novos.

A rede convolucional obteve o melhor resultado (30%) no experimento sem *data augmentation*, mas pontuou apenas cerca de 86% no experimento com *data augmentation* como pode ser visto nas Tabelas 8 e 9, contudo sua variância foi a que mais diminuiu no segundo experimento em relação ao primeiro entre os classificadores investigados, o que indica que a rede convolucional é mais sensível ao efeito de *overfiting*

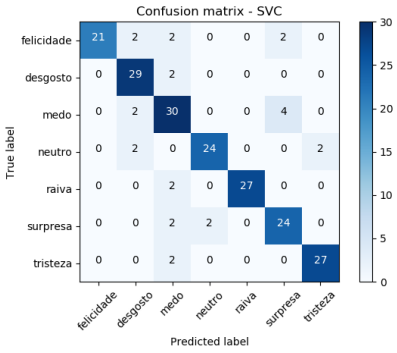
quando há menos instâncias de treino.



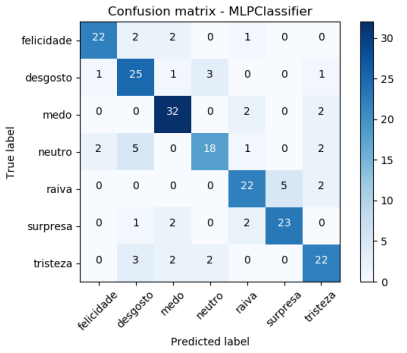
(a) Naive Bayes.



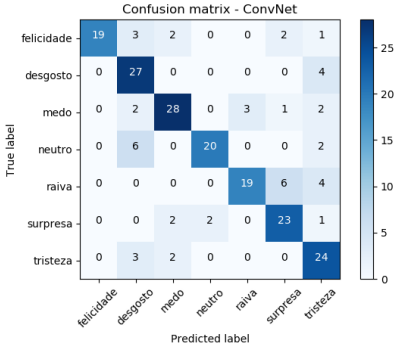
(b) Árvore de decisão.



(c) Máquina de vetor suporte.



(d) Multi layer perceptron.



(e) Rede convolucional.

Figura 36 – Matrizes de confusão para os experimentos com *data augmentation*.

Os hiperparâmetros encontrados para cada classificador nos experimentos sem *data augmentation* podem ser visualizados nas Tabelas 10 à 13 e os hiperparâmetros encontrados nos experimentos com *data augmentation* podem ser visualizados nas Tabelas 14 à 17.

Na Figura 35a- 35e podem ser visualizadas as matrizes de confusão para cada experimento sem *data augmentation*. As matrizes para os experimentos com *data augmentation* são mostradas da Figura 36a-36e. As matrizes de confusão são uma ferramenta para avaliar a performance de algoritmos de aprendizado em tarefas de classificação e tenta demonstrar de uma maneira visual como os erros (ou confusões)

Hiperparâmetro	**Valores**
critério de avaliação do atributo	entropia
quantidade mínima de amostras	2
quantidade máxima de nodos	50

Tabela 14 – Hiperparâmetros obtidos para a árvore de decisão (com *data augmentation*).

Hiperparâmetro	**Valores**
função de ativação	tanh
algoritmo de treino	gradiente descendente (*sgd*)
taxa de aprendizado	0.5
hidden_layer_sizes	100
épocas	500
taxa de inércia (*momentum*)	0.8
parada prematura	Sim

Tabela 15 – Hiperparâmetros obtidos para o multi layer perceptron (com *data augmentation*).

Hiperparâmetro	**Valores**
função kernel	rbf
gamma	0.0001
flexibilidade (*C*)	100
degree	2

Tabela 16 – Hiperparâmetros obtidos para a máquina de vetor suporte (com *data augmentation*).

Hiperparâmetro	**Valores**
épocas	500
tamanho do lote (*batch_size*)	20
número de operações convolucionais	1
quantidade de filtros	8
tamanho da janela de kernel	(3,3)
tamanho da janela de *pooling*	(1,2)
função de ativação	tanh
taxa de aprendizado	0.01
taxa de inércia (*momentum*)	0.8

Tabela 17 – Hiperparâmetros obtidos para a rede convolucional (com *data augmentation*).

ocorreram entre as amostras das diferentes classes do problema, o eixo horizontal significa as classes previstas das amostras e no eixo vertical são as classes originárias das amostras, quando um classificador acerta todas as previsões a matriz de confusão possui valores apenas na sua diagonal principal.

5.7 Teste de hipótese

Foi utilizado o teste de hipótese T-student para ser possível a comparação entre os modelos testados com a pressuposição de variâncias diferentes pois não se conhece a distribuição da população.

