Ernani Rodrigues de São Thiago

# Reconhecimento de Voz utilizando extração de Coeficientes Mel-Cepstrais e Redes Neurais Artificiais

São José - SC

Dezembro/2017

Ernani Rodrigues de São Thiago

# Reconhecimento de Voz utilizando extração de Coeficientes Mel-Cepstrais e Redes Neurais Artificiais

Monografia apresentada à Coordenação de Engenharia de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheiro de Telecomunicações.

Instituto Federal de Santa Catarina – IFSC

Campus São José

Engenharia de Telecomunicações

Orientador: Ramon Mayor Martins

São José - SC

Dezembro/2017

Ernani Rodrigues de São Thiago
Reconhecimento de Voz utilizando extração de Coeficientes Mel-Cepstrais e Redes Neurais Artificiais/ Ernani Rodrigues de São Thiago. – São José - SC, Dezembro/2017-
Orientador: Ramon Mayor Martins

Monografia (Graduação) – Instituto Federal de Santa Catarina – IFSC
Campus São José
Engenharia de Telecomunicações, Dezembro/2017.

1. Mel Cepstral. 2. Redes Neurais. 2. Reconhecimento de Voz. I. Ramon Mayor Martins. II. Instituto Federal de Santa Catarina. III. Campus São José. IV. Reconhecimento de Voz utilizando extração de Coeficientes Mel-Cepstrais e Redes Neurais Artificiais.

Ernani Rodrigues de São Thiago

# Reconhecimento de Voz utilizando extração de Coeficientes Mel-Cepstrais e Redes Neurais Artificiais

Monografia apresentada à Coordenação de Engenharia de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheiro de Telecomunicações.

Trabalho aprovado. São José - SC, Dezembro de 2017:

---

**Ramon Mayor Martins**
**Me. Eng.**
Orientador

---

**Prof. Elen Macedo Lobato**
**Dr. Eng.**
Membro da Banca - IFSC

---

**Prof. Diego da Silva de Medeiros**
**Me. Eng.**
Membro da Banca - IFSC

---

**Prof. Cleber Jorge Amaral**
**Me. Tecno.**
Membro da Banca - IFSC

São José - SC
Dezembro/2017

*Este trabalho é dedicado à minha família, que sempre me apoiou incondicionalmente durante toda a vida acadêmica.*

# Agradecimentos

A Deus por ter me dado a força e oportunidade para concluir este trabalho. A minha família por sempre estar presente, incentivando e indicando o caminho.

Ao meu orientador, Me. Ramon Mayor Martins pelo empenho dedicado na elaboração do escopo deste trabalho. Ao Instituto Federal de Santa Catarina, corpo docente e direção por criarem um ambiente que tornou isto possível.

Aos professores Me. Diego da Silva de Medeiros, Dra. Elen Macedo Lobato, Dr. Carlos Alberto Ynoguti do INATEL (Instituto Nacional de Telecomunicações) e todos aqueles que de alguma forma contribuíram para o desenvolvimento deste projeto.

*"Os que se encantam com a prática sem a ciência são como os timoneiros que entram no navio sem timão nem bússola, nunca tendo certeza do seu destino."*
*(Leonardo da Vinci)*

# Resumo

Com os avanços obtidos nos campos de Inteligência Artificial, os computadores passaram de meros executores de sequências lógicas, para desenvolvedores de soluções, capazes de aprender a partir de experiências (exemplos) coletados do ambiente. Neste contexto, Redes Neurais Artificiais (RNA) são largamente utilizadas para computar classificadores, capazes de classificar (com uma tolerável taxa de erro) diferentes tipos de dados. Neste trabalho, utilizou-se Coeficientes Mel-Cepstrais (MFCC) para extrair vetores de características de dados de voz, com o intuito de treinar diferentes RNAs com os dados processados. Para o treinamento da RNA, desenvolveu-se um *software* capaz de variar cinco parâmetros selecionados pelo usuário: número de neurônios, número de camadas ocultas, algoritmo de treinamento, taxa de aprendizado e função de ativação. Um conjunto de valores específicos foram selecionados, baseados em técnicas empíricas e teorias fundamentadas, no intuito de realizar uma análise a respeito do desempenho de RNAs. A experimentação de diferentes topologias de RNA, alcançou resultados bastante satisfatórios, onde a melhor taxa de acerto encontrada foi de 97,58%. Dessa forma, é possível considerar a utilização de RNAs em sinergia com técnicas empíricas e heurísticas para seleção de parâmetros, como alternativa à métodos mais bem estabelecidos no meio científico, como Modelos Ocultos de Markov (HMM).

**Palavras-chave**: Redes Neurais Artificiais. Inteligência Artificial. Processamento de Sinais. Coeficientes Mel-Cepstrais.

# Abstract

With advances in the fields of Artificial Intelligence, computers have evolved from mere logical sequence implementers to solution developers who are able to learn from experiences (examples) collected from the environment. In this context, Artificial Neural Networks (ANNs) are widely used to compute classifiers, capable of classifying (with a tolerable error rate) different types of data. In this paper, Mel-Frequency Cepstral Coefficients (MFCC) were used to extract characteristics vectors of voice data, in order to train different ANNs with the processed data. For ANN training, a software was developed to vary five parameters selected by the user: number of neurons, number of hidden layers, training algorithm, learning rate and activation function. A set of specific values were selected, based on empirical techniques and grounded theories, in order to perform a comparative analysis. The experimentation of different ANN topologies, reached very satisfactory results, where the best hit rate was 97.58 %. Thus, it is possible to consider the use of ANNs in synergy with empirical and heuristic techniques for parameter selection, as an alternative to more well established methods in the scientific environment, such as Hidden Markov Models (HMM).

**Keywords**: Artificial Neural Networks. Artificial Intelligence. Signal Processing. Mel-Frequency Cepstral Coefficients.

# Lista de tabelas

# Lista de ilustrações

# Lista de abreviaturas e siglas

| | |
|---|---|
| RNA | *Artificial Neural Network*, Rede Neural Artificial |
| MFCC | *Mel-Frequency Cepstral Coefficient*, Coeficiente de Frequência Mel-Cepstral |
| VT | *Vocal Tract*, Trato Vocal |
| HMM | *Hidden Markov Model*, Modelo Oculto de Markov |
| IHM | Interface Homem-Máquina |
| RAV | Reconhecimento Automático de Voz |
| WER | *Word Error Rate*, Taxa de Erro de Palavra |
| SLP | *Single Layer Perceptron*, *Perceptron* de Camada Única |
| MLP | *Multi Layer Perceptron*, *Perceptron* de Múltiplas Camadas |
| A/D | Analógico-Digital |
| LPC | *Linear Predictive Coding*, Codificação Preditiva Linear |
| FFT | *Fast Fourier Transform*, Transformada Rápida de Fourier |

# Sumário

# 1 Introdução

Com o desenvolvimento crescente de novas tecnologias, torna-se cada vez mais necessário melhorar os métodos utilizados para realizar a interação entre usuário e máquina, ou IHMs (interfaces homem-máquina). *Mouses*, botões e monitores, são apenas algumas das diversas IHMs desenvolvidas ao longo do tempo com o objetivo de tornar mais natural e simples a maneira como o ser humano interage com a tecnologia.

Apesar dos exemplos de IHMs citadas serem muito eficientes, não existe nada mais natural para o ser humano do que a fala. A voz é o principal meio utilizado para compartilhar informações, logo, justifica-se o surgimento de técnicas capazes de adaptar as novas tecnologias para interpretarem comandos baseados em voz. Siri, Cortana, Assistente Google e Amazon Echo, são apenas alguns dos diversos assistentes virtuais desenvolvidos com reconhecimento de voz para melhor atender seus usuários.

Entretanto, como descrito em (CHIN-HUI; SOONG; PALIWAL, 2012, p. 2), o sinal de voz produzido pelo sistema vocal humano é extremamente complexo, o que dificulta a tarefa de caracterizá-lo. Existem inúmeros modelos matemáticos sofisticados que tentam simular a produção de voz humana, no entanto, sua capacidade de modelamento ainda é muito limitada.

A voz é um sinal que carrega muita informação, devido a este fato, não é possível sintetizar e analisar diretamente a mesma. Dessa forma, para o desenvolvimento de sistemas modernos de Reconhecimento Automático de Voz (RAVs), utiliza-se a combinação de algoritmos de extração e reconhecimento de características. Sistemas RAV são responsáveis por produzir uma representação textual do sinal de voz. Estes sistemas possuem muitas aplicações em potencial, incluindo ditado, transcrição de discursos gravado, pesquisa em documentos de áudio, comandos de voz e diálogos interativos (GALES, 2008, p. 197).

Algoritmos de extração de características removem quaisquer informação que não seja útil para o reconhecimento do sinal, e ressaltam os aspectos que contribuem para a identificação de diferenças. Algumas técnicas de modelamento acústico utilizadas para a extração de vetores de características são: MFCC (*Mel-Frequency Cepstral Coefficients*) (PRAVEEN et al., 2016), LPC (*Linear Predictive Coding*) (GUPTA; GUPTA, 2016) e PLP (*Perceptual Linear Prediction Coefficients*) (FAN et al., 2013). Já para o reconhecimento, etapa que ocorre a partir da classificação do sinal analisado, é possível utilizar técnicas como HMM (*Hidden Markov Models*) (YOUNG, 2008), SVM (*Support Vector Machines*) (SOLERA-URENA et al., 2007), modelamento linguístico (KURIMO et al., 2006), RNA (ou ANN, *Artificial Neural Networks*) (CHEN; CHEN, 1995) e, mais recentemente, DNN (*Deep Neural Networks*) (HINTON et al., 2012).

Neste trabalho, processou-se uma base de dados com MFCC, composta por palavras isoladas (dígitos de zero a nove, pronunciados em inglês) e, desenvolveu-se um *software* capaz de variar os parâmetros de uma RNA, no intuito de procurar os valores mais adequados para a resolução do problema de identificação. O principal objetivo é extrair um classificador da RNA capaz de reconhecer os dígitos isolados pronunciados, com uma taxa de erro similar (ou inferior) ao alcançado por outros trabalhos, como em (ELENIUS; BLOMBERG, 2011) (3,47%) e em (MARTINS, 2014) (1,88%).

O conteúdo deste trabalho, está dividido da seguinte forma: no Capítulo 2 são introduzidos fundamentos acerca de processamento de voz, sistemas RAV e RNAs. No Capítulo 3 é apresentada a base de dados, bem como os métodos utilizados para a extração MFCC e implementação da RNA. No Capítulo 4 são abordados os resultados obtidos através dos experimentos com diferentes configurações de RNAs. Por fim, no Capítulo 5 conclui-se sobre os resultados obtidos e no Capítulo 6 é apresentado um conjunto de sugestões e possíveis trabalhos futuros.

# 2 Fundamentação Teórica

## 2.1 Características da Fala

A pronúncia de uma palavra ou frase raramente é reproduzida duas vezes da mesma maneira. De acordo com (HECKER, 1971, p. 2), a anatomia e fisiologia do locutor influenciam nas características espectrais e temporais da voz, dessa forma, bons ouvintes podem, com frequência, determinar o sexo, a idade aproximada e até o estado emocional do locutor. Este efeito é identificável via análises acústicas e pode ser classificado como variação interlocutor ou intralocutor. Segundo (QUEIROZ; SANTANA, 2008, p. 17), a variação intralocutor se dá devido a diferenças nos estados físico e emocional do locutor, em contrapartida, a variação interlocutor, além de ser influenciada pelas diferenças de estado, inclui dissimilaridades entre os mecanismos vocais dos locutores e a maneira como estes aprenderam a utilizá-lo.

Um exemplo de variação ocasionada pelas diferenças fisiológicas entre locutores, é o trato vocal (*vocal tract*, VT), mecanismo físico responsável pela produção da voz. O trato vocal consiste na região que compreende a faringe e a cavidade oral, que estão representados na figura 1 como (1) e (2) respectivamente. E (3), (4), (5) e (6), correspondem a cavidade nasal, massa da língua, palato macio e as pregas vocais.

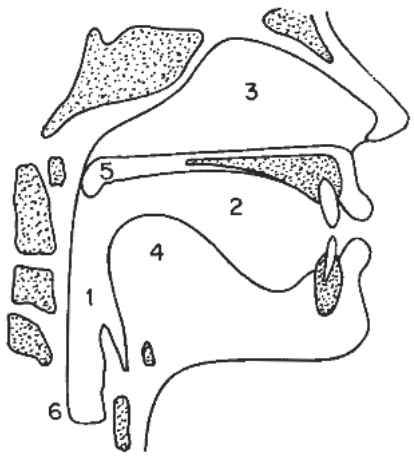


Figura 1 – Mecanismo vocal (HECKER, 1971)

Locutores adultos têm o trato vocal mais longo, enquanto locutores crianças o tem mais curto, deste modo, tratos vocais maiores tendem a produzir vozes mais graves, enquanto que os menores tendem a produzir vozes mais agudas (MARTINS, 2014, p. 3).

De acordo com o trabalho de (HECKER, 1971, p. 7), o processo de produção da fala, inicia com a entrada de ar através do sistema respiratório. A medida com que o ar é

expelido dos pulmões para a traquéia, passa pelas cordas vocais da laringe, fazendos-as vibrar. O fluxo de ar é cortado em pulsos quase periódicos que são então modulados em freqüência ao passar pela faringe (cavidade da garganta), cavidade bucal e, possivelmente, pela cavidade nasal. O som produzido é resultado da posição de vários articuladores como mandíbula, língua, lábios, boca e etc.

A classificação da variação da fala, e o entendimento do processo de geração da voz possuem grande valor, principalmente quando se busca desenvolver um sistema RAV, onde são utilizadas diversas técnicas que buscam mimicar o comportamento do organismo humano. Um exemplo disso, é a escala mel, que é utilizada para imitar a maneira como o aparelho auditivo humano interpreta o espectro de frequências. Na seção 2.2, será introduzido as etapas de um sistema RAV e, em seguida, as técnicas que envolvem o processamento da fala.

## 2.2 Sistemas de Reconhecimento de Voz

O reconhecimento automático de voz por máquinas tem sido a ambição de muitos pesquisadores por décadas e tem inspirado obras de ficção científica como o computador HAL de Stanley Kubrik no filme "Uma Odisséia no Espaço"(2001) e o robô R2D2 no clássico de George Lucas "Guerra nas Estrelas"(1977). Entretanto, apesar da grande quantidade de pesquisas para criar máquinas que pudessem compreender a fala humana, a diversidade de ambientes e locutores, ainda representa uma barreira no avanço ao desenvolvimento de um sistema genérico e funcional.

