Davyd Bandeira de Melo

Um Sistema de Reconhecimento de Comandos de Voz Utilizando a Rede Neural ELM

Fortaleza - CE, Brasil

Maio/2011

Davyd Bandeira de Melo

Um Sistema de Reconhecimento de Comandos de Voz Utilizando a Rede Neural ELM

Orientador:
Prof. Dr. Guilherme de Alencar Barreto

Departamento de Engenharia de Teleinformática
Centro de Tecnologia
Universidade Federal do Ceará

Fortaleza - CE, Brasil

Maio/2011

Monografia de Conclusão de Curso apresentada à Coordenação do Curso de Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de Engenheiro de Teleinformática.

Prof. Dr. Guilherme de Alencar Barreto
Orientador

Prof. Dr. Guilherme de Alencar Barreto
Universidade Federal do Ceará

Prof. Dr. Guilherme de Alencar Barreto
Universidade Federal do Ceará

Resumo

A interação homem-máquina vem a cada dia tomando novas dimensões e não mais se limitando apenas aos periféricos convencionais tais como telas sensíveis ao toque, teclados, mouses, dentre outros. Tem sido buscada uma maior naturalidade na comunicação entre os seres humanos e as entidades computacionais que os rodeiam. Esta mudança de paradigma de interação tem sido evidente em áreas como Robótica e sistemas multimídia de entretenimento, tais como consoles de vídeo game de última geração. Logo, faz necessário fazer um estudo que contribua diretamente para enriquecer mais ainda este cenário. Este trabalho consiste da aplicação de redes neurais artificias, particularmente Máquinas de Aprendizagem Extrema (ELM), no reconhecimento de comandos de voz. Além disso, realiza um estudo comparativo entre duas abordagens de classificação diferentes, uma linear, utilizando Memória Associativa Linear Ótima (OLAM) e outra não linear, utilizando ELM, para a solução do mesmo problema de classificação.

Abstract

The man-machine interaction comes every day taking on new dimensions and not limited only to conventional peripherals such as touch screens, keyboards, mice, among others. Has been pursued in a more natural communication between humans and computational entities that surround them. This change of interaction paradigm has been evident in areas such as robotics and multimedia entertainment systems, such as last generation video game consoles. Therefore, one must do a study that directly contribute to further enrich this scenario. This work consists of applying artificial neural networks, particularly Extreme Learning Machine (ELM), in recognition of voice commands. We also perform a comparative study between two different classification approaches, a linear, using Optimal Linear Associative Memory (OLAM) and other non-linear, using ELM for the solution of the same classification problem.

Dedicatória

Dedico este trabalho a minha família, por todo apoio dado a mim para que mantivesse sempre seguindo em frente com os estudos. Aos amigos fiéis que sempre estiveram presentes e me ajudaram a trilhar o árduo caminho. Ao meu orientador, que transmitiu a mim seus diversos conhecimentos e experiências.

Agradecimentos

Agradeço primeiramente a Deus por ter me dado sabedoria, força e serenidade para realizar as melhores escolhas e por estar presente sempre ao meu lado, em todos os momentos da minha vida. Depois, à minha família e namorada que sempre me apoiaram de maneira incondicional. Finalmente, agradeço a todos os professores, orientadores e amigos que puderam compartilhar comigo o longo caminho de graduação.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

SRCV Sistema de Reconhecimento de Comandos de Voz

ELM Extreme Learning Machine

OLAM Optimal Linear Associative Memory

ARMA AutoRegressive Moving Average

1 *Introdução*

1.1 Motivação

O Reconhecimento de Sinais de Voz é uma área amplamente rica em aplicações. Tais aplicações variam desde a identificação de doenças até a autenticação de pessoas. Algumas dessas aplicações vêm como auxílio a um operador humano, seja na tomadas de decisões ou na mobilidade adquirida por este ao utilizar uma interface de comandos de voz. Por exemplo, em Martins et al. (2009), o processamento de sinais de voz vem aliado ao reconhecimento de padrões para realizar a identificação automática de doenças da laringe através da fala de um paciente. Já em Rodrigues e Reis (2009), o reconhecimento de voz é aplicado na identificação automática da direção que um robô móvel deve assumir.

Este trabalho tem como intuito aplicar Máquinas de Aprendizagem Extrema no reconhecimento de comandos de voz e analisar quão bem esta rede neural se adequa ao problema. Para realizar uma avaliação de desempenho desta rede neural, um estudo comparativo com uma rede linear do tipo Memória Associativa Linear Ótima foi desenvolvido.

1.2 Objetivos

Os objetivos desta monografia, são apresentados a seguir.

- Desenvolver um sistema de reconhecimento de comandos de voz dependente de locutor.
- Aplicar Máquinas de Aprendizagem Extrema no reconhecimento de comandos de voz.
- Desenvolver um estudo comparativo entre um classificador linear (OLAM) e outro não linear (ELM).

1.3 Organização da Monografia

Esta monografia está organizada em 4 capítulos. No capítulo 2 serão apresentados os fundamentos do reconhecimento de voz e as ferramentas básicas de processamentos desses sinais. Esse capítulo dará suporte à extração das informações básicas utilizadas no reconhecimentos dos comandos de voz. No capítulo 3 serão reunidos alguns conceitos básicos de Reconhecimento de Padrões e onde as redes neurais estudadas estão inseridas. No capítulo 4 será apresentado o sistema desenvolvido, os parâmetros de configuração e execução adotados, os resultados obtidos utilizando este simulador. Por fim, o trabalho é concluído no capítulo 5.

2 *Fundamentos do Reconhecimento de Voz*

Neste capítulo são apresentadas as etapas básicas presentes nos sistemas de reconhecimento de voz. Basicamente, estes sistemas são divididos em três etapas: recorte ou deteção de extremos, extração de características e reconhecimento, como ilustrado na Figura 9.

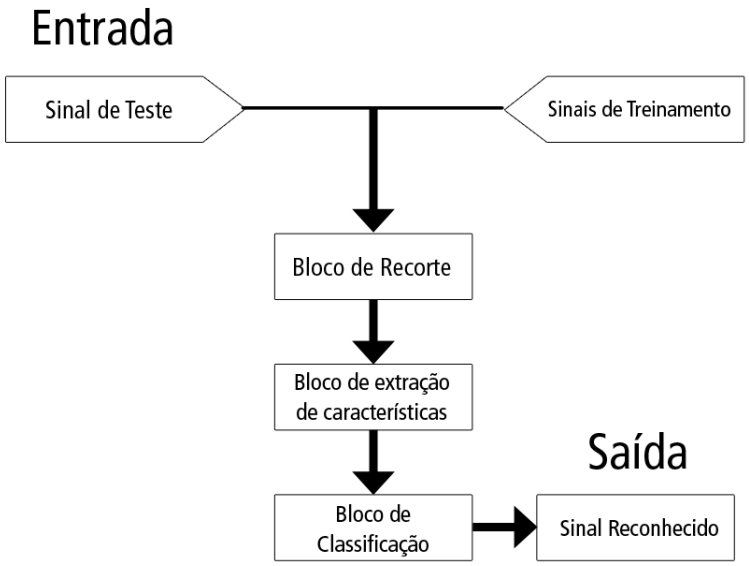


Figura 1: Estrutura básica dos sistemas de reconhecimento de voz.

A etapa de recorte é responsável por determinar o que é silêncio/ruído e o que de fato é informação de voz útil. Além disto, esta etapa reduz a quantidade de informação de voz que é passada para a fase de extração de características, diminuíndo assim o custo computacional do sistema.