A fórmula para o teste T quando duas amostras de distribuições e variância desconhecidas são comparadas pode ser visto a seguir:

$$t = \frac{m_A - m_B}{\sqrt{\frac{S_A^2}{n_A} + \frac{S_B^2}{n_B}}}, \tag{5.1}$$

onde t é o resultado do teste de hipótese (também conhecido por ***p-value***), m_A e m_B são as médias a serem comparadas dos conjuntos de amostras A e B, n_A e n_B são os tamanhos dos conjuntos de amostras A e B. O termo S^2 também é conhecido como variância combinada da amostra e é obtida pela fórmula apresentada a seguir:

$$S^2 = \frac{\sum_{i=1}^{n}(x_i - m)^2}{n-1}, \tag{5.2}$$

onde x é o valor da cada amostra no conjunto, m é o valor médio do conjunto e n a quantidade de amostras.

O grau de liberdade (*degree of freedom*) é obtido a partir dos tamanhos de amostra do primeiro classificador e do segundo classificador e quando é suposta variâncias diferentes e desconhecidas tem a forma apresentada a seguir:

$$df = \frac{\left[\frac{S_A^2}{n_A} + \frac{S_B^2}{n_B}\right]^2}{\frac{\left(S_A^2/n_A\right)^2}{n_A - 1} + \frac{\left(S_B^2/n_B\right)^2}{n_B - 1}}, \tag{5.3}$$

onde S_A^2 e S_B^2 são as variâncias combinadas do conjunto A e B e os termos n_a e n_b são os tamanhos dos conjuntos de amostras A e B respectivamente.

Todos os resultados dos classificadores foram organizados em grupos de dois em dois e submetido ao teste T-student aos pares e com intervalo de confiança de 95%.

Pares do teste de hipótese	Interpretação do resultado	Vencedor
ConvNet e AD	P value obtido foi igual a 0.004158. Logo por convenção a diferença entre as médias é considerada significante.	ConvNet
ConvNet e NB	P value obtido foi igual a 0.000339. Logo por convenção a diferença entre as médias é considerada significante.	ConvNet
AD e SVC	P value obtido foi igual a 0.011593. Logo por convenção a diferença entre as médias é considerada significante.	SVC
AD e MLP	P value obtido foi igual a 0.011038. Logo por convenção a diferença entre as médias é considerada significante.	MLP
ConvNet e MLP	P value obtido foi igual a 0.535972. Logo por convenção a diferença entre as médias é considerada insignificante.	Empate
MLP e SVC	P value obtido foi igual a 0.719721. Logo por convenção a diferença entre as médias é considerada insignificante.	Empate
ConvNet e SVC	P value obtido foi igual a 0.314546. Logo por convenção a diferença entre as médias é considerada insignificante.	Empate
SVC e NB	P value obtido foi igual a 0.000581. Logo por convenção a diferença entre as médias é considerada significante.	SVC
AD e NB	P value obtido foi igual a 0.264341. Logo por convenção a diferença entre as médias é considerada insignificante.	Empate
MLP e NB	P value obtido foi igual a 0.000756. Logo por convenção a diferença entre as médias é considerada significante.	MLP

Tabela 18 – Resultados dos testes de hipótese 1 (sem *data augmentation*).

A hipótese nula é que não há diferença significativa da média de precisão do primeiro algoritmo em relação ao segundo e como hipótese alternativa é que existe uma relevante diferença entre suas médias amostrais de tal forma que se pode estabelecer uma comparação.

Pares do teste de hipótese	**Interpretação do resultado**	**Vencedor**
ConvNet e SVM	P value obtido foi igual a 0.000908. Logo por convenção, a diferença entre as médias é considerada significante.	SVM
MLP e NB	P value obtido foi menor que 0.0001. Logo por convenção, a diferença entre as médias é considerada bastante significante.	MLP
MLP e SVM	P value obtido foi igual a 0.736659. Logo por convenção, a diferença entre as médias é considerada insignificante.	Empate
MLP e ConvNet	P value obtido foi igual a 0.000752. Logo por convenção, a diferença entre as médias é considerada significante.	MLP
MLP e AD	P value obtido foi menor que 0.0001. Logo por convenção, a diferença entre as médias é considerada bastante significante.	MLP
AD e SVM	P value obtido foi menor que 0.0001. Logo por convenção, a diferença entre as médias é considerada bastante significante.	SVM
ConvNet e NB	P value obtido foi menor que 0.0001. Logo por convenção, a diferença entre as médias é considerada bastante significante.	ConvNet
SVM e NB	P value obtido foi menor que 0.0001. Logo por convenção, a diferença entre as médias é considerada bastante significante.	SVM
AD e NB	P value obtido foi menor que 0.0001. Logo por convenção, a diferença entre as médias é considerada bastante significante.	AD
ConvNet e AD	P value obtido foi menor que 0.0001. Logo por convenção, a diferença entre as médias é considerada bastante significante.	ConvNet

Tabela 19 – Resultados dos testes de hipótese 2 (com *data augmentation*).