Em 1930, nos laboratórios Bell, Homer Dudley propôs um dos primeiros modelos para análise e síntese de voz. E somente em 1952, também nos laboratórios Bell, que Davis e Balashek desenvolveram um sistema para reconhecimento de dígitos isolados (números de zero a nove) com apenas um locutor (JUANG, 2004, p. 1).

Mas foi na década de 60 e 70 que as pesquisas na área realmente trouxeram resultados. Nos laboratórios da RCA (*Radio Corporation of America*), uma equipe liderada por Martin (1964), desenvolveu um conjunto de técnicas de normalização para identificar o início e o fim da fala. E Fumitada Itakura (1975), também nos laboratórios Bell, publicou pesquisas que apresentariam grandes avanços na área de identificação de características, como o LPC (*Linear Predictive Coding*).

Desde então, o problema de reconhecimento de voz passou a ser abordado progressivamente, de uma máquina que responde a um pequeno conjunto de sons a um sistema sofisticado capaz de responder a linguagem natural falada fluentemente.

Um sistema RAV é um sistema capaz de transformar um sinal de voz em uma sequência de dados com a qual uma máquina tomará decisões. A figura 2 ilustra a sequência das etapas deste tipo de sistema e como estas se relacionam com os atores externos (usuário e máquina).

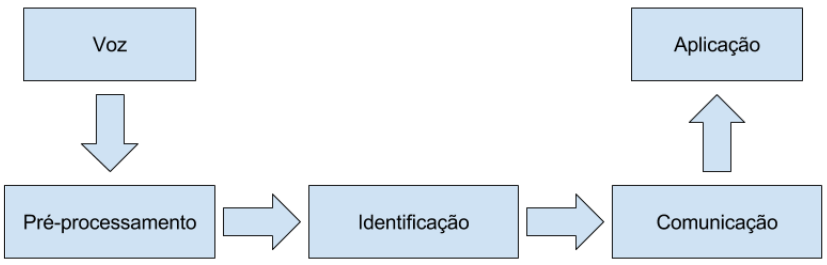


Figura 2 – Etapas de um Sistema de reconhecimento automático de voz

De acordo com (CIPRIANO, 2001, p. 16), este tipo de sistema pode ser dividido em três etapas: Pré-processamento, Identificação e Comunicação.

1. **Pré-processamento:** conversão A/D, filtragem e extração dos parâmetros acústicos.
2. **Identificação:** reconhecimento da informação baseado em representações existentes dos padrões observados.
3. **Comunicação:** envio dos resultados para o ator externo que fará o uso do mesmo.

Atualmente, a fronteira do "estado da arte"de sistemas de reconhecimento automático de voz está na etapa de Identificação, onde utilizam-se diferentes técnicas, como por exemplo: modelos estatísticos, RNAs e aprendizado profundo.

## 2.3 Pré-Processamento do sinal de Voz

O pré-processamento é o primeiro passo de um sistema de reconhecimento de voz. Nesta etapa, é realizada a conversão do sinal (de analógico para digital), a pré-ênfase, a análise espectral e a extração das características acústicas, conforme ilustrado na figura 3.

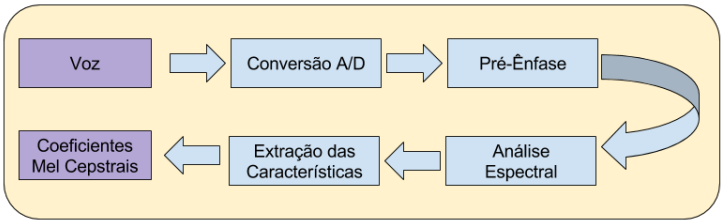


Figura 3 – Etapas do Pré-Processamento

O objetivo da etapa de pré-processamento é eliminar ruídos, descontinuidades e quaisquer efeitos que possam prejudicar o desempenho do sistema, gerando assim, um conjunto de dados contendo somente informações relevantes ao processo de Identificação (CIPRIANO, 2001, p. 19).

### 2.3.1 Conversão do Sinal e Pré-Ênfase

O sinal de voz é, em sua forma original, analógico. A diferença entre sinais analógicos e digitais é a maneira de representar a informação. Sinais analógicos traduzem a informação em pulsos elétricos de amplitude variante, e portanto, podem assumir quaisquer valor. Em contrapartida, sinais digitais traduzem a informação de forma discreta, limitando as possíveis amplitudes em um conjunto de valores finito. O processo de conversão A/D (analógico para digital) consiste na amostragem do sinal analógico, seguido da quantização e codificação das amostras resultantes.

Conforme descrito em (LATHI, 2007, p. 689), o processo de amostragem baseia-se na multiplicação do sinal analógico por um trem de impulsos de frequência $f_s$ (frequência de amostragem). De acordo com o teorema de Nyquist, a frequência de amostragem mínima para representar um sinal, é de $2f_{max}$, onde $f_{max}$ representa a frequência máxima do sinal analógico. Entretanto, uma das consequências do processo de amostragem, é o efeito de *aliasing*. Como todos os sinais práticos são de largura infinita, ocorre a sobreposição entre ciclos espectrais repetidos (*aliasing*). Amostrar em taxas mais altas reduz, mas não elimina o efeito de *aliasing*, uma solução, seria aplicar um filtro passa-baixas (filtro *anti-aliasing*) com frequência de corte de $f_s/2$, para eliminar frequências superiores ao dobro da frequência de amostragem.

A amostragem de um sinal analógico não irá resultar em um sinal digital, pois a amostra de um sinal analógico pode ainda, assumir qualquer valor em uma faixa contínua. O sinal é digitalizado pelo arredondamento de seu valor original para o valor mais próximo

de um dos $L$ números possíveis permitidos. Do ponto de vista prático, trabalhar com uma quantidade muito grande de pulsos distintos é muito difícil. Prefere-se utilizar a menor quantidade possível, devido a simplicidade, economia e facilidade, sendo o menor número possível igual dois. A quantização divide o sinal em níveis de decisão, onde os valores de cada amostra são arredondados para o nível mais próximo. Este processo adiciona erro ao sistema (devido ao arredondamento) que pode ser diminuído, mas não completamente removido, com o aumento da quantidade de níveis de quantização.

O sinal quantizado, apesar de muito mais simples do que o sinal analógico, ainda não é uma informação que possa ser transmitida por um meio de comunicação digital. Para isso é preciso utilizar métodos, como o PCM (*Pulse-Code Modulation*), para converter os níveis de quantização em sequências binárias. Basicamente, atribui-se a cada nível de quantização, uma sequência única de *bits*, sendo o resultado da conversão A/D, o equivalente binário dos níveis de quantização.

De acordo com (CIPRIANO, 2001, p. 19), após a conversão A/D, utiliza-se um filtro passa-altas, denominado pré-ênfase, para melhorar a relação sinal-ruído (SNR) e reduzir os efeitos de ruídos introduzidos em partes subsequentes do sistema. A aplicação deste filtro aumenta a magnitude de frequências (geralmente altas) em comparação com outras frequências (geralmente baixas) e sua função transferência pode ser descrita como:

$$H_{pre}(z) = 1 - a_{pre}z^{-1} \tag{2.1}$$

onde o coeficiente de pré-ênfase $a_{pre}$ varia de 0,9 a 1,0. A figura 4 ilustra a sequência do processo de conversão A/D e pré-ênfase.

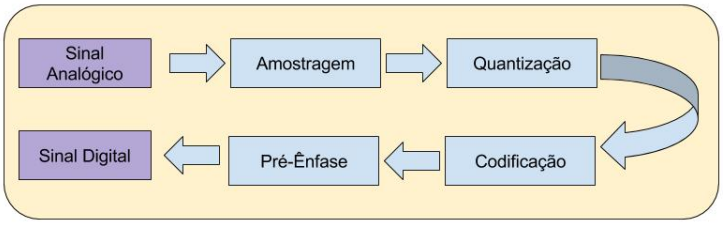


Figura 4 – Conversão A/D e Pré-Ênfase

Uma vez que o sinal for passado pelo filtro de pré-ênfase, o mesmo está pronto para a próxima etapa: a Análise Espectral.

### 2.3.2 Análise Espectral

A análise espectral consiste em converter o sinal, que está no domínio do tempo, para o domínio da frequência. Esta etapa, é muito importante, porque, de acordo com (DELLER JOHN PROAKIS, 2000, p. 936), a audição e percepção humana possuem uma correlação maior com a representação espectral do sinal do que a temporal.

A análise espectral inicia com a divisão do sinal em quadros de duração fixa, num processo denominado janelamento. O janelamento só é possível, porque o sinal de voz, apesar de ser um processo estocástico não-estacionário, pode ser considerado estacionário em pequenos intervalos de tempo, devido as lentas mudanças ocorridas no trato vocal durante a produção da voz (JUANG, 2004, p. 507). O processo de janelamento é realizado com uma superposição entre as janelas para aumentar a correlação entre as mais próximas e evitar grandes variações entre os parâmetros extraídos.

Uma vez que o sinal de voz esteja dividido em janelas, é possível aplicar, efetivamente, a análise espectral. Em sistemas RAV, dois métodos são muito utilizados para análise espectral, os bancos de filtros FFT (*Fast Fourier Transform*) e a codificação preditiva linear, ou LPC (*Linear Predictive Coding*). Com o sinal convertido para o domínio da frequência, é possível iniciar o processo de extração dos parâmetros acústicos.

### 2.3.3 Extração dos Parâmetros

A seleção de um método para a extração dos parâmetros acústicos do sinal de voz é uma das etapas mais importantes do Pré-Processamento dos dados. De acordo com (DAVIS, 1980), o objetivo principal desta etapa é comprimir os dados de voz, eliminando qualquer informação que não seja pertinente à análise dos dados e salientar aqueles aspectos do sinal que contribuem significativamente à detecção das diferenças fonéticas.

Um dos métodos mais populares para a extração de características, em reconhecimento de voz, é a obtenção dos Coeficientes Mel-Cepstrais (MFCC) (SUKSRI, 2012, p. 135). Este método foi mencionado por Bridle e Brown em 1974 e mais detalhadamente desenvolvido por Davis e Mermelstein em 1980. Para extrair um vetor de características com toda a informação linguística de um sinal de voz, a MFCC utiliza a escala Mel para analisar as diferentes frequências presentes no espectro.

Conforme descrito em (ATTNEAVE, 1971, p. 1 e 2), a escala Mel foi desenvolvida experimentalmente na década de 1940 por Stevens e Volkmann para identificar como diferentes frequências eram interpretadas pelo aparelho auditivo humano, sendo o principal objetivo do experimento, descrever uma relação entre a frequência real e o que era interpretado. A partir dos resultados, foi possível concluir que esta relação é linear de 0 a 1000 Hz, e que para frequências superiores a 1000 Hz, a relação pode ser descrita de forma logarítmica. Uma vez definido que 1000 Hz equivale a 1000 mels, é possível representar esta relação matematicamente como:

$$F_{mel} = \frac{1000}{log(2)}[1 + \frac{F_{Hz}}{1000}] \tag{2.2}$$

onde $F_{mel}$ é a frequência resultante na escala Mel medida em mels e $F_{Hz}$ é a frequência medida em Hertz. A figura 5 representa a equação 2.2 em uma plano cartesiano.

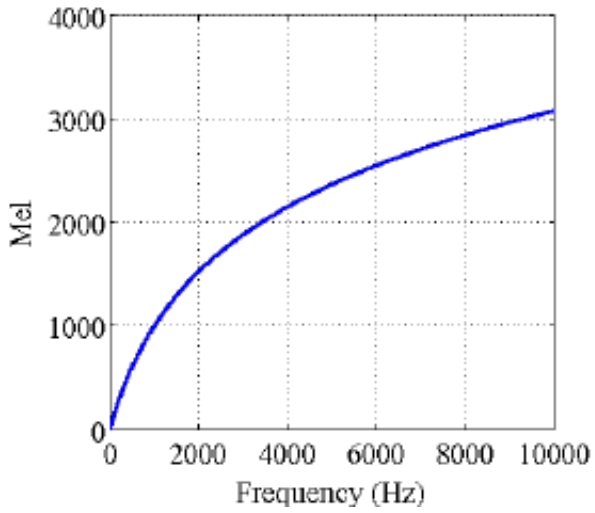


Figura 5 – Relação entre as escalas de Frequência e Mel (COUVREUR et al., 2008)

Na prática, a conversão do sinal para a escala Mel é implementada utilizando filtros passa-banda, de resposta triangular com espaçamento e largura determinados por um intervalo constante de frequência Mel (conforme a figura 6). Na tabela 1, é demonstrada um conjunto de frequências centrais de filtros que podem implementar a escala Mel.


Figura 6 – Banco de filtros Mel (MARTINS; YNOGUTI, 2014)

Uma vez que o sinal de voz for convertido para a escala Mel, calcula-se o *"cepstrum"*. O *cepstrum*, muitas vezes definido como o "espectro do espectro", é uma operação matemática que consiste em aplicar a transformada discreta do cosseno (*Discrete cosine transform*, DCT) no logaritmo da energia (neste caso, da saída de cada filtro).

A equação 2.3 descreve, matematicamente, o processo de obtenção dos coeficientes MFCC:

$$c_n = \sum_{k=1}^{K} log(S_k) cos[n(k - \frac{1}{2})\frac{\pi}{K}], n = 1, ..., L \tag{2.3}$$

onde $L$ é o número de coeficientes e $S_k$ os coeficientes de potência da saída do k-ésimo filtro. Frequentemente, descarta-se algumas das últimas amostras obtidas da DCT,

| Índice dos Filtros | Frequência central (Hz) | Índice dos Filtros | Frequência central (Hz) |
|---|---|---|---|
| 1 | 100 | 13 | 1516 |
| 2 | 200 | 14 | 1741 |
| 3 | 300 | 15 | 2000 |
| 4 | 400 | 16 | 2297 |
| 5 | 500 | 17 | 2639 |
| 6 | 600 | 18 | 3031 |
| 7 | 700 | 19 | 3482 |
| 8 | 800 | 20 | 4000 |
| 9 | 900 | 21 | 4595 |
| 10 | 1000 | 22 | 5278 |
| 11 | 1149 | 23 | 6063 |
| 12 | 1320 | 24 | 6964 |

Tabela 1 – Banco de Filtros Mel (MARTINS; YNOGUTI, 2014, p. 3)

pois estas, contêm pouca informação sobre o formato do trato vocal utilizado na produção da voz (DAVIS, 1980). Com o vetor de coeficientes (características acústicas) pronto, é possível iniciar o processo de reconhecimento com RNAs, descrito na seção 2.4.