No bloco de extração de características são usados algoritmos que atuam sobre o sinal de voz a fim de representá-lo de forma mais compacta. Isto acontece pois o sinal de voz

não é usado diretamente para alimentar o bloco de classificação por ser muito ruidoso, além de questões de custo de armazenamento, visto que um sinal de voz pode ter milhares de amostras.

No bloco de extração de características, o sinal de voz é representado de forma reduzida através da extração de atributos que permitam a diferenciao das elocuções.

A fase de reconhecimento pode ser dividida em treinamento e classificação. No treinamento, os vetores de características das elocuções são utilizados para determinar um modelo que represente cada classe de elocuções. A etapa de classificação usa o modelo gerado no treinamento para determinar qual elocução foi pronuncida. Em sistemas de reconhecimento independentes de locutor, as elocuções devem ser capturadas por um grande número de locutores, preferivelmente de diferentes idades, sotaques e gênero para que o sistema seja capaz de capturar variações entre locutores e então tornar-se mais abrangente.

Antes da descrição detalhada de cada etapa, é necessária uma visão geral sobre a modelagem da produção da fala.

2.1 Modelagem da produção da Fala

Existem duas principais fontes de características da fala específicas aos locutores, as físicas e as adquiridas (ou aprendidas). As características físicas relacionam-se principalmente ao trato vocal, estrutura formada pelas cavidades que vão das pregas vocais até os lábios e o nariz. Na Figura 2 é ilustrado o conjunto de órgãos que formam o trato vocal e compõem o sistema de produção da fala.

Durante a produção da fala o ar vindo dos pulmões passa pelas cordas vocais, situadas na laringe, que são tensionadas e então vibram devido à passagem do fluxo de ar. O som produzido pelas pregas vocais ainda é fraco e possui poucos harmônicos, de modo que ele é então amplificado quando passa pelas chamadas cavidades de ressonância (laringe, faringe, boca e nariz), e ganha "forma" final quando é articulado através de movimentos de língua, lábios, mandíbula, dentes e palato.

À medida que as ondas sonoras passam pelas cavidades do trato vocal, o espectro da voz é alterado pelas ressonâncias do trato vocal. Estas ressonâncias formam picos de energia no espectro de frequência conhecidos como formantes. Com isso, pode-se então estimar a forma do trato vocal a partir da análise espectral da fala produzida.

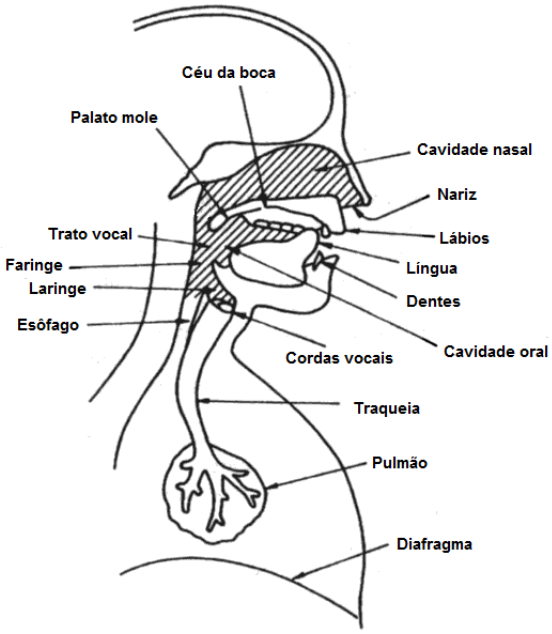
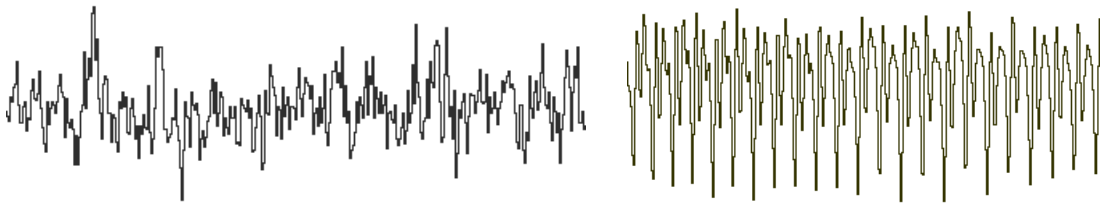


Figura 2: Sistema de produção da fala, adaptado de Henrique (2002).

Uma classificação comum dos eventos sonoros é quanto ao estado da vibração das cordas vocais. Adota-se uma convenção de três estados: silêncio, vozeados (sonoros) e não-vozeados (surdos). O silêncio representa a etapa em que nenhum som é produzido. Os sons ou fonemas sonoros são aqueles em que as cordas vocais são tensionadas e vibram de maneira aproximadamente periódica. Os sons surdos são produzidos quando não há vibração das cordas vocais, de modo que o som é formado basicamente nas cavidades do trato vocal, resultando em um sinal com natureza não-peridica ou aleatória. Na Figura 3 são ilustrados exemplos de sons vozeados e não-vozeados, em que pode-se observar na Figura 3(a) a natureza aleatória dos sons não-vozeados e, na Figura 3(b), a forma quasi-periódica de um fonema vozeado.



(a) elocução do fonema /s/: Som não-vozeado. (b) Fonema /z/: Som vozeado.

Figura 3: Exemplo de eventos sonoros vozeados e não-vozeados.

Existem ainda diversas outras caracterizações de eventos sonoros, como quanto ao modo e ponto de articulação, que fogem ao escopo deste trabalho, mas são apresentadas em (RABINER; JUANG, 1993) e (DELLER; HANSEN; PROAKIS, 2000).

O sistema de produção da fala pode ser modelado em analogia com filtros digitais. Neste contexto, a fala representada como resultante de um sinal de excitação, $u(n)$, convolvido com um filtro linear, $h(n)$. Esta excitação e representada por um "trem periódico" de impulsos no caso de trechos vozeados, e ruído branco nos trechos não-vozeados.

Conforme Deller, Hansen e Proakis (2000) e Rabiner (1978), durante um período de tempo no qual a voz pode ser considerada um sinal estacionário, o filtro linear pode ser caracterizado com boa aproximação por um modelo ARMA (*AutoRegressive Moving Average*), ou seja, um modelo constitudo de pólos e zeros, cuja função de transferência no domnio z é dada por

$$H(z) = G\frac{1 + \sum_{i=1}^{R} b(i)z^{-i}}{1 - \sum_{j=1}^{P} a(j)z^{-j}}, \tag{2.1}$$

em que $a(j)$ e $b(i)$ são os coeficientes de autoregressão e de médias móveis, respectivamente. Os valores P e R representam as ordens dos modelos auto-regressivo e média móvel, e G um fator de ganho associado à excitação $u(n)$.

2.2 Detecção de Extremos

A principal fonte de perturbações é o ruído do ambiente, causado por ar-condicionado, vozes de terceiros, sirenes e buzinas, entre outros. Perturbações tais como o sopro e o estalo da língua e dos lábios são também comuns aos sistemas de reconhecimento. O estalar da língua e dos lábios ocorre principalmente no início ou fim da elocução. O sopro consiste na respirão mais forte, e geralmente encontra-se ao término da elocução.

Detecção de extremos (*endpoints*) consiste em discriminar eventos sonoros de interesse, úteis ao sistema de reconhecimento, de perturbações do ambiente. Uma detecção errada pode degradar consideravelmente o desempenho do classificador (RABINER; JUANG, 1993).