6 Análise dos resultados

Todos os resultados de cada experimento foram agrupados aos pares e submetidos ao teste de hipótese. O melhor classificador é eleito a partir do número maior de vitórias, ou seja, dentre todos os testes realizados para cada classificador qual teve um maior número de vezes em que sua média foi superior e teve uma diferença estatisticamente significativa do seu par no teste de hipótese.

Os valores obtidos no experimento sem *data augmentation* que constam na Tabela 8 foram submetidos ao teste de hipótese, o que totalizou 10 testes. O resultado pode ser visualizado na Tabela 18. Em termos de um melhor classificador houve um empate entre a rede convolucional, a rede neural e a máquina de vetor suporte no primeiro experimento para o teste de hipótese. Cada um obteve duas vitórias.

A rede convolucional alcançou uma precisão de cerca de 30%, mesmo que este resultado não seja muito diferente dos outros este era um resultado esperado devido a facilidade que uma rede convolucional tem de aprender representações de alto nível do problema. A rede neural e a convnet apresentaram dificuldade em diferenciar as classes desgosto de tristeza, porém no caso da rede neural este efeito também afetou as classes medo e neutro.

O naive bayes também demonstrou um comportamento similar a rede neural onde as classes desgosto, medo e neutro foram classificadas como sendo da classe tristeza, isso leva a crer que o motivo está nos atributos altamente correlacionados nos casos dessas classes o que dificulta a diferenciação em relação às outras classes do problema.

A árvore de decisão apresentou um comportamento distinto ainda que ineficaz para o problema, contudo como as árvores escolhem o atributo que maximiza o ganho de informação e levando em conta que as classes desgosto, medo, neutro e tristeza obtiveram resultados baixos é possível notar que os atributos que mais relacionam-se com estas classes também devem estar altamente correlacionados entre eles mesmos, o que levou a um baixo ganho de informação e logo a não escolha destes atributos assim prejudicando a pontuação de todas estas classes.

A máquina de vetor suporte demonstrou dificuldades em diferenciar várias classes do problema, porém conseguiu diferenciar bem melhor que todos os outros a classe tristeza das classes desgosto, medo e neutro. Demonstrando que a fronteira ótima obtida em uma representação do dados utilizando a função kernel é bem indicada ao conjunto de atributos que correlaciona estas classes.

Em termos gerais os resultados de todos os classificadores foram muito baixos no experimento inicial, onde foram utilizados apenas os dados originais do problema como pode ser visto na Figura 35a- 35e. Embora a estratégia de uma validação cruzada aninhada com parada prematura contribua para tornar o conjunto de treinamento ainda menor, o maior fator que determinou a baixa performance destes modelos foi o volume de dados insuficientes disponíveis no banco o que levou a uma situação de *overfitting* devido a condição conhecida como *curse of dimensionality* (RAUDYS; JAIN, 1991).

A aplicação da técnica de *data augmentation* foi realizada a fim de mitigar este problema e um segundo experimento foi executado. De mesmo modo todos os valores obtidos (que podem ser visualizados na Tabela 9) foram agrupados aos pares e o resultado da rodada de teste de hipótese pode ser visualizado na Tabela 19.

No experimento com *data augmentation* todos os classificadores pontuaram bem melhor como pode ser visto na Figura 36a- 36e, o que confirma que o VERBO dataset por apresentar poucas amostras degradou a performance do primeiro experimento. O melhor classificador foi um empate entre a rede neural e a máquina de vetor suporte (ambos com três vitórias), com a rede convolucional em terceiro lugar (duas vitórias). É notável que a técnica de *data augmentation* favoreceu melhores resultados em todos os classificadores, embora houvesse uma expectativa sobre uma melhor performance da rede convolucional esse não foi o caso neste experimento, o que leva à conclusão de que neste problema específico levando em conta a complexidade da regra a ser aprendida, a complexidade do algoritmo classificador e o tamanho do conjunto de treino as técnicas de AM tradicionais forneceram melhores modelos por sofrerem menos com os efeitos do *curse of dimensionality* do que em comparação com as técnicas de aprendizado profundo.