## 2.4 Redes Neurais Artificiais

Conforme descrito na seção 2.2, após a etapa de pré-processamento, segue-se a etapa de identificação. Nesta etapa, utiliza-se RNAs para gerar um classificador, capaz de reconhecer o dígito correspondente ao vetor de características do sinal de voz analisado.

### 2.4.1 Caracterização de uma Rede Neural Artificial

Uma Rede Neural Artificial (RNA) consiste em um sistema computacional de processamento paralelo e distribuído que possui a habilidade de aprender e armazenar conhecimento experimental (CINTRA; VELHO; COCKE, 2016, p. 2). As RNAs são organizadas em camadas, que por sua vez são compostas por um conjunto de unidades de processamento conhecidos por neurônios. Conforme observado na figura 7, cada RNA é composta por três tipos distintos de camadas: a camada de entrada, um conjunto de camadas ocultas (ou intermediárias) e uma camada de saída.

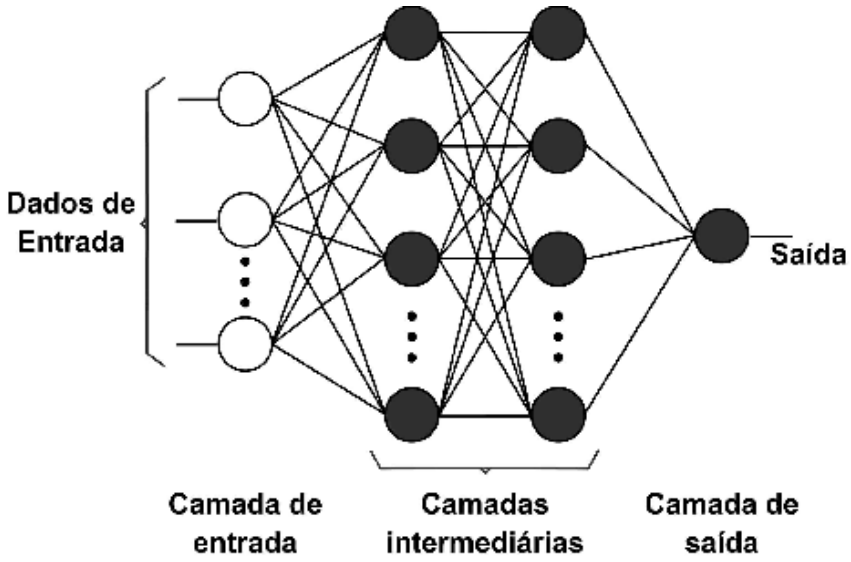


Figura 7 – RNA (FIORIN et al., 2011, Figura 5)

1. **Camada de Entrada:** inicialmente, os dados são apresentados a RNA na camada de entrada, e por isso, o número de neurônios contidos nessa camada deve ser igual ao número máximo de coeficientes obtidos após a etapa de extração de características.

2. **Camadas Ocultas:** a quantidade de camadas ocultas e neurônios escondidos são determinados pelo projetista e variam conforme a aplicação, são nestas camadas que acontecem a maior parte do processamento dos dados.

3. **Camada de Saída:** por fim, o resultado é apresentado na camada de saída, e a quantidade de neurônios presentes nesta camada é determinada pela quantidade de possíveis classificações que os dados possam ter.

### 2.4.2 Modelo Neuronal de McCulloch-Pitts

Neurônios artificiais são unidades de processamento de informação essenciais para a operação das RNAs. Um modelo matemático destes neurônios foi inicialmente apresentado por McCulloch e Pitts (Modelo M-P) em 1943. De acordo com (ZHANG; ZHANG, 1999, p. 1), o modelo M-P consiste em um elemento com $n$ entradas e uma saída (binárias), cuja forma geral pode ser definida pela equação 2.4.

$$y = sgn(w * x - \theta) \tag{2.4}$$

$$sgn(v) = \begin{cases} 1 & , v > 0 \\ -1 & , v \leq 1 \end{cases}$$

onde $x$ é o vetor de entrada $(x_1, x_2, ..., x_n)^T$, $w$ o vetor de pesos $(w_1, w_2, ..., w_n)$ e $\theta$ o limiar.

A entrada efetiva de um neurônio M-P é resultado de uma soma ponderada, representada na equação 2.4 pelo termo $w * x$. Este resultado pode sofrer um atraso $\theta$, que é considerado no parâmetro da função transferência $y$.

### 2.4.3 Modelo Neuronal Não-Linear

O modelo neuronal não-linear, ou modelo geral de neurônio, é uma generalização do modelo M-P, onde a combinação das somas ponderadas das entradas é utilizada por uma função $\varphi$ para produzir um estado de ativação.

De acordo com (HAYKIN, 2001, p. 36), um neurônio não-linear é composto por três elementos básicos:

- Conjunto de Sinapses: conexões que ligam um neurônio aos demais, cada uma composta por um peso sináptico $(w_{kj})$, cujo produto com o sinal transmitido de um neurônio ao outro, gera o sinal de entrada do próximo neurônio. Ao contrário de uma sinapse cerebral, os pesos sinápticos de neurônios artificiais podem estar em um intervalo que inclui valores negativos e positivos.

- Somador: realiza a soma os sinais de entrada $(x_j)$ de um neurônio que são o produto da saída de um outro neurônio com o seu respectivo peso sináptico.

- Função de ativação: define a saída de um neurônio em termos do campo local induzido. Também conhecida como função restritiva, por limitar a amplitude da saída de um neurônio

Dessa forma, é possível descrever um neurônio não-linear com o seguinte par de equações:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.5}$$

$$y(k) = \varphi(u_k + b_k) \tag{2.6}$$

onde $x_j$ $(x_1, x_2, ..., x_m)$ são os sinais de entrada, $w_{kj}$ $(w_{k1}, w_{k2}, ..., w_{km})$ os pesos sinápticos do neurônio $k$, $b_k$ o bias e $\varphi$ a função de ativação.

É possível observar na figura 8 e na equação 2.6 que além dos três elementos básicos, um elemento externo ao modelo neuronal, conhecido como *bias*, é adicionado a entrada da função de ativação. O *bias* é uma ferramenta utilizada por arquitetos de RNAs para melhor aproximar o valor de saída da função de ativação com o valor esperado, e será melhor introduzido nas seções seguintes.

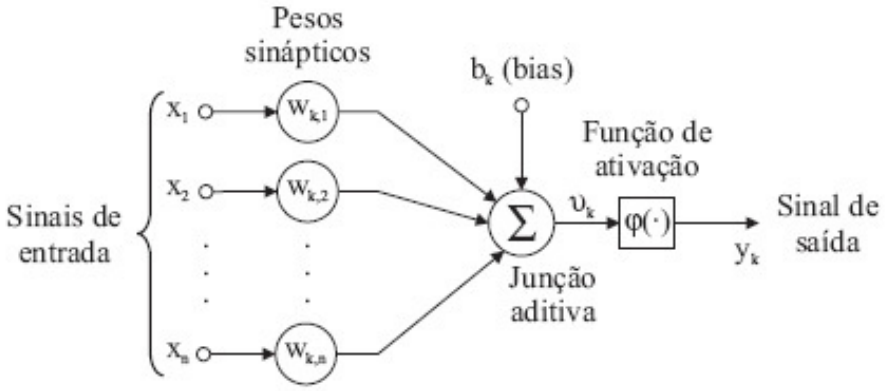


Figura 8 – Modelo Não-Linear de um Neurônio (HAYKIN, 2001, Figura 1.5)

## 2.5 Tipos de aprendizado

Conforme mencionado nas seções anteriores, a característica mais importante de uma RNA, é a sua habilidade de aprender a partir do ambiente. O processo de aprendizado consiste em adaptar os pesos sinápticos, a partir dos exemplos que são, gradativamente, apresentados a RNA. A maneira como estes pesos são adaptados depende do algoritmo de treinamento implementado, sendo um algoritmo de treinamento, um conjunto de regras bem definidas para solucionar problemas de aprendizado. De acordo com (REZENDE; MONARD; CARVALHO, 1999), o objetivo do aprendizado é extrair um classificador, a partir de uma série de exemplos, para então ser capaz de classificar outros exemplos, até então desconhecidos. Outra característica importante, é a maneira como a RNA se relaciona com o ambiente, neste contexto, os seguintes paradigmas de aprendizado devem ser considerados: o aprendizado supervisionado e o não-supervisionado.

Conforme descrito em (BATISTA, 2003, p. 19), no aprendizado supervisionado (utilizado no desenvolvimento deste trabalho), é fornecido ao sistema de aprendizado um conjunto de exemplos $E = \{E_1, E_2, ...E_N\}$, sendo que cada exemplo pertence a uma classe específica. Matematicamente pode-se dizer que cada exemplo pode ser representado por:

$$E_i = (\overrightarrow{x_i}, y_i) \tag{2.7}$$

Na equação 2.7, $\overrightarrow{x_i}$ representa um conjunto de entradas e $y_i$ a classe atribuída. O objetivo deste tipo de aprendizado é induzir um mapeamento do vetor $\overrightarrow{x_i}$ para o valor $y_i$. Dessa forma, o sistema deverá extrair um classificador que implementará uma função desconhecida $f$, que permitirá predizer valores de $y$ desconhecidos. No caso da aprendizagem não-supervisionada (ou auto-supervisionada), um conjunto de exemplos também é introduzido ao sistema de aprendizado, contudo, estes exemplos não possuem uma classificação $y$ como no aprendizado supervisionado. O objetivo deste tipo de aprendizado é desenvolver um modelo de procura por redundâncias (regularidades) nos exemplos, agrupando-os ou formando *clusters* de exemplos com características similares.

## 2.6 *Perceptron* de Camada Única

O *perceptron* de Rosenblatt, comumente denominado *perceptron* de camada única, consiste em um sistema composto por um neurônio não-linear, isto é, um neurônio M-P (seção 2.4.2) capaz de classificar um conjunto de valores (entradas) em duas classes distintas. Em 1959, no livro "*Princípios da Neurodinâmica*", Rosenblatt provou que um *perceptron* conseguirá classificar corretamente um conjunto de dados em duas classes diferentes se, estas classes forem linearmente separáveis (ou seja, podem ser divididas por um hiperplano).

Na figura 9, são ilustradas duas classes distintas, círculos hachurados ($\pi_1$) e não-hachurados ($\pi_2$), sendo os dados de entrada o conjunto de coordenadas $x_1$ e $x_2$. É possível observar que quaisquer um dos hiperplanos representados na figura separam completamente as amostras de classe $\pi_1$ e $\pi_2$, sendo portanto, possíveis soluções para o problema de classificação.

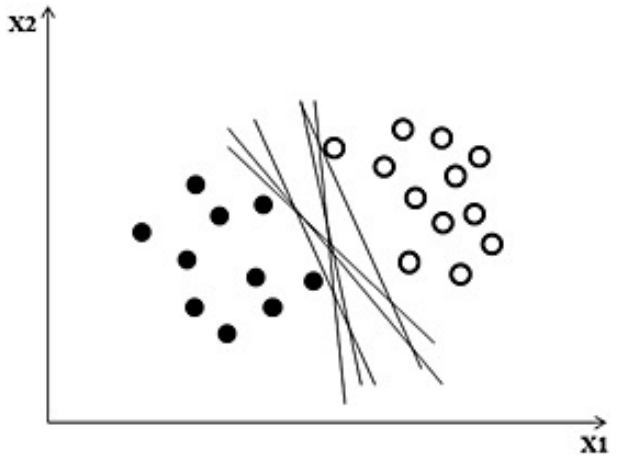


Figura 9 – Dados Linearmente Separados (ZHAO et al., 2014)

Os *perceptrons* de camada única podem ser utilizados tanto com valores contínuos ou binários. Esta rede neural simplificada ganhou muita atenção quando inicialmente apresentada, devido a sua habilidade de aprender a reconhecer padrões simples. Segundo (LIPPMANN, 1987, p. 13), o *perceptron* classifica um conjunto de entradas através de uma operação que consiste na soma ponderada dos elementos de entrada, seguido da adição de um limiar (*bias*) e finalizado pela aplicação de uma função limitadora não-linear (função de ativação) que garante que o resultado seja +1 ou -1 (representação numérica das classes $\pi_1$ e $\pi_2$). A equação 2.8 demonstra como o *perceptron* utiliza os pesos sinápticos $w_i$ para definir se os vetores de entrada $x_i$ pertencem ao conjuto $\pi_1$ ou $\pi_2$. Entretanto, o valor destes pesos não é constante, através de um processo iterativo denominado treinamento, o

*perceptron* ajusta os pesos a medida que novos exemplos são apresentados.

$$y(t) = f_n(\sum_{i=1}^{m} w_i x_i + bias) \tag{2.8}$$

O treinamento de um *perceptron* consiste em um aprendizado supervisionado, conforme visto na seção anterior, isto significa que todos os exemplos são acompanhados da classe a qual este pertence. Com esta informação, é possível quantificar o erro do *perceptron* subtraindo o valor numérico da classe correspondente (+1 ou -1), pelo valor presumido pelo *perceptron* (equação 2.8).

$$w_i(t+1) = w_i(t) + \eta[d(t) - y(t)]x_i(t) \tag{2.9}$$

Na equação acima, a constante $\eta$ representa a taxa de aprendizado da rede, este parâmetro é utilizado pelos algoritmos de treinamento para tratar o problema do mínimo local e será melhor explicitado nas seções seguintes. Durante o processo de treinamento, os pesos sinápticos $w_i$ são adaptados de iteração para iteração, sendo que em cada iteração, um novo exemplo é utilizado. A equação 2.9 demonstra como o *perceptron* utiliza o erro, representado pelo termo $[d(t) - y(t)]$, para ajustar os pesos da rede.

## 2.7 *Perceptron* de Múltiplas camadas

A maior desvantagem de se utilizar o *perceptron* elementar (de camada única), é justamente sua maior limitação. Além de o *perceptron* de Rosenblatt não conseguir classificar os dados de entrada em mais de duas classes, também não consegue resolver problemas linearmente não-separáveis.

Um problema clássico linearmente não-separável, e portanto, não solucionável com um perceptron de camada única, é a operação booleana ou-exclusivo (XOR).