Segmentação de palavras conectadas e detecção de extremos por si só abrangem toda uma área de pesquisa. Diversas técnicas têm sido utilizadas, entre elas pode-se citar as que utilizam energia e taxa de cruzamento por zero (RABINER; SAMBUR, 1975), (LAMEL et al., 1981), (TANYER; OZER, 2000); características espectrais (RAMREZ et al., 2004); filtros estatísticos sobre medidas de energia (RAMIREZ et al., 2005). Um outro método rápido

que se adequa para utilização em um sistema embarcado é o baseado em energia, descrito em Wang et al. (2008). Uma avaliação dos principais métodos de detecção de extremos pode ser encontrada em Tuononen, Hautamki e Franti (2008).

Neste trabalho, será utilizada uma abordagem baseada no trabalho de Rabiner e Sambur (1975), devido a sua simplicidade, relativo baixo custo computacional e fácil compreensão. Embora grande parte dos algoritmos de recorte sejam aplicados antes da etapa extração de características, existem abordagens que se utilizam desta etapa para identificar silêncio e voz, e estão presentes em trabalhos como o de Haigh e Mason (1993) e Martin, Charlet e Mauuary (2001).

A idéia subjacente aos algoritmos que utilizam energia e cruzamento por zero consiste na diferença de energia e frequência entre a fala e o período de silêncio. A energia por quadro de um sinal de voz é maior no instante dos fonemas pronunciados do que durante o silêncio. Assim, a energia é utilizada como primeira estimativa para início e fim da elocução. Em seguida, a taxa de cruzamento por zero é usada para estender os limiares da palavra, no caso em que fonemas com baixas energias, mas altas frequências ocorrem no início ou fim da palavra pronunciada. Antes de iniciar o estudo sobre o algoritmo utilizado neste trabalho, é necessário definir as medidas de energia e taxa de cruzamento por zero.

2.2.1 Energia e Taxa de Cruzamento por Zero

A energia de um sinal é um conceito amplamente utilizado em diversas áreas de Processamento de Sinais. Seja $\{s(n)\}_{n=1}^{N}$ um sinal de voz de tempo discreto, sua energia é dada por

$$E(m) = \sum_{n=m-L+1}^{m} s^2(n). \tag{2.2}$$

A taxa de cruzamento por zero, $ZC(m)$, de um segmento do sinal de voz $s(n)$ representa a quantidade de vezes que o sinal cruza a mediana na escala da amplitude, normalizada pelo tamanho do quadro, L, e pode ser representada matematicamente como

$$ZC(m) = \frac{1}{L} \sum_{n=m-L+1}^{m} \frac{|\operatorname{sgn}\{s(n)\} - \operatorname{sgn}\{s(n-1)\}|}{2}. \tag{2.3}$$

em que m representa um índice associado a um segmento, ou quadro, do sinal de voz, e

$$\text{sgn}\{s(n)\} = \begin{cases} +1, & s(n) > 0, \\ -1, & c.c. \end{cases} \tag{2.4}$$

Se a escala de amplitude do sinal estiver simetricamente distribuda em relação à amplitude nula, então a taxa de cruzamento por zero pode ser implementada em *software* simplesmente através da multiplicação de uma amostra pela sua anterior. Caso o resultado forneça um valor negativo, então o sinal cruzou a amplitude nula.

2.2.2 Algoritmo para Determinação de Extremos

O algoritmo de Rabiner e Sambur (1975) usa estatísticas de primeira e segunda ordem para determinação de extremos. Estas estatísticas são extraídas dos primeiros 100 a 200 ms do sinal, trecho em que se assume que o sinal consiste somente de ruído, e servem para calcular três limiares definidos a seguir.

- Limiar superior de energia : $\mu_e + A\sigma_e^2$
- Limiar inferior de energia : $\mu_e + B\sigma_e^2$
- Limiar da taxa de cruzamento: $\mu_c + C\sigma_c^2$

em que $A > B$, μ_e é a média da energia, σ_e^2 é a variância da energia, μ_c e σ_c^2 a média e a variância da taxa de cruzamentos por zero, respectivamente. Os valores de A, B e C são definidos heuristicamente e podem ser adaptados conforme a necessidade do sistema de recorte. importante que o valor de A seja maior que o de B, pois o limiar superior de energia é a estimativa mais "grosseira" do ponto de início e fim da palavra pronunciada. Na Figura 4 são ilustrados os passos relativos ao algoritmo de recorte.

O algoritmo pode ser dividido em duas etapas. A primeira etapa do algoritmo consiste na segmentação do sinal de voz, em janelas (sem superposição) com duração entre 10 e 20 ms. Para cada segmento m, calculam-se os valores de energia $E(m)$ e taxa de cruzamento por zero $ZC(m)$. A partir destes valores pertencentes aos 10 a 20 primeiros segmentos, são extraídas as estatísticas de média e variância que são utilizadas para calcular os limiares superior e inferior de energia, e da taxa de cruzamento por zero.

A segunda e mais importante etapa, subdivide-se em três iterações. Na primeira iteração percorre-se em direção ao centro do sinal a partir dos extremos em busca de segmentos com valores de energia maiores que o limiar superior de energia. Caso encontrado,

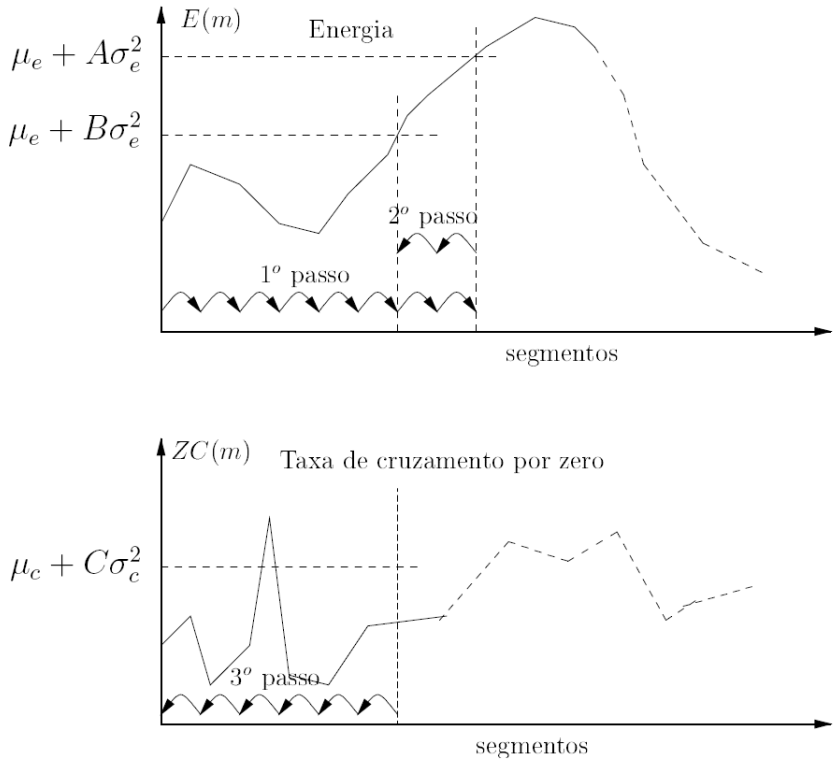


Figura 4: Passos executados no algoritmo de recorte, adaptado de Lima (2000).

o incio deste segmento é a primeira estimativa do incio ou fim da palavra. Na segunda iteração, percorre-se a partir dos pontos encontrados na iteração anterior aos extremos. Se for encontrado um valor de energia menor que o limiar inferior, é marcado esse ponto e inicia-se a próxima iteração. A terceira iteração busca por taxas de cruzamento nos 20 segmentos subsequentes (em direção aos extremos) com valores maiores que o limiar da taxa de cruzamento. Caso existam três ou mais segmentos que excedam este limiar, então marca-se como segmento de recorte o último que ultrapassou o limiar. Do contrário, os pontos de recorte são os valores encontrados após a segunda iteração.