A relação de volume de dados e performance de alguns algoritmos é bem conhecida e pode ser visualizada na Figura 31.

Pelos resultados observados, podemos concluir que algoritmos como MLP e SVM (que pertencem às classes de redes neurais e algoritmos tradicionais de AM) podem, de fato, obter uma melhor performance neste conjunto de dados em comparação a rede convolucional (que pertence a classe de algoritmos de aprendizado profunda ou *deep learning* por sofrerem menos com os efeitos do *curse of dimensionality*) neste conjunto de dados.

O classificador naive bayes não se mostrou muito adequado ao problema devido ao seu viés de assumir que os atributos ocorrem de maneira independente e que no caso dos dados deste experimento possuem uma correlação positiva. Suposição apoiada pelos bons resultados alcançados pela rede convolucional que possui um viés oposto (que os atributos possuem relações de vizinhança).

A SVM encontrou hiperparâmetros idênticos para ambos os experimentos (como pode ser visto nas Tabelas 12 e 16) o que pode indicar que um classificador com SVM apresenta menor variância nas fronteiras de decisão obtidas podendo generalizar melhor em instâncias novas.

Os melhores classificadores encontrados foram utilizadas para montar os gráficos em caixa presentes nas Figuras 37 à 41, demonstrando como as emoções distintas pontuaram durante a classificação e como sua precisão variou em cada partição de dados que referenciaremos apenas como a relevância do classificador em relação a uma determinada emoção no conjunto de dados.

Em comparação aos resultados obtidos no experimento de (NETO et al., 2018) pode se notar algumas similaridades. Embora houve certa facilidade em se classificar as instâncias da classe felicidade e raiva, estas foram a que apresentaram maior variância em relação a relevância da previsão o que demonstra que estas emoções são representadas de maneiras diferentes por diferentes locutores e necessitando que o modelo aprenda múltiplas representações diferentes. As emoções de desgosto, neutro, medo, surpresa e tristeza obtiveram resultados excelentes quando considerados os resultados da SVM em relação ao experimento com jurados, no qual pode se perceber uma menor variância na relevância das previsões. Contudo, o conjunto de dados tem classes desbalanceadas e pode ter favorecido justamente a modelagem das classes desgosto e medo que são as mais numerosas, conforme pode ser visualizado na Figura 33.

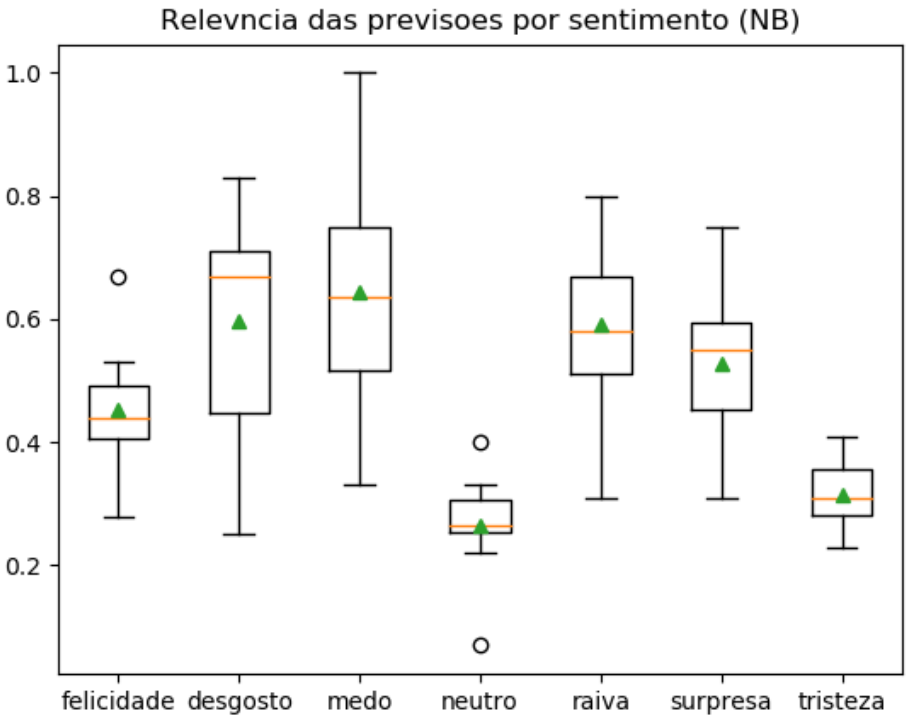


Figura 37 – Relevância da previsão por emoção (NB).