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Tabela 2 – Tabela Verdade do Ou-Exclusivo

Na tabela acima, $A$ e $B$ são um conjunto de entradas que pode ser classificado como 0 ou 1, todavia, não existe um hiperplano que separe totalmente as duas classes. Este fato pode ser observado com mais clareza se considerarmos as entradas, como um conjunto de coordenadas, e as inserirmos em um plano cartesiano.

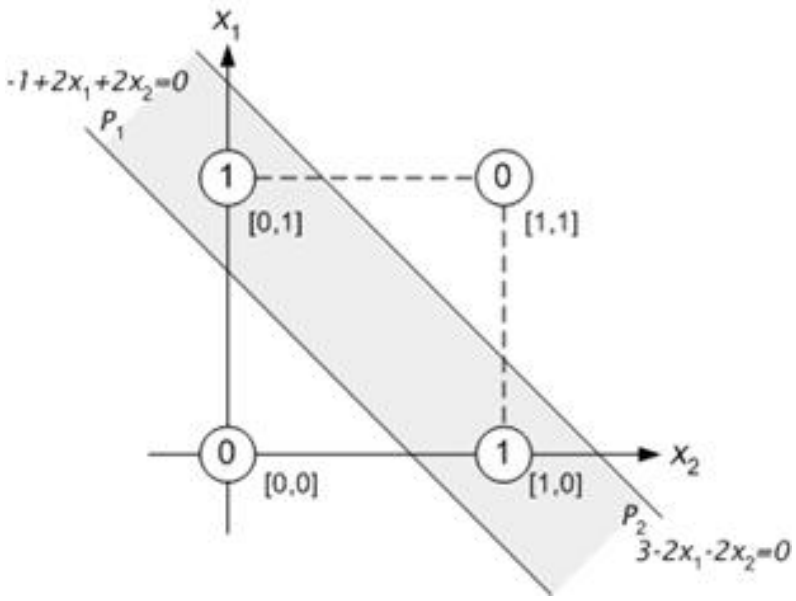


Figura 10 – XOR separado por dois hiperplanos (byclb.com, Acessado 2017-10-17)

Conforme demonstrado na figura 10, é necessário, pelo menos, dois hiperplanos para categorizar corretamente as duas classes. Uma possível solução, seria classificar como 1, todos os pontos entre as retas $-1 + 2x_1 + 2x_2 = 0$ e $3 - 2x_1 - 2x_2$, e todo o resto como 0. Concluí-se então, que o uso de um único neurônio produz apenas uma fronteira de decisão, e que, para resolver problemas mais complexos, é necessário utilizar mais neurônios.

O *perceptron* de múltiplas camadas (MLP), pode ser considerado como uma generalização do *perceptron* de camada única (SLP). Conforme descrito no trabalho de (GARDNER; DORLING, 1998, p. 2627), um MLP, assim como o SLP, não faz qualquer suposição sobre o conjunto de dados analisados e pode ser treinado para aproximar, com uma tolerável taxa de erro, qualquer função mensurável. É a superposição de várias funções não-lineares simples que permite ao MLP aproximar funções extremamente não-lineares, e a sua principal utilidade é ser treinado com um conjunto de dados conhecido, para posteriormente, classificar dados desconhecidos.

Em (HAYKIN, 2001), Haykin afirma que os MLPs foram aplicados com sucesso em diversas áreas para resolver problemas difíceis, através de um treinamento supervisionado com um algoritmo muito popular conhecido como *algoritmo de retropropagação de erro (error back-propagation)*.

Na aprendizagem por retropropagação de erro, gera-se um sinal, conhecido como sinal funcional, que tem por objetivo, propagar o conjunto de entradas por todos os neurônios da rede, camada por camada, para produzir então, o conjunto de saídas. É possível definir o sinal funcional que aparece na saída de um neurônio $j$, como:

$$d_j(n) = \varphi_j(\sum_{i=0}^{m} w_{ij}(n)x_i(n)) \tag{2.10}$$

onde $\varphi_j$ é a função de ativação, $w_{ij}$ o peso sináptico entre os neurônios $i$ e $j$, e $x_i$ a entrada oriunda do neurônio $i$.

Uma vez que o algoritmo de retropropagação tenha finalizado o cálculo do sinal funcional para todos os neurônios da iteração, este computa o sinal de erro que será propagado no sentido inverso ao das conexões sinápticas. O sinal de erro é utilizado para realizar uma correção em cada um dos pesos sinápticos da rede, e pode ser calculado através da seguinte equação:

$$e_j(n) = d_j(n) - y_j(n) \tag{2.11}$$

onde $d_j$ é a saída desejada, e $y_j$ o sinal funcional do neurônio $j$, conforme a equação 2.10.

Ainda de acordo com (HAYKIN, 2001, p. 190), é possível definir a correção aplicada a um peso sináptico entre um neurônio $i$ e $j$ como:

$$\Delta w_{ij}(n) = \eta\delta_j(n)y_i(n) \tag{2.12}$$

onde $\eta$ é a taxa de aprendizagem, e $\delta_j$ o gradiente local (calculado a partir do sinal de erro).

Conforme visto nesta seção, o MLP utiliza diversos parâmetros diferentes para realizar o treinamento da rede, no capítulo 3, é descrito o funcionamento de um *software* desenvolvido especificamente para variar os valores dos parâmetros de um MLP, e no capítulo 4, os valores utilizados e os resultados dos experimentos são analisados.

# 3 Desenvolvimento

Sistemas de reconhecimento automático de voz (RAVs) são sistemas extremamente complexos que dependem da seleção adequada de seus parâmetros para alcançarem níveis de desempenho razoáveis. Nos próximos capítulos são descritos os processos utilizados para extrair as características acústicas dos dados de voz e treinar as diferentes arquiteturas de RNA.

## 3.1 Pré-Processamento

Conforme descrito na seção 2.3, o Pré-Processamento é a primeira etapa em um sistema RAV. Nesta etapa, é realizada a conversão do sinal, o processo de pré-ênfase, a análise espectral e a extração de características acústicas. Contudo, utilizou-se uma base de dados cujos sinais de voz, já haviam sido preparados, ou seja, já haviam passado pelo processo de conversão do sinal.

### 3.1.1 Base de Dados

A base de dados utilizada em todos os experimentos deste trabalho, foi projetada e coletada pela *Texas Instruments* em 1982. Esta base de dados contêm cerca de 25 mil sequências de dígitos, falada no idioma inglês-americano, por 326 locutores: 111 homens, 114 mulheres, 50 meninos e 51 meninas (LEONARD, 1984). Entretanto, destes locutores, as crianças foram removidas devido ao seu trato vocal ser muito parecido com o de mulheres (MARTINS, 2014, p. 14), isso evita que a RNA seja super treinada para mulheres (voz aguda). Também foi garantido que apenas um locutor de cada região fosse utilizado, isto porquê, utilizar muitos locutores da mesma região, deixaria a rede orientada a sotaque, ou seja, que sotaques de regiões com mais locutores influenciassem mais do que outros. Na tabela 3 é detalhado a quantidade de amostras de voz utilizadas nos experimentos.

| **Sexo** | **Amostras** | **Faixa etária** |
|---|---|---|
| Homens | 1290 | 21 - 70 |
| Mulheres | 780 | 17 - 59 |

Tabela 3 – Base Utilizada

Os dados foram coletados em um ambiente silencioso e digitalizados em 20 kHz com 16 bits de resolução. Para todos os experimentos, as amostras foram ajustadas para uma taxa de amostragem de 8 kHz, valor amplamente utilizado em trabalhos da área para fins de comparação de resultados, como em (LEE; ROSE, 1998), (MERTINS; RADEMACHER, 2005) e (BURNETT; FANTY, 1996).

As amostras coletadas de cada locutor consistem em sequências de dígitos. No total, 77 amostras foram coletadas, sendo possível dividi-las da seguinte forma:

- 11 dígitos isolados (*oh, zero, one, two, three, four, five, six, seven, eight, nine*), cada um repetido duas vezes.
- 11 sequências de 2, 3, 4, 5 e 7 dígitos consecutivos.

Apesar da grande variedade de sequências de dígitos presentes nesta base de dados, o foco deste trabalho é diferenciar apenas dígitos isolados e, pelas mesmas limitações mencionadas anteriormente, apenas uma parcela dos dígitos isolados foi realmente utilizada nos experimentos. No total, foram utilizadas 78 amostras de mulheres e 129 amostras de homens para cada um dos 10 dígitos (excluiu-se o dígito *oh*).

### 3.1.2 Análise Espectral e Extração de Características

Nesta etapa, o objetivo é utilizar o método MFCC (seção 2.3.3) para ressaltar os aspectos da informação que contribuem para a identificação de diferenças entre os dígitos, e remover quaisquer outros dados que não sejam úteis.

Para obter o vetor de característica de cada sinal de voz, utilizou-se o *software* HTK (*Hidden Markov Model Toolkit*). De acordo com o site oficial do HTK (HTK, Acessado 24/10/2017), este foi desenvolvido originalmente pelo departamento de engenharia da Universidade de Cambridge e, posteriormente, teve os direitos de distribuição adquiridos pela *Microsoft*, que permite o livre uso do mesmo.

O HTK consiste em um conjunto de ferramentas para desenvolver e manipular modelos ocultos de Markov (HMM), que é um método de modelagem estatística muito utilizado em sistemas RAV. Apesar de HMMs não ser o foco deste trabalho, o HTK possui uma série de ferramentas que implementam diferentes métodos de extração de características acústicas, como o MFCC.

De acordo com (MESEGUER, 2009), todas as ferramentas disponíveis no HTK foram construídas, principalmente, através de módulos e bibliotecas. Estas ferramentas são executadas através de comandos *shell*, que servem como abstração para os módulos e bibliotecas de *software*, bem como para os diversos formatos de arquivos que o HTK utiliza para manter a integridade dos dados, e otimizar os processos.

Para copiar e manipular arquivos de voz, o HTK disponibiliza a ferramenta HCopy. O HCopy utiliza um arquivo de configuração para parametrizar os dados de voz, em suma, os dados são lidos diretamente de arquivos, parametrizados conforme o arquivo de configuração e o resultado copiado para arquivos de saída, conforme esquematizado na figura 11.

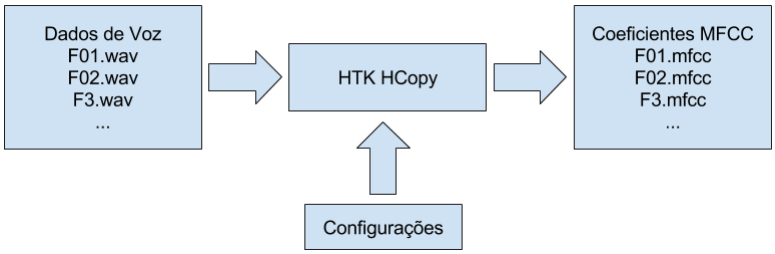


Figura 11 – Modo de uso do HTK HCopy

Os parâmetros presentes no arquivo de configuração especificam as características do processo de extração MFCC, e os valores utilizados em cada configuração foram baseados em outros trabalhos da área, que também realizaram processamento MFCC sobre a base TIDIGITS, (MESEGUER, 2009) e (MARTINS, 2014). Uma vez que todos os arquivos necessários forem criados, é possível extrair os vetores MFCC dos dados de voz. Para entender os resultados obtidos do HTK, é preciso descrever as configurações utilizadas e seus respectivos valores, conforme a tabela 4.

| **Configuração** | **Valor** |
|---|---|
| TARGETKIND | MFCC_0 |
| SOURCERATE* | 625 |
| SOURCEFORMAT | WAV |
| PREEMCOEF | 0,97 |
| TARGETRATE* | 10000 |
| WINDOWSIZE* | 250000 |
| USEHAMMING | T |
| NUMCHANS | 24 |
| NUMCEPS | 12 |

Tabela 4 – Configuração do HCopy (*expresso em unidades de 100ns)

As configurações *TARGETKIND* e *SOURCEFORMAT* indicam ao HCopy a taxa de amostragem dos dados de voz e que a extração dos coeficientes MFCC deve utilizar primeiro coeficiente ($C_0$) como componente de energia. Já o formato dos dados de voz, é definido como arquivos de extensão .WAV pelo parâmetro *SOURCEFORMAT*, devido ao fato de que os dígitos isolados disponíveis na base de dados TIDIGITS se encontram neste padrão.

Conforme visto na seção 2.3.1 é necessário a aplicação de um filtro de pré-ênfase antes da análise espectral. O valor do coeficiente de pré-ênfase da equação 2.1 é definido pela configuração *PREEMCOEF*. E o processo de janelamento é descrito pelas configurações TARGETRATE, WINDOWSIZE e USEHAMMING, que foram configuradas para aplicar uma janela de Hamming de 25ms a cada 10ms, resultando em uma sobreposição de 15ms. Por fim, para computar os coeficientes MFCC, o HCopy aplica a FFT seguida de uma DCT (YOUNG et al., 2002), o número de canais nos bancos de filtro foi configurado

pelo parâmetro NUMCHANS e o número de coeficientes MFCC foi especificado pela configuração NUMCEPS.

Uma vez que se tenha utilizado o *HCopy* para extrair os vetores de características acústicas, é necessário converter os dados para um formato adequado às RNAs, afinal, o *HTK* possui uma forma específica de apresentar os resultados obtidos. O processo de preparação dos dados está descrito na seção 3.2.

## 3.2 Treinamento

Neste capítulo é descrito os processos necessários para converter os dados gerados pelo HTK, para uma base de dados que uma RNA possa utilizar, bem como o algoritmo de força bruta desenvolvido para avaliar o desempenho dos parâmetros da RNA.