2.3 Extração de Características

A etapa de extração de características consiste na utilização de técnicas de transformação do sinal original em uma representação matemática que permita a identificação de uma dada elocução. Desse modo, o sinal é geralmente representado por um conjunto de vetores de características que podem então ser utilizados com mais eficiência na etapa de reconhecimento.

Essa etapa deve proporcionar uma redução no espaço de dados para análise sem perda significativa de informação útil. Nas subseções a seguir serão mostradas as etapas

envolvidas na extração de características.

2.3.1 Pré-Processamento

Para o reconhecimento da fala, o primeiro passo é a conversão do sinal analógico em digital. A aquisição é feita através de um transdutor que, em geral, é um microfone. A conversão analógico-digital inclui tarefas de codificação e amostragem. Na codificação, utiliza-se, geralmente, 8 ou 16 bits para representar digitalmente uma amostra do sinal capturado. A amostragem é efetuada entre 8k - 44k Hz, satisfazendo o teorema de Nyquist (OPPENHEIM; WILLSKY; NAWAB, 1996). Em muitos sistemas é comum a aplicação de um filtro passa-baixas para limitar a banda de frequência do sinal. Com isto, pode-se eliminar o fenômeno conhecido como *aliasing*.

Observa-se que, para sinais de voz, a energia presente nas altas frequências é pequena quando comparada com as baixas. A pré-ênfase é então necessária para que sejam obtidas amplitudes mais homogêneas das frequências dos formantes, pois informações importantes sobre a elocução também podem estar presentes nas altas frequências. Isto pode ser feito através de um filtro digital FIR (*Finite Impulse Response*) de primeira ordem, cuja função de transferência no domínio z

$$G(z) = 1 - \alpha z^{-1}, \tag{2.5}$$

sendo α o parâmetro responsável pela pré-ênfase. Um valor bastante utilizado para α é 0,95 (MAFRA, 2002).

2.3.2 Análise de Curta Duração

O sinal de voz é não-estacionário e ruidoso, de modo que a analogia com filtros digitais estabelecida na Seção 2.1 só é válida para um período de tempo aproximadamente estacionário da fala, que geralmente é em torno de 10 a 30 ms. Dessa forma, todos os métodos de extração de características utilizados neste trabalho, utilizam a chamada análise de curta duração (*Short-Term Analysis*).

Para esse fim, define-se uma janela (10-30 ms) que é movida ao longo do sinal de voz, com ou sem superposicionamento entre quadros[1] adjacentes. Existem diversas formas de implementar o janelamento do sinal, uma das janelas mais comuns é a de Hamming

[1]Parte do sinal compreendida em uma janela.

(OPPENHEIM; WILLSKY; NAWAB, 1996), definida como

$$w(n) = 0,54 - 0,46\cos\Big(\frac{2n\pi}{N_w - 1}\Big), \tag{2.6}$$

em que N_w corresponde ao tamanho da janela. Então, a partir de cada quadro são extraídos os atributos do sinal de voz. Um esquema geral da análise de curta duração é ilustrado na Figura 5.

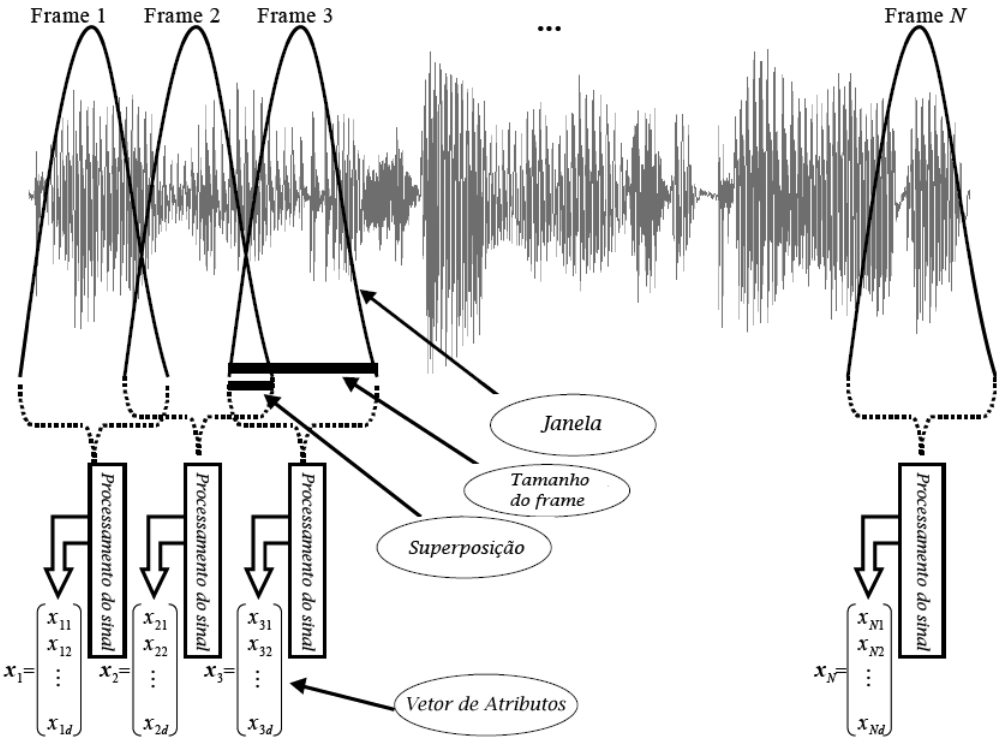


Figura 5: Visão geral da abordagem de análise de curta duração.

2.3.3 Codificação Linear Preditiva

Codificação Linear Preditiva (LPC, *Linear Predictive Coding*) é uma técnica de parametrização amplamente utilizada no reconhecimento de voz e locutor, pois permite uma representação do sinal por um número relativamente pequeno de parâmetros (KARPOV,2003).

Apesar da modelagem da fala ser representada inicialmente como um sistema ARMA, conforme discutido na Seção 2.1, as técnicas de extração de características em sistemas de RAV, utilizam, em geral, um modelo mais simples. Este, $H(z)$, corresponde a um

filtro AR (*AutoRegressive*), isto é, um filtro constituído somente de pólos (*all-pole*) e cuja função de transferência é dada por

$$H(z) = \frac{1}{1 - \sum_{i=1}^{P} a(i) z^{-i}}. \tag{2.7}$$

em que P é a ordem do preditor e $a(i)$ são os coeficientes de autoregressão.

A vantagem óbvia de utilizar modelos AR é a simplificação analítica do problema. Dessa forma, podem-se desenvolver métodos mais eficientes e simples para estimar os parâmetros do modelo. Além disto, na prática, zeros estão relacionadas à fase, que não tem efeito considerável na percepção. Se houver interesse somente em codificar e armazenar a fala, um modelo AR, que leva em conta somente o espectro de magnitude, não o de fase, é completamente aceitável e útil (DELLER; HANSEN; PROAKIS, 2000).

A função de transferência do modelo AR descrito na Equação (2.7) representa o sinal de voz, $s(n)$, através da combinação linear de seus P valores passados juntamente com a excitação $u(n)$:

$$s(n) = \sum_{i=1}^{P} a(i) s(n-i) + G u(n), \tag{2.8}$$

em que $s(n)$ é a saída do modelo, $a(i)$ são os coeficientes de predição linear e P é a ordem do preditor. Visto que nas aplicações de processamento de voz o termo $u(n)$, que representa a excitação glotal e entrada do sistema, é desconhecido, geralmente este é ignorado (CAMPBELL JR, 1997), (MAKHOUL, 1975). Vale ressaltar que o termo "desconhecido" significa que não se tem acesso ao valor de $u(n)$ pois tudo que se pode medir diretamente são as amostras do sinal da voz, embora estatísticas da excitação sejam conhecidas.