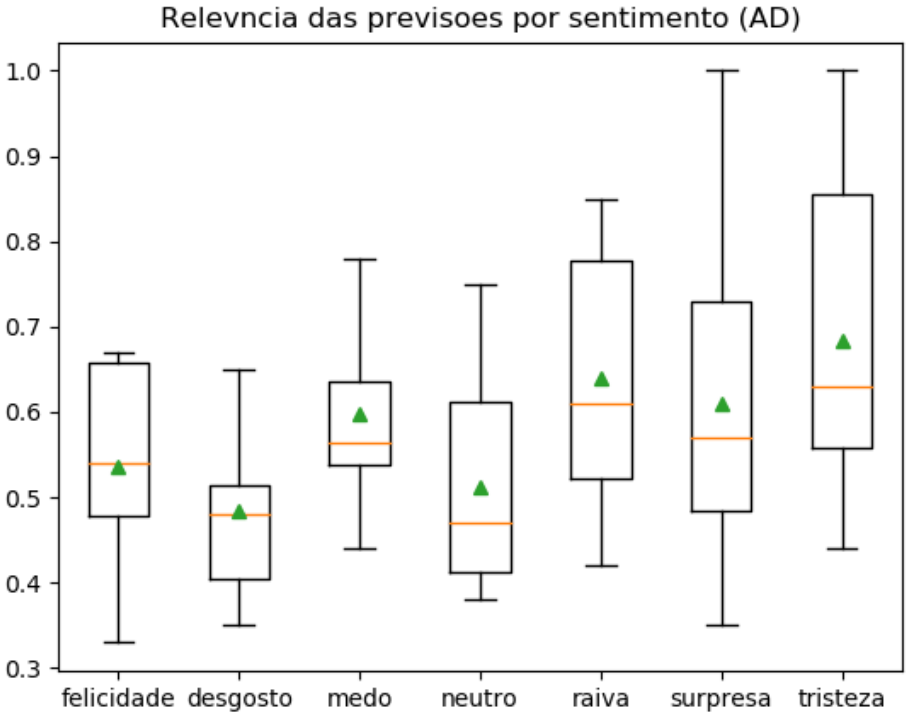


Figura 38 – Relevância da previsão por emoção (AD).

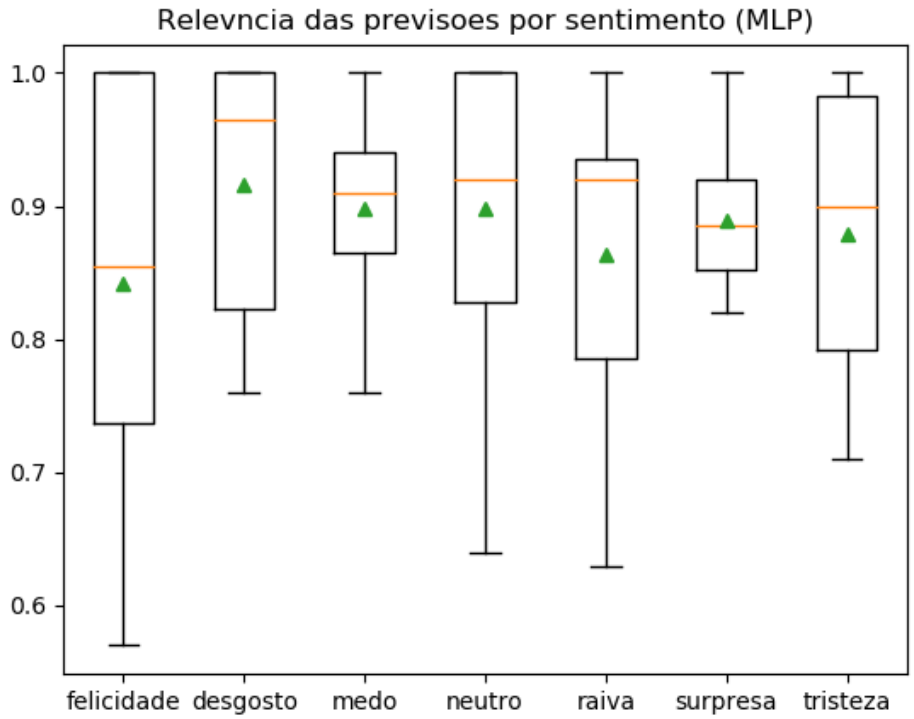


Figura 39 – Relevância da previsão por emoção (MLP).