### 3.2.1 Preparação dos Dados

Segundo (YOUNG et al., 2002, p. 69), o *HTK* apresenta os resultados através de arquivos binários, compostos por uma sequência de amostras precedidas por um cabeçalho. O cabeçalho dos arquivos binários produzidos tem um total de 12 *bytes* contendo as seguintes informações:

| **Informação** | **Tamanho** | **Descrição** |
|---|---|---|
| nSamples | 4 *bytes* | Número de amostras em um arquivo |
| sampPeriod* | 4 *bytes* | Período de amostragem |
| sampSize | 2 *bytes* | Número de *bytes* por amostra |
| parmKind | 2 *bytes* | Código indicador do tipo de amostra |

Tabela 5 – Informações presentes no cabeçalho *HTK* (*em unidades de 100ns)

Cada dígito isolado processado pelo *HTK* passa pelo processo de extração do vetor de características do arquivo binário. Este processo consiste em converter todos os *bytes* entre o fim do cabeçalho e o final do arquivo, em um vetor de tamanho *nSamples*, como ilustrado pela figura 12:

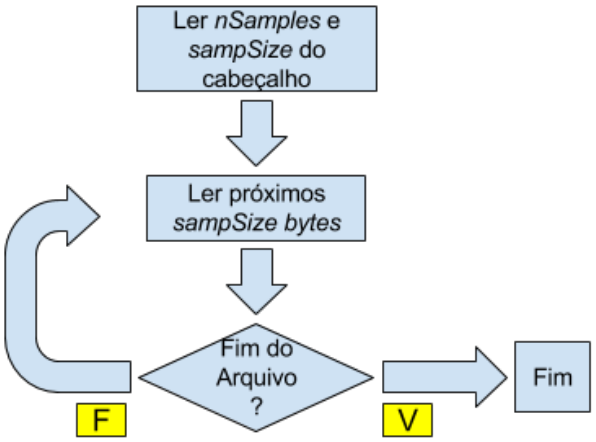


Figura 12 – Algoritmo de Extração de Coeficientes do Arquivo binário

Primeiramente, lê-se o tamanho e a quantidade de amostras presentes, respectivamente *sampSize* e *nSamples*. Em seguida, inicia-se o *loop* de leitura das amostras, onde interpreta-se os próximos 4 *bytes* como um valor *float*, armazena-o em um vetor, e continua o processo até não haverem mais amostras a serem lidas.

Cada amostra presente no vetor resultante, representa um coeficiente melcepstral, que é utilizado para identificar similaridades entre os dígitos isolados. Entretanto, cada exemplo processado pelo *HTK* produz um vetor de tamanho diferente, o que representa uma dificuldade para a RNA, dado que os exemplos introduzidos devem possuir o mesmo tamanho. Para aumentar o tamanho dos vetores de característica menores, é possível repetir os coeficientes melcepstrais nas regiões estáveis da palavra. Ou seja, calcula-se a distância euclidiana entre dois vetores consecutivos e se insere o valor obtido entre aqueles que apresentarem menor distância, até que ambos possuam o mesmo tamanho.

### 3.2.2 Normalização e Aleatorização dos Exemplos

Cada vetor de característica é utilizado pela RNA como um exemplo para reconhecer dígitos isolados, todavia, os coeficientes presentes no vetor, possuem magnitudes diferentes, o que pode prejudicar o processo de treinamento.

Por exemplo, uma RNA cuja função é prever o preço de uma casa baseado no tamanho do terreno, quantidade de quartos e quantidade de banheiros. Cada um destes parâmetros seria convertido para um neurônio da camada de entrada, e seus valores considerados durante a etapa de ajuste dos pesos sinápticos (descrito na seção 2.7). Como os exemplos apresentados a esta RNA estão, claramente, em magnitudes diferentes (número de quartos pode ser 1 e o tamanho da casa 100 metros quadrados), o processo de aprendizagem seria prejudicado devido a maior relevância atribuída aos valores de maior magnitude.

Para resolver este problema, normaliza-se os exemplos processados para uma faixa dinâmica (neste caso, entre -1 e 1), conforme a equação 3.2.

$$z_i = 2 * \frac{x_i - min(x)}{max(x) - min(x)} - 1 \tag{3.1}$$

Após a etapa de normalização dos coeficientes melcepstrais, o dado esta pronto para ser introduzido a RNA. Entretanto, conforme (DESAI et al., 2010, p. 957) menciona em seu trabalho, a ordem em que os exemplos são apresentados a RNA é muito importante para evitar que a rede se torne tendenciosa (*biased*). Se todos os exemplos de uma categoria específica (por exemplo, o dígito 1), forem introduzidos a RNA em sequência, quando um exemplo de outra categoria for apresentado (por exemplo, o dígito 2), a RNA será tendenciosa durante a etapa de ajuste de pesos. Para prevenir este tipo de situação, é necessário aleatorizar a ordenação dos exemplos, para distribuir as diferentes categorias entre a ordem de apresentação.

### 3.2.3 Implementação e Configuração da RNA

Neste trabalho, desenvolveu-se um *software* capaz de realizar o processo de treinamento de uma RNA variando seus parâmetros dentro de uma faixa pré-definida pelo usuário. Esta faixa foi escolhida através de variações empíricas dos valores padrão (*default*) da biblioteca de *software* FANN (NISSEN, 2003), utilizada devido a grande quantidade de recursos e ferramentas disponíveis. Com este *software*, comparou-se diferentes arquiteturas de RNA através da variação dos seus parâmetros, e com a utilização da biblioteca FANN, foi possível manipular quatro parâmetros diferentes: função de ativação, algoritmo de treinamento, taxa de aprendizado e número de neurônios e camadas ocultas.

Basicamente, o *software* analisa a faixa de parâmetros informada e compila as arquiteturas de RNA que serão testadas. Por exemplo, se o usuário informar 5 valores diferentes de cada um dos quatro parâmetros, o *software* compilará $5^4$ arquiteturas e as testará sequencialmente (uma após a outra). O processo de execução de uma arquitetura de RNA pode ser dividido em duas etapa:

1. **Treinamento:** processo iterativo responsável por ajustar os pesos sinápticos (seção 2.6).
2. **Teste:** etapa onde calcula-se a taxa de erro de palavra (WER, *word error rate*), com exemplos que não foram introduzidos durante a etapa de treinamento, para verificar a eficiência da RNA.

O *software* desenvolvido utiliza uma ferramenta estatística para seleção de modelos conhecida como validação cruzada. De acordo com (GENCAY; QI, 2001, p. 727), "Em validação cruzada, os dados são divididos em três subconjuntos: treinamento, validação e teste. O subconjunto de treinamento é utilizado para ajustar os pesos sinápticos dos vários modelos candidatos, entre os quais o que obtiver a melhor performance no subconjunto de validação, é selecionado para ter seu desempenho medido com o subconjunto de teste."

Em suma, em cada época (iteração) do processo de treinamento, testa-se a RNA com o subconjunto de validação, este teste é feito através da obtenção do MSE (Erro Médio Quadrático) que, nesta etapa, é utilizado como principal métrica para definir se a rede esta generalizando corretamente o problema, ou se esta ficando "viciada"(*overfitting*). A figura 13 mostra a evolução do MSE de treinamento quando comparado ao de validação, pode-se observar que existe um claro ponto onde a MSE de validação começa a subir e a de treinamento continua descendo. Este ponto é chamado de "parada antecipada"e é onde o *software* interrompe o treinamento, isto porque, parar o treinamento neste ponto, previne o *overfitting* da rede.

É imprescindível que se separe os dados utilizados na etapa de treinamento dos da etapa de teste. Isto porque, o classificador gerado no processo de treinamento, precisa ser

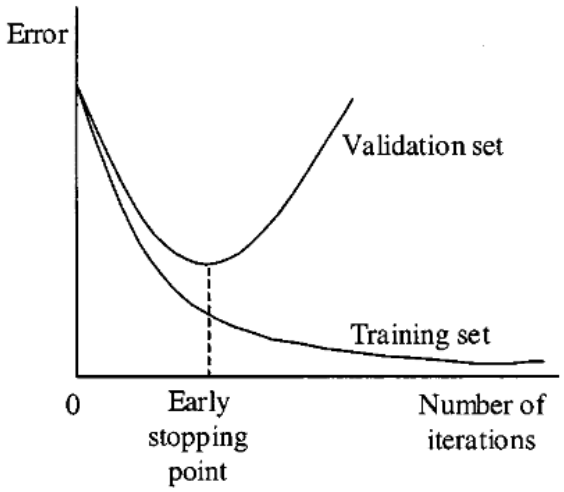


Figura 13 – Comparação da MSE de treinamento com a de validação (GENCAY; QI, 2001, p. 729)

capaz de classificar qualquer amostra que lhe for apresentada, e não apenas um conjunto específico introduzido durante o treinamento. Dessa forma, se uma base de dados diferente da de treinamento for utilizada para teste, o WER calculado representará a eficiência do classificador extraído pela RNA com dados que ainda não lhe foram apresentados.

Para calcular o WER de uma RNA, testa-se o classificador gerado pela mesma com cada um dos exemplos disponíveis na base de teste, dessa forma, o WER pode ser obtido através do percentual de classificações erradas feitas pelo classificador. Por exemplo, se uma RNA classificar corretamente 190 dos 200 exemplos presentes na base de testes, então, o classificador terá errado em 10 exemplos (ou 5%), matematicamente:

$$WER = 100 * \frac{nHits}{nExamples} \tag{3.2}$$

onde $nHits$ é o total de classificações corretas feitas pela RNA e $nExamples$ o total de exemplos presentes na base de teste. Neste projeto, utilizou-se 10% da base para teste, 20% para validação e 70% para treinamento.

Por fim, o *software* apresenta os resultados obtidos em um arquivo, onde cada linha contêm os parâmetros utilizados na RNA e seus respectivos resultados. De cada RNA testada, são extraídos quatro resultados: o tempo total necessário (para treinamento, validação e teste), o número de épocas de treinamento, a MSE e o WER. Estes dados serão apresentados na próxima seção, bem como as conclusões feitas a partir destes.

# 4 Resultados

Conforme mencionado na seção anterior, o *software* desenvolvido é capaz de manipular cinco parâmetros diferentes de uma RNA: função de ativação, algoritmo de treinamento, taxa de aprendizado e número de neurônios e camadas ocultas. Nas seções a seguir, é feita uma análise dos resultados compilados pelo *software* descrito na seção 3.2.3, bem como apresentado os valores utilizados para cada parâmetro. Todos os resultados apresentados foram obtidos utilizando quatro máquinas, cujas configurações estão descritas no apêndice A.

## 4.1 Número de Neurônios e Camadas Ocultas

Durante o desenvolvimento de qualquer aplicação que implemente uma RNA, a mesma pergunta sempre surge: quantos neurônios e camadas ocultas deverão ser utilizados? Escolher a quantidade certa de neurônios para solucionar um problema específico, pode ser um tarefa muito complicada, isto porque, poucos neurônios podem ser insuficientes para generalizar o problema e, muitos neurônios, podem transformar a RNA em um banco de memória (*overfitting*), o que faria com que a mesma não classifique corretamente dados que não foram introduzidos durante o treinamento.

Conforme (STATHAKIS, 2009, p. 2133) menciona em seu trabalho, a maioria dos trabalhos existentes utiliza modelos empíricos combinados com ajustes baseados em tentativa e erro. Neste trabalho, dividiu-se as topologias testadas em três categorias, para serem analisadas separadamente.

- **Fundamentado:** Topologias baseadas em teoremas ou regras específicas.
- **Empírico (1 ou 2 camadas):** Topologias de uma e duas camadas ocultas que possuem o número de neurônios baseado no tamanho das camadas de entrada e saída.
- **Empírico (3 ou mais camadas):** Topologias com três ou mais camadas ocultas que também possuem o número de neurônios baseado no tamanho das camadas de entrada e saída.

### 4.1.1 Fundamentado

Em 1957, Andrei Kolmogorov publicou um artigo (KOLMOGOROV, 1957) que provou que qualquer qualquer função definida em um cubo n-dimensional pode ser representado pelas somas e superposições de funções contínuas de uma variável. Posteriormente,

na década de 1980, Hecht-nielsen extendeu este conceito para provar que qualquer função pode ser representada por uma RNA de camada única contendo $2n + 1$ neurônios ocultos, onde $n$ é o quantidade de neurônios na camada de entrada (HECHT-NIELSEN, 1987).

Apesar do grande progresso feito por Hecht-nielsen, esta topologia de RNA além de conter muitos neurônios, não se adapta muito bem as diferentes funções de ativação que um projetista pode escolher utilizar. Por este motivo, Huang sugeriu admitir uma pequena quantidade de erro e distribuir menos neurônios em uma topologia com duas camadas ocultas, o que diminuiria drasticamente o tempo de treinamento e se adaptaria melhor à outras funções de ativação (HUANG, 2003). As equações 4.1 e 4.2 apresentam o número de neurônios ocultos na primeira e segunda camada respectivamente, conforme proposto por Huang.

$$L1 = \sqrt{(m+2)N} + 2\sqrt{\frac{N}{m+2}} \tag{4.1}$$

$$L2 = m\sqrt{\frac{N}{m+2}} \tag{4.2}$$

onde $N$ é o número de exemplos disponíveis e $m$ a quantidade de neurônios na camada de saída.

Sabendo que o número de exemplos disponíveis para treinamento corresponde a 70% do total (ou 1449, conforme mencionado na seção 3.2.3), que a quantidade de neurônios na camada de entrada é de 732 (vetor de características MFCC) e que na camada de saída são apenas 10 neurônios (dígitos de 0 a 9), as implementações Hecht-nielsen e Huang para este projeto corresponderiam às seguintes topologias:

| - | **Hecht-nielsen** | **Huang** | |
|---|---|---|---|
| **Camada de Entrada** | 732 | 732 | |
| **Camadas Ocultas** | 1465 | 154 | 110 |
| **Camada de Saída** | 10 | 10 | |

Tabela 6 – Topologias Fundamentadas

Para comparar a performance de Hecht-nielsen e Huang, foram selecionadas as cinco melhores RNAs treinadas com estas topologias, utilizando o WER como métrica.

A figura 14 ilustra as RNAs selecionadas em um plano cartesiano, onde o eixo horizontal corresponde ao tempo de treinamento (em segundos), e o eixo vertical ao WER (percentual).