A análise LPC parte do pressuposto que o sinal de voz pode ser representado pela combinao linear das amostras atrasadas. Dessa forma, o novo valor estimado do sinal $s(n)$ dado por

$$\widehat{s}(n) = \sum_{i=1}^{P} \widehat{a}(i) s(n-i). \tag{2.9}$$

É importante lembrar que a modelagem acima é aplicada na análise de curta duração. Assim, o sinal de voz deve ser janelado, formando quadros, com ou sem superposio entre

eles. A partir disso, são computados coeficientes LPC para cada quadro do sinal de voz. Na prática, utilizam-se ordens de predição que variam, em geral, de 10 a 20 coeficientes. Mais detalhes sobre a escolha da ordem do preditor podem ser encontrados em Huang, Acero e Hon (2001).

Em suma, o problema básico na anlise LPC é determinar os coeficientes de predição, $\{\widehat{a}(i)\}_{i=1}^{P}$. Dentre os métodos mais usados para isso esto aqueles baseados no método clássico dos Mínimos Quadrados.

2.3.3.1 Estimação pelo Método de Yule-Walker

Uma das técnicas utilizadas para encontrar os parâmetros $\{a(i)\}$ do modelo, é conhecida como método de Yule-Walker (MORETTIN; TOLOI, 2004). Ele minimiza o erro quadrático médio (MSE - *Mean Squared Error*) entre o valor desejado, $s(n)$, e o valor estimado $\widehat{s}(n)$. A função custo $J(n)$ então definida como

$$J(n) = E[(s(n) - \widehat{s}(n))^2] = E[(s(n) - \sum_{i=1}^{P} \widehat{a}(i)s(n-i))^2]. \tag{2.10}$$

Sabendo que a função custo é uma forma quadrática, o ponto de mínimo global é dado pelo ponto em que o vetor gradiente da função é igual ao vetor nulo, ou seja,

$$\nabla \mathbf{J}(n) = \mathbf{0}. \tag{2.11}$$

Resolvendo a Equação (2.11) obtém-se o valor ótimo segundo o critério MSE, escrito na forma matricial:

$$\widehat{\mathbf{a}} = \mathbf{R}_s^{-1}\mathbf{r}_s, \tag{2.12}$$

em que

$$\mathbf{r}_s = \begin{pmatrix} r_s(1) \\ r_s(2) \\ \vdots \\ r_s(P) \end{pmatrix}, \quad \widehat{\mathbf{a}} = \begin{pmatrix} \widehat{a}(1) \\ \widehat{a}(2) \\ \vdots \\ \widehat{a}(P) \end{pmatrix}, \quad \mathbf{R}_s = \begin{pmatrix} r_s(0) & r_s(1) & \cdots & r_s(P-1) \\ r_s(1) & r_s(0) & \cdots & r_s(P-2)) \\ \vdots & \vdots & \vdots & \vdots \\ r_s(P-1) & r_s(P-2) & \cdots & r_s(0) \end{pmatrix}, \tag{2.13}$$

e $r_s(i) = E[s(n)s(n-i)]$ é a autocorrelação do sinal da voz com atraso i. A Equação (2.12) representa as equações de Yule-Walker na forma matricial.

O problema da inversão da matriz $\mathbf{R}_s$ é simplificado devido a matriz de autocorrelação ser Toeplitz[2]. O algoritmo recursivo de Durbin-Levinson é uma das formas mais eficientes de resolver esse tipo de sistema de equações (MAKHOUL, 1975). Uma descrição detalhada do algoritmo pode ser encontrada em Haykin (1991).

2.4 Abordagens para Reconhecimento de Voz

Existem diversas abordagens e algoritmos para reconhecimento de voz tais como Quantização Vetorial, Alinhamento Temporal Dinâmico e Redes Neurais Artificiais. Nesta seção, será apresentada uma descrição mais detalhada da abordagem utilizada neste trabalho.

2.4.1 Redes Neurais Supervisionadas

Uma forma muito comum de reconhecimento de voz consiste na utilização de redes neurais com aprendizagem supervisionada. Dentre elas estão a MLP (*MultiLayer Perceptron*), a OLAM (*Optimal Linear Associative Memory*) e a ELM (*Extreme Learning Machine*). Uma discussão em detalhes dos algoritmos foge do escopo deste trabalho, podendo ser encontrada em PRINCIPE, EULIANO e LEFEBVRE (2000) e HAYKIN (1994).

O principal problema na utilização de redes neurais supervisionadas está no algoritmo de normalização da elocução, necessário para fixar o número de atributos que serão apresentados ao classificador. Este problema torna-se mais evidente quando se refere ao reconhecimento de palavras pronunciadas espontâneamente. Ou seja, uma mesma palavra pronunciada por pessoas diferentes pode variar consideravelmente sua duração.

Para contornar esse problema, os vetores de características apresentados às redes neurais não são concatenados de forma a obter-se um vetor de dados maior, porém são apresentados com o mesmo formato após extraídos de cada quadro de uma elocução. A Seção 4.1.5 explanará como os dados são apresentados às redes neurais.

[2]Matriz em que cada diagonal descendente da esquerda para a direita possui o mesmo valor.

3 *Redes Neurais e Reconhecimento de Padrões*

Neste capítulo são abordados os conceitos básicos sobre sistemas de reconhecimento de padrões, sua estrutura básica, sua importância e suas aplicações, e como as redes neurais são empregadas em tais sistemas. Além disso, serão definidos todos os passos necessários para projetar um sistema de classificação de padrões. Estes passos foram seguidos em sua maior parte na implementação deste trabalho.

3.1 Sistemas de Reconhecimento de Padrões

Segundo Bishop (2006), o campo de reconhecimento de padrões se ocupa com a descoberta automática de regularidades em conjuntos de dados através do uso de algoritmos computacionais e com o uso dessas regularidades para tomar ações como classificar dados em categorias. Segundo Chen, Pau e Wang (2005) um sistema de reconhecimento de padrões possui a seguinte arquitetura, que é ilustrada pela imagem abaixo:

Cada bloco desta arquitetura é altemente dependente um do outro. Esta dependência normalmente é utilizada de forma a melhorar o desempenho final do sistema, ou seja, cada fase pode se ajustar dinamicamente de acordo com os resultados apresentados pelas outros fases (THEODORIDIS; KOUTROUMBAS, 2008).

O principal objetivo de um sistema de classificação é a generalização de um dado conjunto de dados. Através dela, o sistema pode inferir a qual classe pertence um objeto do mundo real. Vale ressaltar que esse objeto normalmente é extraído de um conjunto finito de elementos que é definido pelo problema em questão.

O primeiro estágio de um sistema de classificação de padrões é aquele que compreende as fontes de dados que alimentam o sistema. Tais fontes de dados podem variar desde um banco de dados relacional até um conjunto de sensores anexados ao sistema de classificação de padrões. Questões como a presença de ruído na aquisição de dados, precisão dos dados

adquiridos e calibração são levadas em conta nesta fase.