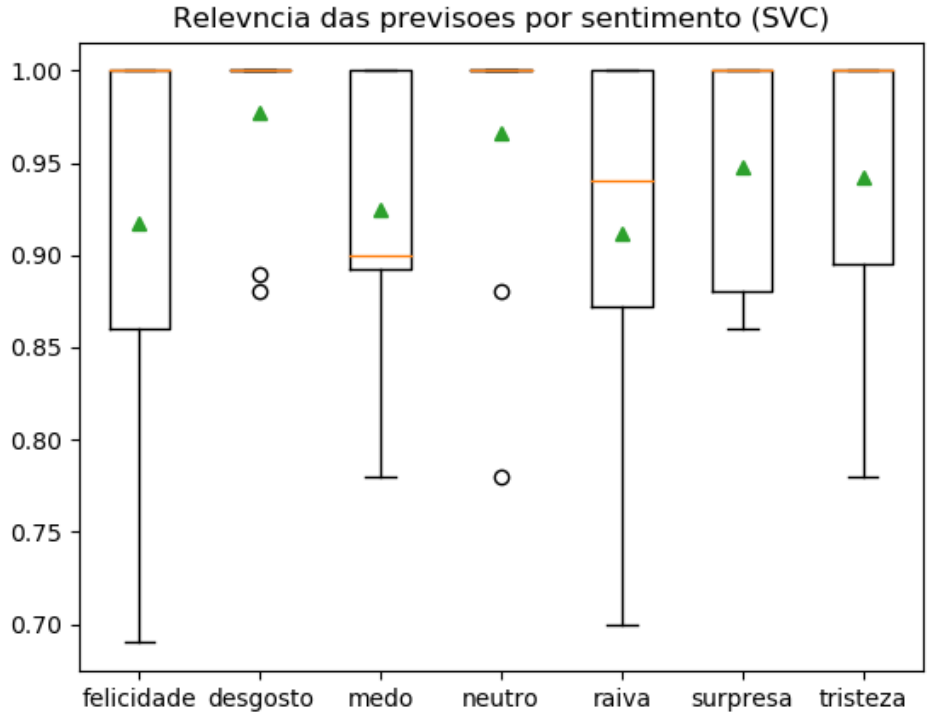


Figura 40 – Relevância da previsão por emoção (SVM).

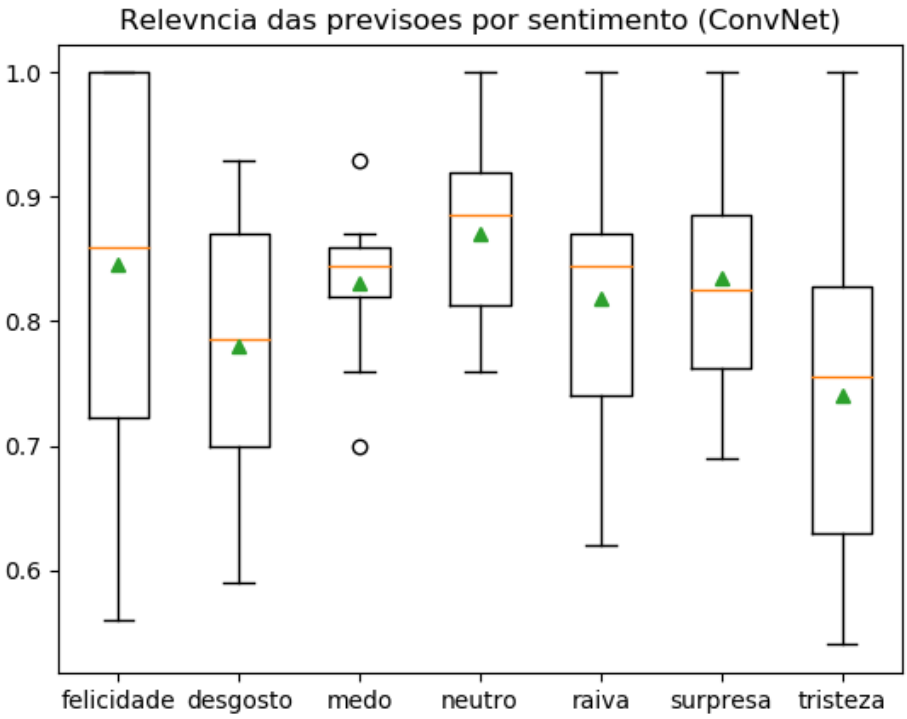


Figura 41 – Relevância da previsão por emoção (ConvNet).

7 Conclusões

Neste trabalho foi investigado a viabilidade de um sistema de reconhecimento de emoções da fala, condição que acreditamos essencial para uma boa compreensão do usuário nos seus diversos estados de humor e, por tanto, melhores resultados na aplicação do reconhecimento de fala em aplicações voltadas a interface com o usuário.