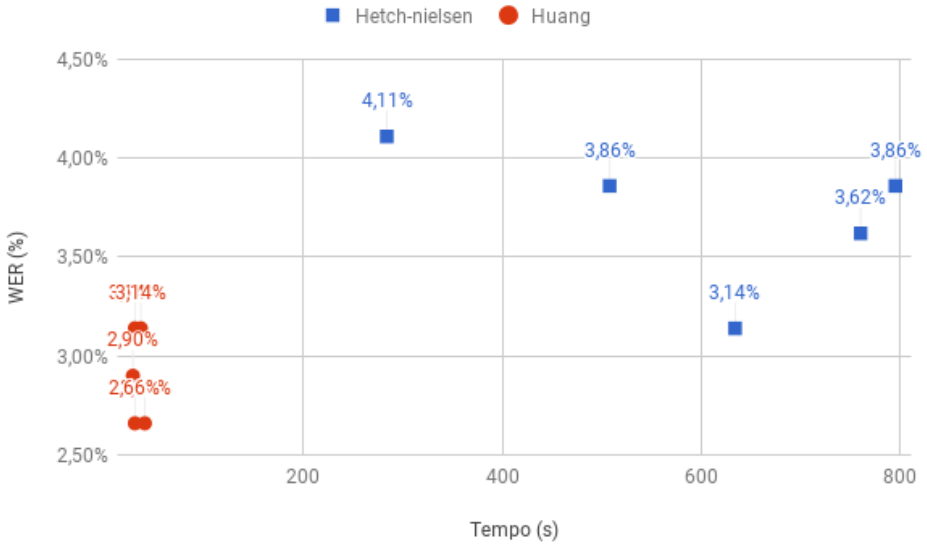


Figura 14 – Comparação Hetch-nielsen e Huang

É possível observar, que a maioria das RNAs de Hetch-nielsen demoraram mais de 10 minutos para serem treinadas, enquanto a mais demorada de Huang não chegou a 50 segundos. Também ficou claro pela análise da figura 14 que as RNAs de Huang tiveram melhor desempenho, visto que o melhor WER de Hetch-nielsen obteve o mesmo valor do pior de Huang (3,14%). Pelos experimentos realizados, também foi possível comprovar que a topologia de Huang se adapta melhor as diferentes funções de ativação. Este foi observado experimentalmente, ao analisar as funções de ativação utilizadas pelas 100 melhores RNAs, cerca de 5% do total de RNAs testada para cada topologia. No caso de Huang, seis funções diferentes apareceram, enquanto que Hetch-nielsen apresentou apenas quatro.

### 4.1.2 Empírico (1 ou 2 camadas)

Além de utilizar teoremas para determinar o número de neurônios e camadas ocultas, alguns autores sugerem utilizar equações baseadas na quantidade de neurônios presentes nas camadas de entrada e saída, que podem ser ajustadas empiricamente (STATHAKIS, 2009, p. 2135).

Neste trabalho, testou-se duas topologias empíricas com somente uma camada oculta e apenas uma com duas camadas ocultas. Nas topologias de camada única, utilizou-se as equações $\frac{m}{2}$ e $\frac{2m}{3} + n$ para definir a quantidade de neurônios ocultos, onde $m$ e $n$ são a quantidade de neurônios nas camadas de entrada e saída respectivamente. E na

topologia com duas camadas ocultas, utilizou-se um quarto da diferença de neurônios entre as camadas de entrada e saída, matematicamente: $\frac{m-n}{4}$. A tabela 7 demonstra a implementação destas topologias no contexto deste trabalho, onde a camada de entrada e a camada de saída possuem 732 e 10 neurônios respectivamente.

| | EMP1 | EMP2 | EMP3 | |
|---|---|---|---|---|
| Camada de Entrada | 732 | 732 | 732 | |
| Camadas Ocultas | 362 | 498 | 181 | 181 |
| Camada de Saída | 10 | 10 | 10 | |

Tabela 7 – Topologias Empíricas de uma e duas camadas

Assim como na seção 4.1.1, para comparar a performance destas topologias, gerou-se uma visualização (figura 15) que demonstra as cinco melhores RNAs encontradas experimentalmente, utilizando o WER como métrica.

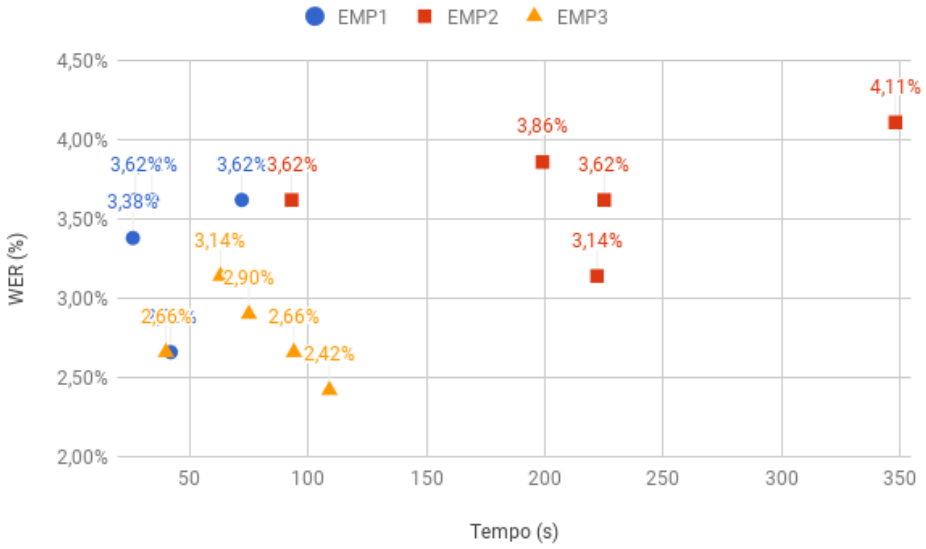


Figura 15 – Comparação Empíricas de uma e duas camadas

A partir da análise da figura 15 é possível notar que a topologia EMP2 precisou de muito mais tempo para ser treinada, devido ao fato de possuir 136 neurônios ocultos a mais que as demais topologias. Aumentar a quantidade de neurônios ocultos, não gerou uma melhora no desempenho da rede, visto que o WER da topologia EMP2 não é o menor observado, na verdade, das quinze RNAs apresentadas na figura 15, os quatro piores WERs são de RNAs com a topologia EMP2.

Em termos de configuração da RNA, a única diferença entre as topologias EMP1 e EMP3 é a divisão da quantidade de neurônios ocultos ($\frac{m}{2}$) para uma ou duas camadas. Ao calcular-se a média do tempo levado para treinar cada uma, obtêm-se 40,2 e 76,2 segundos, para EMP1 e EMP3, respectivamente. Ou seja, apenas aumentar o número de camadas (sem alterar a quantidade de neurônios), leva a um aumento substancial do

tempo de treinamento. Entretanto, este aumento pode ser compensado por uma melhora na performance, visto que o melhor WER encontrado nos experimentos realizados neste trabalho (2,42%), têm a configuração EMP3 .

### 4.1.3 Empírico (3 ou mais camadas)

Até o momento, foram observadas apenas RNAs de uma ou duas camadas ocultas, a seguir é avaliada a performance de RNAs com três ou mais camadas ocultas, variando a quantidade de neurônios em cada camada. A motivação de uso de mais camadas ocultas é para permitir a identificação dos resultados obtidos por DNNs (*Deep Neural Networks*), topologia de RNA que está sendo muito utilizada em sistemas de reconhecimento supervisionado.

Para este experimento, foram introduzidas três novas topologias, de 4, 6 e 10 camadas ocultas, discriminadas na tabela 8 como CAM4, CAM6 e CAM10 respectivamente.

| | Camada de Entrada | Camadas Ocultas | Camada de Saída |
|---|---|---|---|
| CAM4 | 732 | $\eta_1\ \eta_2\ \eta_3\ \eta_4$ | 10 |
| CAM6 | 732 | $\eta_1\ \eta_2\ \eta_3\ \eta_4\ \eta_5\ \eta_6$ | 10 |
| CAM10 | 732 | $\eta_1\ \eta_2\ \eta_3\ \eta_4\ \eta_5\ \eta_6\ \eta_7\ \eta_8\ \eta_9\ \eta_{10}$ | 10 |

Tabela 8 – Topologias Empíricas com mais de 3 camadas

O valor do parâmetro $\eta$ corresponde a quantidade de neurônios em cada camada, que por sua vez, foi testado com três valores diferentes: 32, 46 e $\frac{n}{2*\theta}$ (sendo $\theta$ o número de camadas ocultas utilizadas). O objetivo deste experimento é observar o desempenho das RNAs nas seguintes situações:

- Aumento da quantidade de neurônios sem aumentar o número de camadas.
- Aumento da quantidade de camadas sem aumentar a quantidade de neurônios.
- Aumento do número de camadas e neurônios.

Assim como nos outros experimentos, para melhor interpretar os resultados obtidos, foram geradas algumas ilustrações. A figura 16 demonstra os resultados obtidos com quatro camadas ocultas, observa-se que o número de neurônios em cada camada está indicado na legenda (para $\frac{n}{2*\theta}$, indicou-se o valor calculado, 91).

Conforme visto na seção 4.1.2, a quantidade total de neurônios influencia diretamente no tempo de treinamento, logo 32 neurônios levaram menos tempo que 46 e, 46 levaram menos tempo que 91. Contudo, ao contrário do observado na seção 4.1.2, o aumento de neurônios não levou a uma deterioração de performance. De 32 para 46 neurônios não houve grande variações no WER (em média), entretanto de 46 para 91

nota-se claramente a melhora de desempenho. Todavia, este fato pode ter ocorrido como consequência do total de neurônios ocultos presentes na nesta RNA, 364. 364 é um número muito próximo de 362, utilizado na topologia EMP3 da seção 4.1.2, que obteve a melhor performance. Em outras palavras, talvez a quantidade ótima de neurônios esteja mais perto de 364 do que os valores utilizados em outras topologias.

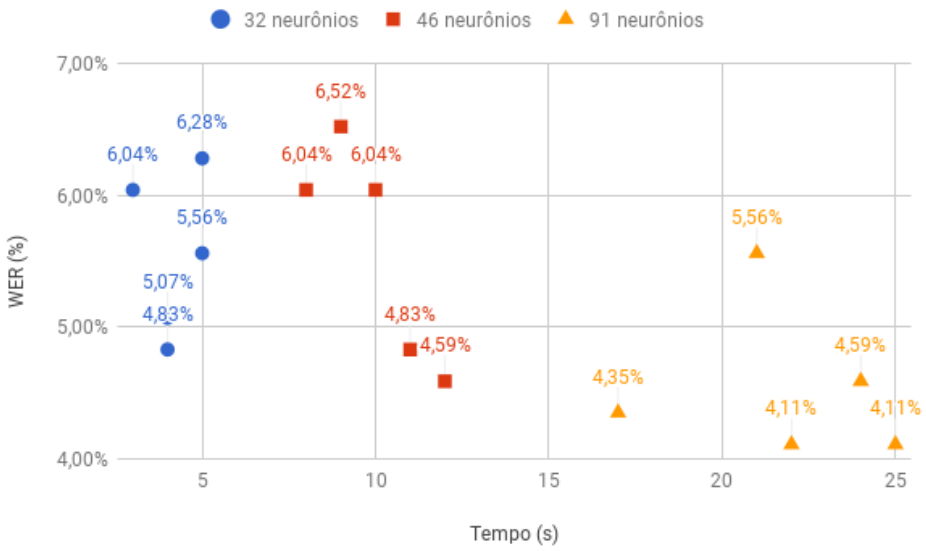


Figura 16 – Adição de neurônios, mesmo número de camadas

Na figura 17 foram ilustradas três variações da topologia com $\frac{n}{2*\theta}$ neurônios, de 4, 6 e 10 camadas ocultas. E como a quantidade de neurônios é dividida igualmente entre as camadas, todas as variações possuem a mesma quantidade de neurônios. A conclusão aqui é bem direta, a performance da rede piorou drasticamente com o aumento da quantidade de camadas, e há também uma grande diferença de performance entre $\frac{n}{2*\theta}$ de quatro camadas e o exemplo com a mesma quantidade de neurônios da seção 4.1.2, que possui apenas duas camadas. Em outras palavras, apenas adicionar camadas não contribuiu para melhorar o desempenho da RNA.

Por último, a figura 18 ilustra topologias com 32 neurônios em cada camada oculta, ou seja, há um aumento de neurônios e camadas ocultas entre as variações ilustradas.

Assim como na figura 17, a conclusão é bem direta, além do esperado aumento de tempo de treinamento, houve uma grande degradação de performance ao aumentar quantidade de neurônios e camadas ocultas simultaneamente.

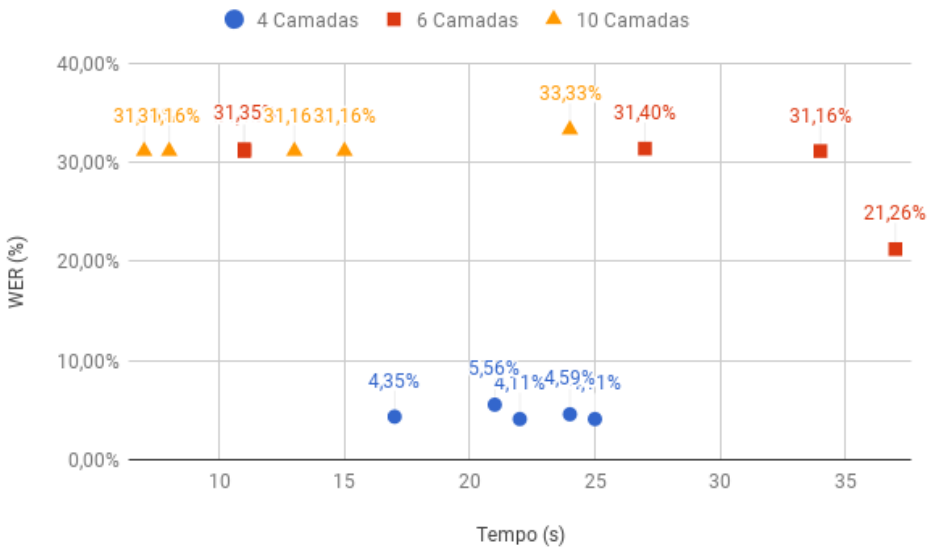


Figura 17 – Adição de camadas, mesmo número de neurônios

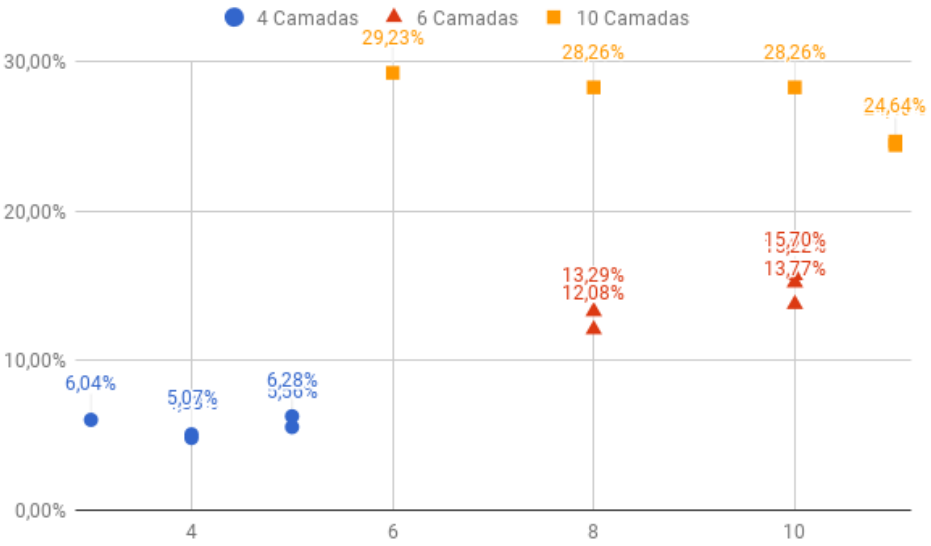


Figura 18 – Adição de camadas e neurônios

## 4.2 Algoritmo de Treinamento e Taxa de Aprendizado

De acordo com (NISSEN, 2003, p. 17), o algoritmo de treinamento é responsável por ajustar os pesos sinápticos propagando o erro no sentido inverso, da camada de saída para a camada de entrada. O erro inversamente propagado pode ser calculado para um padrão de treinamento único (incremental), ou como a soma dos erros de um conjunto de treinamento (lote ou *batch*). Por padrão, a biblioteca FANN utiliza o algoritmo de treinamento RPROP, que é do tipo lote, contudo, esta biblioteca também oferece suporte a um algoritmo do tipo incremental e outros dois do tipo lote, que são descritos em mais detalhes no apêndice B. Observe a biblioteca FANN disponibiliza uma implementação de um algoritmo incremental chamado INCREMENTAL e outro do tipo lote chamado LOTE.