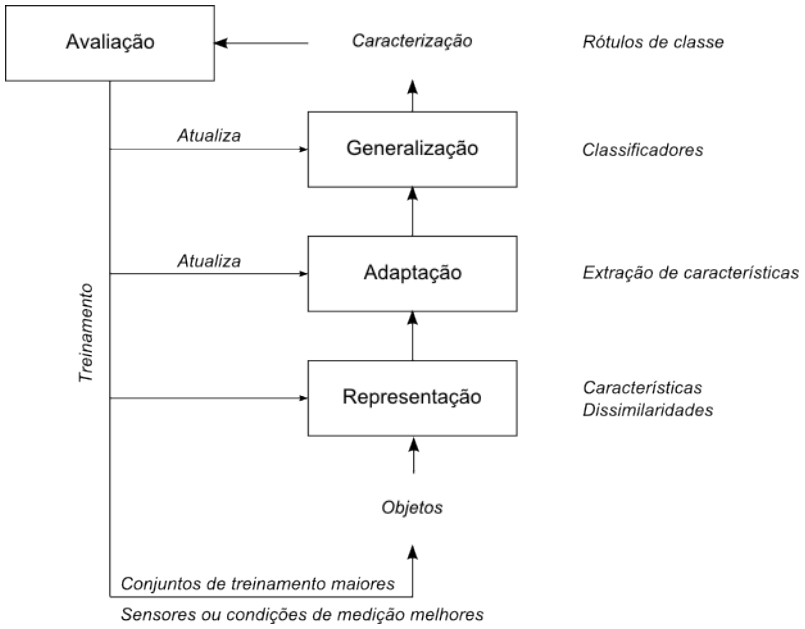


Figura 6: Arquitetura de Sistemas de Reconhecimento de Padrões.

Como pode ser observado na Figura 6, um classificador geralmente não recebe os dados crus dos objetos do mundo real. Esses dados normalmente são processados de forma a obter um vetor de p características que representem devidamente um objeto em um dado problema e que proporcionem uma boa capacidade de separação das classes. Este processo é chamado de extração de características ou fase de representação. Por exemplo, em um sistema de reconhecimento de comandos de voz, as diversas elocuções inseridas no sistema podem passar por uma fase de representação em que são extraídos diversos coeficientes de predição linear que representam o envelope espectral do sinal. A formação dos vetores de características reduz significantemente a quantidade de dados que são processados nos classificadores, melhorando assim o desempenho do sistema.

Após a extração das características, os vetores que representam os diversos objetos do mundo real são submetidos a um subsistema adaptador. Esse subsistema é responsável por gerar um conjunto menor de características, de dimensão q, que são mais relevantes ao problema de classificação. Essa adaptação pode ser feita de duas maneiras: seleção ou extração (CHEN; PAU; WANG, 2005). Na seleção, o novo conjunto de q características é um subconjunto do conjunto de original de características, não sofrendo qualquer tipo de processamento. Já na extração, novas características são derivadas das características originais através de algum processamento. Para exemplificar a importância desta fase, suponha que duas características concentram uma boa quantidade de informação de

classificação, ou seja, ajudam melhor a distinguir um padrão dos outros, e ambas forem altamente correlacionadas, talvez seja desnecessária a presença de uma delas no vetor de características, pois não haveria um ganho em relação ao processo de classificação (THEODORIDIS; KOUTROUMBAS, 2008).

O próximo subsistema na arquitetura é chamado de classificador. Para o projeto deste bloco, deve ser escolhido um critério de otimização para que o sistema classificador ajuste de forma eficiente uma curva que faz a separação das diversas classes no espaço de características. Esta separação não necessariamente constitui uma reta, porém pode gerar curvas altamente não lineares de acordo com o projeto do classificador. A próxima seção é dedicada à definição de duas classes de problemas: linearmente separáveis e linearmente não-separáveis.

É no contexto do projeto do classificador que estão as redes neurais artificiais, que são o foco de estudo desse trabalho. Foram estudadas e aplicadas duas abordagens, uma linear (OLAM) e outra não linear (ELM) para que pudesse ser estabelecido um estudo comparativo entre elas. Tais redes neurais passam, logo após a fase de adaptação das caracterísiticas, por um processo de treinamento que faz com que elas aprendam a separar as classes do problema. Para dar início ao treinamento, é apresentado para o classificador um conjunto de treinamento, denominado $\mathbf{X_{tr}}$, de N vetores. Cada vetor $\mathbf{x}$, pertence a uma classe y_i conhecida previamente. Cada classe y_i pertence ao conjunto das classes do problema, denominado w e definido como $w = w_0, w_1, ..., w_{K-1}$, onde K é o número de classes do problema. É através do fornecimento de exemplos presentes em $\mathbf{X_{tr}}$ que o classificador aprende a generalizar o problema em questão.

Tendo o classificador passado pela fase de treinamento, alguns métodos podem ser utilizados para medir o quão eficiente está o processo de generalização. Em Chen, Pau e Wang (2005) são apresentadas algumas maneiras de se realizar a avaliação de desempenho do sistema de classificação de padrões. As informações de saída deste estágio, assim como as dos anteriores, podem ser realimentadas no projeto do sistema para que alguns ajustes sejam feitos, melhorando assim o desempenho geral do sistema.

3.2 Tipos de Problemas de Interesse

Os problemas de classificação de padrões podem ser classificados quanto aos tipo de curva que separam suas classes: linearmente separáveis e linearmente não-separáveis.

Um problema linearmente separável permite o uso de classificadores mais simples,

uma vez que a decisão é tomada baseada no valor da combinação linear das informações de entrada $\mathbf{x} = (x_1...x_p)^T$, ou seja

$$g_j(\mathbf{x}) = \mathbf{w_j x} + w_{j0} = \sum_i w_{ij} + w_{j0} \tag{3.1}$$

em que $\mathbf{w} \in R^p$ é um vetor de parâmetros do modelo.

A superfície de decisão gerada pela função discriminante mostrada na Equação 1.5 que, para $\mathbf{x} \in R^2$, torna-se uma reta. Um exemplo desse tipo de superfície de decisão é mostrado na Figura 1.6

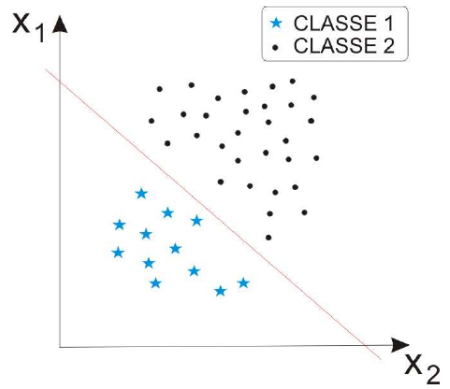


Figura 7: Problema linearmente separável.

Já para um problema não-linearmente separável, um hiperplano não consegue separar totalmente os padrões de classes diferentes, como mostrado na Figura 1.7.

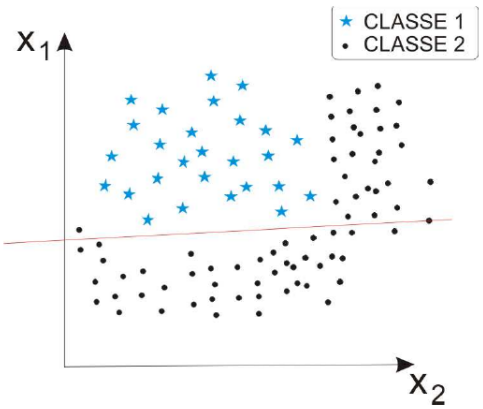


Figura 8: Problema linearmente não-separável no qual um classificador linear foi aplicado.

No entanto, se os padrões $\mathbf{x} \in$ S em que S $\subset R^n$ sofrerem uma transformação $\phi(x) = x_i$, com $x_i \in R$ e $\phi(\mathbf{x})$ uma função não-linear definida como

$$\phi(x) = f(\mathbf{w}^{\mathbf{T}}\mathbf{x} + w_0) \tag{3.2}$$

então as classes podem ser separadas linearmente no espaço de ϕ (WEBB, 2002).