Os trabalhos revisados da literatura por vezes utilizaram conjunto de dados abundantes que facilita o desenvolvimento de pesquisas com algoritmos de AM de aprendizagem profunda, este estudo contou com um conjunto de dados bem limitado, o que impôs uma dificuldade a aplicação de algoritmos de aprendizagem profunda, contudo esta limitação pode ser mitigada com a adoção de uma técnica adequada de *data augmentation* que representa o estado da arte (PARK et al., 2019).

Do ponto de vista da seleção de atributos este trabalho adota apenas os atributos espectrais capturados pela técnica MFCC o que contrasta com os trabalhos revisados na literatura que por vezes empregam diversas técnicas de extração de atributos ou a mistura com atributos prosódicos para o modelo, esta escolha embora motivada por motivos de orçamento se mostrou coerente para a realidade do conjunto de dados. Uma vez que se adiciona mais dimensões na forma de atributos para diferenciar o problema, o volume de dados tem que crescer exponencialmente apenas para manter a mesma densidade de distribuição nessa nova representação e quando não há dados suficientes o classificador acaba modelando os detalhes menos importantes dos poucos pontos que existem nessas novas dimensões, ou seja, generalizando menos por *overfiting*.

Estes resultados ajudam a validar que o conjunto de atributos relacionados ao domínio das frequências e o viés introduzido com a adoção dos atributos espectrais durante o pré-processamento foram adequados para um modelo que generaliza bem. Obtendo um resultado similar ao que o comitê de jurados obteve no experimento de (NETO et al., 2018).

O modelo proposto para a classificação das emoções ”alegria”, ”surpresa”, ”desgosto”, ”neutro”, ”medo”, ”raiva” e ”tristeza” emprega *data augmentation* e extração de atributos espectrais e pode ser implementado com redes neurais ou por máquinas de vetor suporte, sendo que a menor variância pôde ser encontrada com a SVM.

A rede convolucional demonstrou vantagens quanto ao treino mais rápido da rede porém seu resultado ótimo não pôde ser alcançado devido ao volume de dados disponível que levou a um modelo sobre ajustado e com dificuldade em generalizar.

Observou-se que o modelo com redes neurais apresentou o melhor resultado mesmo que estatisticamente não muito diferente da máquina de vetor suporte. Estes dois algoritmos possuem diversas características que corroboram para o bom resultado de ambos, como a capacidade das redes neurais de ser robusta a ruídos e atributos irrelevantes e no caso da SVM, como a mesma encontra a fronteira de decisão ótima entre as classes mesmo em dados de alta dimensionalidade.

Um trabalho futuro pode ser a investigação em dados mais volumosos, possivelmente com a inclusão de outros métodos de *data augmentation*. Este experimento demonstrou como um vasto conjunto de dados é imprescindível para que algoritmos de aprendizado profunda como as redes convolucionais possam demonstrar uma boa performance.

Se dados suficientes estiverem disponíveis, a inclusão de outros algoritmos de aprendizado profunda como as redes recorrentes (*long short-term memory* ou LSTM) e redes convolucionais profundas (*Deep ConvNets*) podem obter resultados ainda melhores e constarão em uma avaliação futura.

Referências

ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <http://tensorflow.org/>. Citado na página 60.

AYADI, M. E.; KAMEL, M. S.; KARRAY, F. Survey on speech emotion recognition: Features, classification schemes, and databases. *Pattern Recognition*, Elsevier, v. 44, n. 3, p. 572–587, 2011. Citado na página 12.

BISHOP, C. M. *Bishop pattern recognition and machine learning*. [S.l.]: Springer, New York, 2001. Citado na página 40.

BRINK, H.; RICHARDS, J. W.; FETHEROLF, M. *Real-world machine learning*. [S.l.]: Manning, 2017. Citado 2 vezes nas páginas 30 e 31.

BURKHARDT, F. et al. A database of german emotional speech. In: *Ninth European Conference on Speech Communication and Technology*. [S.l.: s.n.], 2005. Citado na página 59.

CHOLLET, F. et al. *Keras*. 2015. <https://keras.io>. Citado na página 60.

COOK, N. D. *Tone of voice and mind: The connections between intonation, emotion, cognition and consciousness*. [S.l.]: John Benjamins Publishing, 2002. v. 47. Citado na página 13.

CORDIOLI, A. V.; ZIMMERMANN, H. H.; KESSLER, F. Rotina de avaliação do estado mental. *Consultado em*, v. 4, 2012. Citado na página 12.

COWAN, M. L. et al. It's the way he tells them (and who is listening): men's dominance is positively correlated with their preference for jokes told by dominant-sounding men. *Evolution and Human Behavior*, Elsevier, v. 37, n. 2, p. 97–104, 2016. Citado na página 12.