Os principais objetivos desta seção são identificar os melhores algoritmos de treinamento e o efeito da variação da taxa de aprendizado na performance da RNA. Na tabela 9 estão listados os algoritmos de treinamento utilizados neste trabalho e seus respectivos tempo de treinamento e WER, bem como a taxa de aprendizado estipulada, com exceção dos algoritmos RPROP e QUICKPROP, por não utilizarem a taxa de aprendizado.

Para a obtenção destes dados, utilizou-se a função de ativação sigmóide, a topologia EMP3 (que alcançou os melhores resultados nos experimentos da seção 4.1.2), e variou-se a taxa de aprendizado empiricamente entre os valores 0,01, 0,1, 0,3, 0,7 e 1.

| **Algoritmo de Treinamento** | **Taxa de Aprendizado** | **WER (%)** | **Tempo (s)** |
|---|---|---|---|
| LOTE | 0,01 | 31,16 | 118 |
| LOTE | 0,1 | 9,9 | 120 |
| LOTE | 0,3 | 5,8 | 107 |
| LOTE | 0,7 | 5,71 | 79 |
| LOTE | 1 | 50,72 | 32 |
| INCREMENTAL | 0,01 | 2,66 | 40 |
| INCREMENTAL | 0,1 | 12,32 | 37 |
| INCREMENTAL | 0,3 | 19,81 | 41 |
| INCREMENTAL | 0,7 | 26,09 | 42 |
| INCREMENTAL | 1 | 30,92 | 41 |

Tabela 9 – Resultados Variando a Taxa de Aprendizado

A taxa de aprendizado é um dos parâmetros utilizados pelo algoritmo de treinamento para ajustar os pesos sinápticos, demonstrado na equação 2.12 da seção 2.7. Todavia, variou-se o valor deste parâmetro empiricamente, com o objetivo de encontrar RNAs que convergissem mais rapidamente ao mínimo global.

É possível observar que houve significativa melhora no WER do algoritmo INCREMENTAL, isto porquê, utilizar taxas de aprendizado baixas, evita que o processo de treinamento fique preso no mínimo local. Entretanto, não é possível observar o mesmo

efeito no algoritmo de treinamento LOTE. Este algoritmo, por não alterar os pesos diversas vezes a cada época, pode fazer o uso de otimizações globais que não são possíveis com algoritmos incrementais. No entanto, conforme (NISSEN, 2003, p. 17), não existe resposta clara para a pergunta de qual seria o melhor algoritmo de treinamento. Para algumas topologias de RNAs, algoritmos incrementais podem encontrar o mínimo global mais eficientemente, para outras, o excesso de atualizações dos pesos levaria a uma perda de performance que seria evitada com algoritmos do tipo lote.

Contudo, apesar de não se poder inferir qual o melhor algoritmo, é possível comparar os melhores resultados obtidos para cada um. Na tabela 10, foi listado os melhores WER alcançados, utilizando as funções de ativação descritas no apêndice C e as topologias utilizadas nos experimentos da seção 4.1.

| **Algoritmo de Treinamento** | **WER (%)** | **Tempo (s)** |
|---|---|---|
| LOTE | 2,42 | 109 |
| RPROP | 2,66 | 31 |
| QUICKPROP | 5,56 | 50 |
| INCREMENTAL | 2,66 | 40 |

Tabela 10 – Melhores Resultados obtidos para cada Algoritmo

É possível observar que, apesar de o algoritmo LOTE obter o melhor WER (2,42%), este levou três vezes mais tempo para ser treinado do que o RPROP, que obteve um resultado muito próximo do melhor alcançado (diferença de apenas 0,24%). No caso do algoritmo INCREMENTAL, este também obteve bons resultados, tanto de WER quanto de tempo de treinamento, contudo, nenhum destes foi superior a performance do RPROP. E por fim, o QUICKPROP obteve o pior WER e o segundo pior tempo de treinamento (50 segundos), devido ao fato de que este converge muito rápido, não alcançando o mínimo global e dificultando para a RNA classificar corretamente os exemplos.

## 4.3 Função de Ativação

Como descrito na seção 2.4, a função de ativação é um componente básico do modelo neuronal não-linear, cuja função consiste em transformar um nível de ativação em um sinal de saída. Nesta seção, analisa-se as funções de ativação disponíveis na biblioteca FANN (descritas no apêndice C), excluindo as normalizadas entre 0 e 1.

Na tentativa de encontrar a função de ativação que melhor se adapta aos diferentes algoritmos de treinamento, foi calculada a média do tempo de treinamento e WER das melhores redes alcançadas com cada algoritmo (para cada função de ativação). A tabela 11 ilustra as médias calculadas.

| **Função de Ativação** | **WER (%)** | **Tempo (s)** |
|---|---|---|
| Linear | 31,52 | 130 |
| Linear Limitada | 7,85 | 40 |
| Sigmóide | 24,88 | 47 |
| Sigmóide Progressiva | 5,49 | 27 |
| Elliot | 4,83 | 46 |
| Senoidal | 11,05 | 40 |
| Cossenoidal | 32,75 | 50 |

Tabela 11 – Média dos Melhores Resultados Obtidos por Função de Ativação

A função de ativação sigmóide é muito complexa e, computacionalmente, difícil de ser calculada. Por este motivo, a biblioteca FANN disponibiliza uma implementação menos precisa desta função, no intuito de diminuir significativamente seu tempo de treinamento. A função Sigmóide Progressiva é uma destas implementações menos precisas e, conforme demonstrado pela comparação do seu tempo de treinamento médio (27 segundos) com o da função Sigmóide tradicional (47 segundos), foi possível comprovar experimentalmente que há uma grande redução de tempo de treinamento (a Sigmóide Progressiva levou cerca de 42,55% menos tempo).

Entretanto, a função Sigmóide Progressiva não obteve o melhor WER. A função de ativação que alcançou o melhor WER médio foi a função Elliot. Esta função se aproxima da função sigmóide para valores pequenos, contudo, possui uma curva mais suave para valores grandes. A função Elliot necessitou de mais épocas para convergir, principalmente quando utilizada com valores próximos dos seus extremos (neste caso -1 e 1).

Porém, apesar de a função Elliot ter alcançado o melhor WER médio, esta não foi utilizada na arquitetura que obteve o melhor WER. O melhor WER geral alcançado em todos os experimentos deste trabalho (2,42%), foi obtido utilizando a função de ativação senoidal em conjunto com o algoritmo de treinamento LOTE.

# 5 Conclusões

O foco principal deste trabalho foi apresentar um modelo de sistema de reconhecimento de voz capaz de identificar palavras isoladas (dígitos de 0 a 9, pronunciados em inglês), utilizando MFCC para extrair o vetor de características acústicas e avaliando a performance de diferentes topologias e configurações de RNAs. Dentre as diversas topologias e configurações de RNAs treinadas durante o desenvolvimento deste projeto, a melhor performance obtida foi de 2,42% de WER, uma elevada taxa de acerto, mesmo que inferior a técnicas mais tradicionais e bem estabelecidas no meio científico, como o HMM, que no trabalho de (MARTINS, 2014) chegou a um WER de 1,88%.

Baseando-se nos resultados apresentados na seção 4, é possível afirmar que a escolha de RNAs, mais especificamente MLPs, para a etapa de reconhecimento, produziu bons resultados. Os experimentos comprovaram a eficácia da aplicação de RNAs à problemas complexos, e também permitiram analisar a influência dos seus parâmetros no WER, sendo eles: número de neurônios, número de camadas ocultas, taxa de aprendizado, função de ativação e algoritmo de treinamento.

Ficou claro que SLPs, apesar de obterem boa performance (em termos de WER), acabaram se mostrando muito difíceis de se treinar, principalmente devido ao fato de precisarem de uma quantidade muito maior de neurônios para atingir os mesmos níveis atingidos por MLPs. Também ficou evidente que somente aumentar o número de camadas e neurônios ocultos não contribuiu para a melhora de performance das redes, visto que o WER e o tempo de treinamento aumentaram proporcionalmente ao número de camadas e neurônios ocultos adicionados. Isto se deve, principalmente, devido ao fato de que a camada de entrada da RNA utilizada neste trabalho, possuía apenas 732 neurônios, o que torna o valor da distribuição de pesos nas sinapses muito pequeno, impedindo a RNA de encontrar uma solução. Esta solução teria melhor resultado se a camada de entrada tivesse milhares de neurônios.

Contudo, este trabalho comprovou que a utilização de teoremas, como o de Hetchnielsen para determinação do número de neurônios e camadas ocultas, pode ser o passo inicial de uma série de tentativas empíricas que visam refinar a performance do ajuste de pesos realizados pelo algoritmo de retropropagação (*backpropagation*). Também foi identificado melhora significativa do WER, variando-se a taxa de aprendizado, principalmente para o algoritmo Incremental que é muito sensível a altas taxas de aprendizado. E por fim, a função de ativação que obteve o melhor WER médio entre os algoritmos de treinamento, foi a função Elliot, apesar de que a função senoidal tenha sido utilizada na topologia que alcançou a maior taxa de acerto deste projeto.

Cada vez mais surgem novas aplicações para comandos de voz, de acordo com (VERTO, 2017), o assistente virtual Siri, que funciona inteiramente baseado em comandos de voz, possui mais de 40 milhões de usuários mensais. Baseado nesta aplicação e nas diversas outras citadas no capítulo 1, é justificado o investimento em novas técnicas para tratamento e reconhecimento de voz. No capítulo 6 são listadas algumas sugestões e otimizações para a continuidade deste trabalho.

# 6 Oportunidades Futuras

Conforme discutido na seção 2.2, apesar de o estado da arte de sistemas RAV residir nas técnicas de identificação, o seu desempenho (WER e tempo de treinamento) pode ser significativamente melhorado através de otimizações na etapa de pré-processamento.

Na etapa de pré-processamento, onde foi utilizado a ferramenta HTK para extrair os coeficientes MFCC dos arquivos .WAV, não se mediu o tempo levado para executar esta operação, simplesmente por não ter-se utilizado outros métodos para a realização de um comparativo. Entretanto, o HTK suporta outras técnicas como o LPC (*Linear Predictive Coding*) que, de acordo com (JAVIER; KIM, 2014, p. 1831), poderia ser utilizado para reduzir o custo computacional da extração dos coeficientes, viabilizando o processamento em tempo real. Ainda na etapa de processamento dos dados, é possível utilizar técnicas para otimizar a quantidade de dados que serão utilizados na etapa de identificação, como o PCA (*Principal Component Analysis*). O PCA é um procedimento estatístico cujo principal objetivo é identificar a base mais significativa do conjunto de dados, na expectativa de filtrar o ruído e revelar a estrutura oculta (SHLENS, 2014, p. 2). Como oportunidades para trabalhos futuros, sugere-se a utilização de técnicas como o PCA, para diminuir a quantidade de coeficientes passada para a etapa de identificação, simplicando-a.

Neste trabalho, desenvolveu-se um algoritmo de força bruta, descrito em 3.2.3, capaz de variar o valor dos parâmetros da RNA baseado nas opções selecionadas pelo usuário. Conforme proposto em (STATHAKIS, 2009, p. 2135), é possível utilizar algoritmos genéticos para realizar buscas mais eficientes para selecionar os melhores parâmetros das RNAs. Algoritmos genéticos são muito eficientes na procura de soluções ótimas ou semi-ótimas, pois não impõem muitas restrições, diferente de métodos de busca tradicionais. A otimização do processo de busca resultaria numa redução drástica do tempo de treinamento e, consequentemente, um ganho de performance.

Ainda de acordo com (STATHAKIS, 2009), outra maneira de melhorar o desempenho das RNAs é utilizar técnicas de poda. Nas RNAs desenvolvidas neste trabalho, todas as camadas são totalmente conectadas umas com as outras, em outras palavras, cada neurônio está conectado a todos os neurônios da próxima camada. Técnicas de poda são utilizadas para otimizar o número de conexões, simplificando a rede e tornando o processo de treinamento mais eficiente (rápido). Técnicas de poda visam encontrar um número ótimo de conexões iterativamente, adicionando ou removendo ligações em uma rede que possui redundâncias ou inicialmente, nenhuma conexão.

Apesar dos resultados promissores obtidos neste trabalho, existe uma grande quantidade de técnicas alternativas sendo desenvolvidas no meio científico, como Florestas

Aleatórias (*Random Forest*), KNN (*K-Nearest Neighbour*) e SVM (*Support Vector Machine*). Um trabalho comparativo de performance entre estas técnicas com o proposto neste trabalho, agregaria muito à discussão de sistemas RAV.

# Referências

ATTNEAVE, R. K. O. F. *Pitch as a Medium: A New Approach to Psychophysical Scaling*. 1971. Disponível em: <http://www.jstor.org/stable/1421351>. Citado na página 31.

BATISTA, G. *Pré-Processamento de Dados em Aprendizado de Máquina Supervisionado*. Dissertação (Doutorado) — Universidade de São Paulo, Março 2003. Citado na página 37.

BURNETT, D. C.; FANTY, M. Rapid unsupervised adaptation to children's speech on a connected-digit task. In: *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. [S.l.: s.n.], 1996. v. 2, p. 1145–1148 vol.2. Citado na página 43.