Com base nisso, a função discriminante não-linear pode ser representada pela seguinte expressão

$$g_j(\mathbf{x}) = \sum_{i=1}^{q} m_{ij}\phi_i(x) + m_{j0} \tag{3.3}$$

Na equação acima, m_{ji} são os parâmetros do classificador, q representa a quantidade defunções $\phi(i)$ e c a quantidade de classes do problema. Assim, a regra do discriminante é dada por

$$\mathbf{x} \text{ pertence à classe k se } g_k(\mathbf{x}) = max_j g_j(\mathbf{x}) \tag{3.4}$$

em que x pertence a classe com o maior valor da função discriminante. Dessa forma, superfícies de decisão não-lineares podem ser formadas, como a da Figura 1.8.

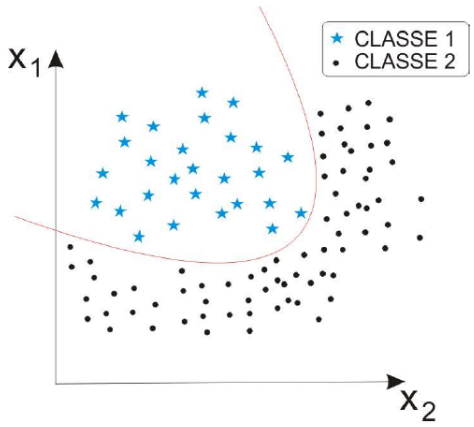


Figura 9: Problema linearmente não-separável no qual um classificador não-linear foi aplicado.

Em Nehmzow & McGonigle (1994), os autores afirmam que não encontraram aplicações no mundo real que necessitassem da aquisição de mapeamentos entrada-saída que são não linearmente separáveis. Em seus trabalhos, fazem uso de um processo de aprendizagem supervisionada externo, no qual os pesquisadores forneciam reforço positivo ou negativo por meio dos sensores de luz do robô, além de um detector de novidades.

O questionamento levantado no começo desta seção motiva a aplicação de diferentes

tipos de classificadores, com o objetivo de descobrir se um discriminante linear é o bastante para separar corretamente as classes do problema.

Redes neurais como a OLAM e a ELM são normalmente adequadas para problemas linearmente separáveis e não-linearmente separáveis respectivamente, e por isso serão utilizadas como um meio de descobrir em qual classe de problema está a classificação de comandos de voz. A próxima seção descreverá mais profundamente o bloco de classificação, tratando das especificidades das duas redes neurais empregadas neste trabalho: Memória Associativa Linear Ótima (OLAM - Optimal Linear Associative Memory) e Máquinas de Aprendizado Extremo (ELM – Extreme Learning Machine) .

3.3 Memória Associativa Linear Ótima

3.3.1 Definições Preliminares

Inicialmente, vamos assumir que existe uma lei matemática linear, também chamada de transformação ou mapeamento linear, que relaciona um vetor de entrada qualquer, $\mathbf{x} \in \mathbb{R}^{p+1}$, com um vetor de saída, $\mathbf{d} \in \mathbb{R}^m$. Esta relação pode ser representada matematicamente da seguinte forma:

$$\mathbf{d} = \mathbf{W}\mathbf{x} \tag{3.5}$$

em que se assume que a matriz $\mathbf{W} = [w_{ij}]$ é uma matriz de dimensão $m \times (p+1)$, totalmente desconhecida, ou seja, não sabemos de antemão quais são os valores de seus elementos w_{ij}, $i = 1, \ldots, m$, $j = 0, 1, \ldots, p$.

Lembre-se que a formulação matricial mostrada na Eq. (3.5) nada mais é do que uma representação compacta do seguinte sistema de equações lineares:

$$\begin{aligned}
d_1 &= w_{10}x_0 + w_{11}x_1 + w_{12}x_2 + \cdots + w_{1p}x_p \\
d_2 &= w_{20}x_0 + w_{21}x_1 + w_{22}x_2 + \cdots + w_{2p}x_p \\
d_3 &= w_{30}x_0 + w_{31}x_1 + w_{32}x_2 + \cdots + w_{3p}x_p \\
\vdots &= \quad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
d_m &= w_{m0}x_0 + w_{m1}x_1 + w_{m2}x_2 + \cdots + w_{mp}x_p
\end{aligned} \tag{3.6}$$

Uma linha específica do sistema de equações lineares da Eq. (3.6) pode ser represen-

tada vetorialmente como

$$d_i = \mathbf{w}_i^T \mathbf{x}, \tag{3.7}$$

em que $\mathbf{w}_i \in \mathbb{R}^{p+1}$ é o vetor de coeficientes (ou pesos) associados i-sima saída do modelo. Note que a Eq. (3.7) é equivalente ao modelo ADALINE (neurônio linear), do ponto de vista matemático.

Supondo que a única fonte de informação que nós temos a respeito sobre a transformação linear é apresentada na Equação (3.5) um conjunto finito de N pares entrada-saída observados, ou seja:

$$\begin{aligned} \mathbf{x}_1, &\quad \mathbf{d}_1 \\ \mathbf{x}_2, &\quad \mathbf{d}_2 \\ \vdots &\quad \vdots \\ \mathbf{x}_N, &\quad \mathbf{d}_N \end{aligned} \tag{3.8}$$

Os pares entrada-saída mostrados acima podem ser representados de maneira simplificada como $\{\mathbf{x}_\mu, \mathbf{d}_\mu\}$, em que μ é apenas um índice simbolizando o μ-ésimo par do conjunto de dados. Uma maneira de se adquirir informação sobre a matriz de transformação $\mathbf{W}$ se dá exatamente através dos uso destes pares.

Para isso, pode-se utilizar técnicas de manipulação de matrizes e vetores, oriundas da disciplina de álgebra linear, para implementar um mapeamento linear aproximado, representado matematicamente como

$$\mathbf{y}_\mu = \widehat{\mathbf{W}} \mathbf{x}_\mu \tag{3.9}$$

em que $\widehat{W}$ é uma matriz de transformação que produz um vetor de saída $\mathbf{y}_\mu$ que, espera-se, seja muito próximo da saída real $\mathbf{d}_\mu$. O vetor de erros entre $\mathbf{d}_\mu$ e $\mathbf{y}_\mu$ para o par entrada-saída $\{\mathbf{x}_\mu, \mathbf{d}_\mu\}$ é definido como:

$$\mathbf{e}_\mu = \mathbf{d}_\mu - \mathbf{y}_\mu. \tag{3.10}$$

O grau de similaridade entre $\mathbf{d}_\mu$ e $\mathbf{y}_\mu$ pode ser medido através do acúmulo dos quadrados da norma do vetor de erros para todo o conjunto de dados:

$$SEQ = \sum_{\mu=1}^{N} \|\mathbf{e}_\mu\|^2 = \sum_{\mu=1}^{N} \|\mathbf{d}_\mu - \mathbf{y}_\mu\|^2 = \sum_{\mu=1}^{N} (\mathbf{d}_\mu - \mathbf{y}_\mu)^T (\mathbf{d}_\mu - \mathbf{y}_\mu), \tag{3.11}$$

em que $\|\cdot\|$ denota a norma euclidiana.

A seguir será apresentada uma técnica de estimação de parâmetros para determinar a matriz $\widehat{\mathbf{W}}$ que produza o menor valor para SEQ. Na área de redes neurais, este procedimento leva ao modelo conhecido como Memória Associativa Linear Ótima (*Optimal Linear Associative Memory*, OLAM). Na área afim de Identificação de Sistemas e na Estatística, tal procedimento de estimação de parâmetros é conhecido como Método dos Mínimos Quadrados (MQ), ou ainda como Método da Pseudoinversa (PI).