EKMAN, P. Are there basic emotions? American Psychological Association, 1992. Citado na página 16.

GRAVES, A.; JAITLY, N. Towards end-to-end speech recognition with recurrent neural networks. In: *International conference on machine learning*. [S.l.: s.n.], 2014. p. 1764–1772. Citado na página 12.

HAYKIN, S. *Neural networks: a comprehensive foundation*. [S.l.]: Prentice Hall PTR, 1994. Citado 4 vezes nas páginas 36, 44, 45 e 47.

HUBEL, D. H.; WIESEL, T. N. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, Wiley Online Library, v. 148, n. 3, p. 574–591, 1959. Citado na página 47.

LI, X.; ZHOU, Z. Speech command recognition with convolutional neural network. 2017. Citado na página 15.

MERMELSTEIN, P. Distance measures for speech recognition, psychological and instrumental. *Pattern recognition and artificial intelligence*, v. 116, p. 374–388, 1976. Citado 2 vezes nas páginas 23 e 26.

MILAN, P.; KLUGE, D. C. O papel da frequência fundamental e da intensidade para distinguir enunciados interrogativos de afirmativos na região nuclear do dialeto curitibano. *Signum: Estudos da Linguagem*, v. 20, n. 3, p. 265–292, 2017. Citado 2 vezes nas páginas 12 e 13.

MITRA, V.; FRANCO, H. Time-frequency convolutional networks for robust speech recognition. In: IEEE. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. [S.l.], 2015. p. 317–323. Citado 2 vezes nas páginas 23 e 58.

NAKAGAWA, S. A survey on automatic speech recognition. *IEICE TRANSACTIONS on Information and Systems*, The Institute of Electronics, Information and Communication Engineers, v. 85, n. 3, p. 465–486, 2002. Citado na página 12.

NASICHUDDIN, M. A.; ADJI, T. B.; WIDYAWAN, W. Performance improvement using cnn for sentiment analysis. *IJITEE (International Journal of Information Technology and Electrical Engineering)*, v. 2, n. 1, p. 9–14, 2018. Citado na página 15.

NETO, J. R. T. et al. Verbo: voice emotion recognition database in portuguese language. *Journal of Computer Science*, Science Publications, 2018. Citado 4 vezes nas páginas 17, 61, 77 e 81.

PARK, D. S. et al. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019. Citado 3 vezes nas páginas 57, 64 e 81.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 60.

RAUDYS, S. J.; JAIN, A. K. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, IEEE, n. 3, p. 252–264, 1991. Citado 3 vezes nas páginas 55, 56 e 76.

RUDNICKY, A. I.; HAUPTMANN, A. G.; LEE, K.-F. Survey of current speech technology. *Communications of the ACM*, Citeseer, v. 37, n. 3, p. 52–57, 1994. Citado na página 12.

RUSSELL, J. A. A circumplex model of affect. *Journal of personality and social psychology*, American Psychological Association, v. 39, n. 6, p. 1161, 1980. Citado 2 vezes nas páginas 14 e 16.

SCHERER, K. R. What are emotions? and how can they be measured? *Social science information*, Sage Publications Sage CA: Thousand Oaks, CA, v. 44, n. 4, p. 695–729, 2005. Citado na página 16.

SHRAWANKAR, U.; THAKARE, V. M. Techniques for feature extraction in speech recognition system: A comparative study. *arXiv preprint arXiv:1305.1145*, 2013. Citado na página 26.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado 2 vezes nas páginas 15 e 58.

TAKAHASHI, N. et al. Deep convolutional neural networks and data augmentation for acoustic event detection. *arXiv preprint arXiv:1604.07160*, 2016. Citado 2 vezes nas páginas 58 e 64.

VLADIMIR, N. V. *The Nature of Statistical Learning Theory (Information Science and Statistics)*. [S.l.]: Springer, November, 1999. Citado 2 vezes nas páginas 43 e 46.

VOLKMANN, J.; STEVENS, S.; NEWMAN, E. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, ASA, v. 8, n. 3, p. 208–208, 1937. Citado na página 26.

WU, S.; FALK, T. H.; CHAN, W.-Y. Automatic speech emotion recognition using modulation spectral features. *Speech communication*, Elsevier, v. 53, n. 5, p. 768–785, 2011. Citado 4 vezes nas páginas 12, 18, 23 e 59.

ZENG, Z. et al. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 31, n. 1, p. 39–58, 2008. Citado na página 12.

ZHANG, Y. et al. Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*, 2017. Citado 2 vezes nas páginas 15 e 58.