CHEN, S.-H.; CHEN, W.-Y. Generalized minimal distortion segmentation for ann-based speech recognition. In: *In Proc. HLT-NAACL*. [S.l.: s.n.], 1995. p. 487–494. Citado na página 23.

CHIN-HUI, L.; SOONG, F.; PALIWAL, K. *Automatic Speech and Speaker Recognition: Advanced Topics*. 2012. Disponível em: <https://books.google.com.br/books?id=NCvjBwAAQBAJ&dq=voice+recognition+book&lr=&hl=pt-BR&source=gbs_navlinks_s>. Citado na página 23.

*Tracking the model: Data assimilation by artificial neural network*. Disponível em: <http://ieeexplore.ieee.org.ez130.periodicos.capes.gov.br/document/7727227/>. Citado na página 34.

CIPRIANO, J. L. G. *Desenvolvimento de Arquitetura Para Sistemas de Reconhecimento Automático de Voz Baseados em Modelos Ocultos de Markov*. 2001. Disponível em: <http://www.lume.ufrgs.br/handle/10183/2633?locale-attribute=pt_BR>. Citado 3 vezes nas páginas 28, 29 e 30.

COUVREUR, L. et al. Audio thumbnailing. In: *CQPSR of the numediart research program, Vol. 1*. [S.l.: s.n.], 2008. p. 67–85. Citado 2 vezes nas páginas 17 e 32.

DAVIS, P. M. S. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 28, n. 4, p. 357–366, 1980. Citado 2 vezes nas páginas 31 e 33.

DELLER JOHN PROAKIS, J. H. J. *Discrete-Time Processing of Speech Signals*. 2000. Citado na página 30.

DESAI, S. et al. Spectral mapping using artificial neural networks for voice conversion. *IEEE Transactions on Audio, Speech, and Language Processing*, v. 18, n. 5, p. 954–964, July 2010. ISSN 1558-7916. Citado na página 48.

ELENIUS, D.; BLOMBERG, M. Dynamic vocal tract length normalization in speech recognition. 2011. Citado na página 24.

FAHLMAN, S. E. An empirical study of learning speed in backpropagation networks. 1988. Citado na página 71.

FAN, L. et al. Power-normalized plp (pnplp) feature for robust speech recognition. In: *Chinese Spoken Language Processing (ISCSLP), 2012 8th International Symposium*. [S.l.: s.n.], 2013. p. 224–228. Citado na página 23.

FIORIN, D. V. et al. Aplicações de redes neurais e previsões de disponibilidade de recursos energéticos solares. *Revista Brasileira de Ensino de Física*, v. 1, n. 33, p. 1309, 2011. Disponível em: <https://www.researchgate.net/publication/235932975_Aplicacoes_de_redes_neurais_e_previsoes_de_disponibilidade_de_recursos_energeticos_solares>. Citado 2 vezes nas páginas 17 e 34.

GALES, S. Y. M. *The Application of Hidden Markov Models in Speech Recognition*. 2008. Citado na página 23.

GARDNER, M.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, v. 32, n. 14, p. 2627 – 2636, 1998. ISSN 1352-2310. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1352231097004470>. Citado na página 41.

GENCAY, R.; QI, M. Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE Transactions on Neural Networks*, v. 12, n. 4, p. 726–734, Jul 2001. ISSN 1045-9227. Citado 3 vezes nas páginas 17, 49 e 50.

GUPTA, H.; GUPTA, D. Lpc and lpcc method of feature extraction in speech recognition system. In: *Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference*. [S.l.: s.n.], 2016. p. 498–502. Citado na página 23.

HAYKIN, S. *Redes Neurais: Princícios e Práticas*. [S.l.]: Bookman, 2001. Citado 4 vezes nas páginas 17, 35, 36 e 41.

HECHT-NIELSEN, R. Kolmogorov's mapping neural net- works existence theorem. In: *First IEEE International Conference on Neural Networks*. [S.l.: s.n.], 1987. p. 11–14. Citado na página 52.

HECKER, M. H. L. *Speaker Recognition: An Interpretive Survey of the Literature*. 1971. Citado 2 vezes nas páginas 17 e 25.

HINTON, G. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, v. 29, n. 6, p. 82–97, Nov 2012. ISSN 1053-5888. Citado na página 23.

HTK. *What is HTK?* [S.l.], Acessado 24/10/2017. Disponível em: <http://htk.eng.cam.ac.uk/>. Citado na página 44.

HUANG, G.-B. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, v. 14, n. 2, p. 274–281, Mar 2003. ISSN 1045-9227. Citado na página 52.

IGEL, C.; HüSKEN, M. Improving the rprop learning algorithm. In: *Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000)*. [S.l.: s.n.], 2000. Citado na página 71.

JAVIER, R. J.; KIM, Y. Application of linear predictive coding for human activity classification based on micro-doppler signatures. *IEEE Geoscience and Remote Sensing Letters*, v. 11, n. 10, p. 1831–1834, Oct 2014. ISSN 1545-598X. Citado na página 63.

JUANG, L. R. R. B. *Automatic Speech Recognition – A Brief History of the Technology Development*. 2004. Citado 2 vezes nas páginas 27 e 31.

KOLMOGOROV, A. On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. In: *Dokl. Akad. Nauk SSSR*. [S.l.: s.n.], 1957. p. 179–182. Citado na página 51.

KURIMO, M. et al. Unlimited vocabulary speech recognition for agglutinative languages. In: *In Proc. HLT-NAACL*. [S.l.: s.n.], 2006. p. 487–494. Citado na página 23.

LATHI, B. *Sinais e Sistemas Lineares*. [S.l.]: Bookman, 2007. Citado na página 29.

LEE, L.; ROSE, R. A frequency warping approach to speaker normalization. *IEEE Transactions on Speech and Audio Processing*, v. 6, n. 1, p. 49–60, 1998. ISSN 1063-6676. Citado na página 43.

LEONARD, R. A database for speaker-independent digit recognition. In: *ICASSP '84. IEEE International Conference on Acoustics, Speech, and Signal Processing*. [S.l.: s.n.], 1984. v. 9, p. 328–331. Citado na página 43.

LIPPMANN, R. An introduction to computing with neural nets. *IEEE ASSP Magazine*, v. 4, n. 2, p. 4–22, Apr 1987. ISSN 0740-7467. Citado na página 38.

MARTINS, R.; YNOGUTI, C. Normalização do locutor em sistemas de reconhecimentode fala para usuários crianças. In: *XIII Simpósio Brasileiro Sobre Fatores Humanos em Sistemas Computacionais*. [S.l.: s.n.], 2014. Citado 4 vezes nas páginas 15, 17, 32 e 33.

MARTINS, R. M. *Análise comparativa entre os métodos HMM e GMM-UBM na busca pelo alpha-ótimo dos locutores crianças para utilização da técnica VTLN*. 2014. Citado 5 vezes nas páginas 24, 25, 43, 45 e 61.

MERTINS, A.; RADEMACHER, J. Vocal tract length invariant features for automatic speech recognition. In: *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.* [S.l.: s.n.], 2005. p. 308–312. Citado na página 43.

MESEGUER, N. *Speech Analysis for Automatic Speech Recognition*. Dissertação (Tese de Mestrado) — Norwegian University of Science and Technology, Julho 2009. Citado 2 vezes nas páginas 44 e 45.

NISSEN, S. Implementation of a fast artificial neural network library (fann). 12 2003. Citado 3 vezes nas páginas 49, 58 e 59.

NISSEN, S. *Algoritmos de Treinamento FANN*. 2017. <http://leenissen.dk/fann/fann_1_2_0/r1996.html>. [Online; acessado 10/11/2017]. Citado na página 71.

NISSEN, S. *Funções de Ativação FANN*. 2017. <http://leenissen.dk/fann/html/files/fann_data-h.html>. [Online; acessado 10/11/2017]. Citado na página 73.

PRAVEEN, A. et al. Speech recognition using arithmetic coding and mfcc for telugu language. In: *Computing for Sustainable Global Development (INDIACom)*. [S.l.: s.n.], 2016. p. 265–268. Citado na página 23.

QUEIROZ, A.; SANTANA, M. *Reconhecimento de vocábulos utilizando redes neurais*. Dissertação (Bachelor Thesis) — UniCEUB (Centro Universitário de Brasília), 2008. Citado na página 25.

REZENDE, S. O.; MONARD, M. C.; CARVALHO, A. C. P. L. Sistemas inteligentes para engenharia: Pesquisa e desenvolvimento. In: *Anais III Workshop de Sistemas Inteligentes para Engenharia*. Belo Horizonte: Editora UFMG, 1999. Citado na página 37.

RIEDMILLER, M.; BRAUN, H. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In: *IEEE International Conference on Neural Networks*. [S.l.: s.n.], 1993. p. 586–591 vol.1. Citado na página 71.

SHLENS, J. A tutorial on principal component analysis. 2014. Citado na página 63.

SOLERA-URENA, R. et al. Roo *perceptron* bust asr using support vector machines. v. 49, p. 253–267, 04 2007. Citado na página 23.

STATHAKIS, D. How many hidden layers and nodes? v. 30, p. 2133–2147, 04 2009. Citado 3 vezes nas páginas 51, 53 e 63.

SUKSRI, S. Speech recognition using mfcc. In: *International Conference on Computer Graphics, Simulation and Modeling (ICGSM 2012)*. [S.l.: s.n.], 2012. p. 135–138. Citado na página 31.

VERTO, A. *Rise of the Machines: How AI-Driven Personal Assistant Apps Are Shaping Digital Consumer Habits*. [S.l.], 2017. Citado na página 62.

YOUNG, S. *HMMs and Related Speech Recognition Technologies*. [S.l.]: Springer, 2008. 539-557 p. Citado na página 23.

YOUNG, S. et al. *The HTK book*. [S.l.: s.n.], 2002. Citado 2 vezes nas páginas 45 e 47.

*A Geometrical Representation of McCulloch–Pitts Neural Model and Its Applications*, v. 4. 925-929 p. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.459.7236&rep=rep1&type=pdf>. Citado na página 35.

ZHAO, X. et al. Application of support vector machine for pattern classification of active thermometry-based pipeline scour monitoring. v. 22, 12 2014. Citado 2 vezes nas páginas 17 e 38.

# APÊNDICE A – Configurações de *Software* e *Hardware*

| **Configuração** | |
|---|---|
| Versão FANN | 2.2.0 |
| Processador | Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz |
| Memória RAM | 16 GB |
| Sistema Operacional | Ubuntu 16.04 LTS (64 bits) |

Tabela 12 – Configurações

# APÊNDICE B – Algoritmos de Treinamento FANN

## Descrição

Este apêndice traz uma lista dos algoritmos de treinamento disponíveis para uso na biblioteca de *software* FANN (NISSEN, 2017a).

- **INCREMENTAL:** Algoritmo de retropropagação de erro padrão, onde os pesos são atualizados após cada padrão de treinamento. Isto significa que os pesos são atualizados diversas vezes durante uma época. Por esta razão, alguns problemas serão solucionados rapidamente com este algoritmo, enquanto outros mais complexos não serão treinados eficientemente.

- **LOTE:** Algoritmo de retropropagação de erro padrão, onde os pesos são atualizados após o cálculo do erro médio quadrático para todo o conjunto de treinamento. Isto significa que os pesos são atualizados somente uma vez por época. Por esta razão, este algoritmo demora a treinar. Entretanto, como o erro médio quadrático é calculado mais eficientemente do que no treinamento incremental, alguns problemas encontrarão melhores soluções com este algoritmo.

- **RPROP:** Um algoritmo de treinamento em lote mais avançado, que atinge bons resultados para vários problemas. O algoritmo de treinamento RPROP é adaptativo, e portanto, não utiliza a taxa de aprendizado. Contudo, alguns outros parâmetros podem alterar a maneira como este algoritmo trabalha.

  O algoritmo RPROP é descrito em (RIEDMILLER; BRAUN, 1993), mas o algoritmo de treinamento realmente implementado nesta biblioteca é o iRPROP descrito em (IGEL; HüSKEN, 2000), que é uma variação do algoritmo de treinamento RPROP padrão.

- **QUICKPROP:** Um algoritmo de treinamento em lote mais avançado, que atinge bons resultados para vários problemas. O algoritmo QUICKPROP utiliza a taxa de aprendizado em conjunto com outros parâmetros mais avançados, mas é somente recomendado alterar estes parâmetros, usuários que tenham conhecimento sobre o funcionamento deste algoritmo.

  O algoritmo de treinamento QUICKPROP é descrito em (FAHLMAN, 1988).

# APÊNDICE C – Funções de Ativação FANN

Este apêndice traz detalhes sobre a implementação das funções de ativação disponíveis na biblioteca de *software* FANN (NISSEN, 2017b).

## Parâmetros:

- $x$: entrada da função de ativação
- $y$: saída
- $s$: inclinação (*steepness*)
- $d$: derivação

## Funções:

- **Linear:** função de ativação linear.
  - Faixa: $-\inf < y < \inf$
  - $y = x * s, d = 1 * s$
  - Não pode ser utilizada com ponto fixo.
- **Sigmóide Simétrica:** função de ativação sigmóide simétrica, também conhecida como tangente hiperbólica.
  - Uma das funções de ativação mais utilizadas.
  - Faixa: $-1 < y < 1$
  - $y = tanh(s * x) = 2/(1 + exp(-2 * s * x)) - 1$
  - $d = s * (1 - (y * y))$
- **Sigmóide Simétrica Progressiva (*Stepwise*):** Aproximação progressiva da função sigmóide simétrica.
  - Mais rápida que a sigmóide simétrica, porém um pouco menos precisa.
- **Elliot Simétrica:** função de ativação rápida (parecida com a sigmóide simétrica) definida por David Elliott.
  - Faixa: $-1 < y < 1$

  - $y = (x * s)/(1 + |x * s|)$
  - $d = s * 1/((1 + |x * s|) * (1 + |x * s|))$

- **Linear Simétrica Limitada:** função de ativação linear simétrica limitada.
  - Faixa: $-1 < y < 1$
  - $y = x * s, d = 1 * s$

- **Senoidal Simétrica:** função de ativação senoidal periódica.
  - Faixa: $-1 < y < 1$
  - $y = sin(x * s)$
  - $d = s * cos(x * s)$

- **Cossenoidal Simétrica:** função de ativação cossenoidal periódica.
  - Faixa: $-1 < y < 1$
  - $y = cos(x * s)$
  - $d = s * -sin(x * s)$