3.3.2 Memória Associativa Linear Ótima

O modelo de OLAM foi proposto por vários autores de maneira independente, sendo que aquela a ser descrita nesta seção é devido a Kohonen (KOHONEN, 1989). A formulação matemática que utilizaremos na descrição do OLAM não é baseada em argumentos de otimização da função-custo definida pela SEQ, mas sim em argumentos de Álgebra Linear; ou seja, nas dimensões das matrizes e dos vetores envolvidos, de modo a garantir a inversão de uma certa matriz.

Primeramente, vamos separar aleatoriamente os pares entrada-saída em dois conjuntos, um de treinamento e outro de teste. O conjunto de teste contém $N_1 < N$ pares entrada-saída, enquanto o conjunto de teste contém $N_2 = N - N_1$ pares entrada-saída. Por exemplo, 80% dos pares entrada-saída poderiam ser selecionados aleatoriamente para compor o conjunto de treinamento. Os restantes 20% dos pares entrada-saída comporiam automaticamente o conjunto de teste.

Em seguida, precisamos definir duas matrizes, $\tilde{\mathbf{X}}$ e $\tilde{\mathbf{D}}$. As colunas da matriz $\tilde{\mathbf{X}}$ são formadas pelos vetores de entrada $\{\mathbf{x}_\mu\}_{\mu=1}^{N_1}$ selecionados para treinamento. Já as colunas da matriz $\tilde{\mathbf{D}}$ são formadas pelos vetores de saída $\{\mathbf{d}_\mu\}_{\mu=1}^{N_1}$ correspondentes. Deste modo, as matrizes $\tilde{\mathbf{D}}$ e $\tilde{\mathbf{X}}$ podem ser representadas como

$$\tilde{\mathbf{X}} = [\mathbf{x}_1 \mid \mathbf{x}_2 \mid \cdots \mid \mathbf{x}_{N_1}] \quad \text{e} \quad \tilde{\mathbf{D}} = [\mathbf{d}_1 \mid \mathbf{d}_2 \mid \cdots \mid \mathbf{d}_{N_1}]. \tag{3.12}$$

Podemos notar que as dimensões das matrizes $\tilde{\mathbf{X}}$ e $\tilde{\mathbf{D}}$ são as seguintes:

$$\dim(\tilde{\mathbf{X}}) = (p+1) \times N_1 \quad \text{e} \quad \dim(\tilde{\mathbf{D}}) = m \times N_1. \tag{3.13}$$

Vamos usar as matrizes $\tilde{\mathbf{X}}$ e $\tilde{\mathbf{D}}$ para construir o seguinte mapeamento linear

$$\tilde{\mathbf{D}} = \widehat{\mathbf{W}}\tilde{\mathbf{X}}, \tag{3.14}$$

que é a versão matricial do mapamento mostrado na Equação (3.9), em que todas as saídas $\mathbf{d}_\mu$, $\mu = 1, \ldots, N_1$ que formam as colunas de $\tilde{\mathbf{D}}$ são calculadas de uma só vez.

É importante ressaltar que estamos interessados em calcular a matriz $\widehat{\mathbf{W}}$ que melhor satisfaça a relação matemática mostrada na Equação (3.9). A dimensão de $\widehat{\mathbf{W}}$ é $\dim(\widehat{\mathbf{W}}) = m \times (p+1)$. Para isso, como as matrizes $\tilde{\mathbf{X}}$ e $\tilde{\mathbf{D}}$ são conhecidas, poderíamos num primeiro momento pensar em tentar isolar $\widehat{\mathbf{W}}$ na Equação (3.14) invertendo a matriz $\tilde{\mathbf{X}}$, porém isto não é permitido pois esta matriz não é quadrada.

Um procedimento alternativo para se isolar $\widehat{\mathbf{W}}$ consiste primeiro em multiplicar pela direita ambos os lados da Equação (3.14) pela transposta de $\tilde{\mathbf{X}}$, ou seja

$$\tilde{\mathbf{D}}\tilde{\mathbf{X}}^T = \widehat{\mathbf{W}}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T, \tag{3.15}$$

em que se percebe que a matriz $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$ é quadrada, ou seja, sua dimensão é $\dim(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T) = (p+1) \times (p+1)$.

Assim, agora possível isolar $\widehat{\mathbf{W}}$ multiplicando pela direita ambos os lados da Equação (3.15) pela inversa de $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$, ou seja

$$\tilde{\mathbf{D}}\tilde{\mathbf{X}}^T \left(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\right)^{-1} = \widehat{\mathbf{W}}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T \left(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\right)^{-1}, \tag{3.16}$$

de onde obtemos

$$\widehat{\mathbf{W}} = \tilde{\mathbf{D}}\tilde{\mathbf{X}}^T \left(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\right)^{-1}. \tag{3.17}$$

Uma vez que a matriz $\widehat{\mathbf{W}}$ tenha sido calculada, pode-se testar o desempenho do método OLAM para o conjunto de N_2 pares entrada-saída restantes. Isto pode ser feito individualmente ou em bloco.

Para o caso em que isto feito individualmente, podemos calcular o vetor de saída gerado pelo mapeamento para um único vetor de entrada como

$$\mathbf{y}_\mu = \widehat{\mathbf{W}}\mathbf{x}_\mu, \tag{3.18}$$

enquanto para o caso em que isto é feito em bloco, os vetores de saída gerados para todos os vetores de entrada disponíveis são calculados de uma única vez da seguinte forma:

$$\tilde{\mathbf{Y}} = \widehat{\mathbf{W}}\tilde{\mathbf{X}}. \tag{3.19}$$

O algoritmo OLAM pode ser usado tanto em aplicações de aproximação de funções, quanto em aplicações de classificação de padrões. Neste último caso é prática comum representar o vetor de saídas desejadas $\mathbf{d}_\mu$ como um vetor que tem apenas uma componente com valor igual a 1, as outras possuem valor -1. A saída desejada com valor $+1$ representa a classe do vetor de entrada correspondente.

Assim, a dimensão do vetor $\mathbf{d}_\mu$ sempre é igual ao número de classes do problema. Por exemplo, para um problema de classificação com três classes, teríamos $\dim(\mathbf{d}_\mu) = 3$, sendo que os respectivos vetores de saídas desejadas, representando os rótulos numéricos das classes, seriam representados por

$$\text{Classe } 1 \Rightarrow \mathbf{d}_\mu = \begin{bmatrix} +1 \\ -1 \\ -1 \end{bmatrix}, \quad \text{Classe } 2 \Rightarrow \mathbf{d}_\mu = \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix} \quad \text{e Classe } 3 \Rightarrow \mathbf{d}_\mu = \begin{bmatrix} -1 \\ -1 \\ +1 \end{bmatrix} \tag{3.20}$$

Durante o teste, a classe $\hat{\omega}$ atribuída a um novo vetor de entrada $\mathbf{x}_\mu$ édeterminada pela seguinte regra de decisão:

$$\hat{\omega} = \omega_{i^*_\mu}, \quad \text{tal que} \quad i^*_\mu = \arg\max_{\forall i}\{y^i_\mu\}, \tag{3.21}$$

em que i^*_μ o índice da componente de maior valor do vetor $\mathbf{y}_\mu$ e y^i_μ é a i-ésima sada do vetor $\mathbf{y}_\mu$, calculado como na Equação (3.18).

Quando o algoritmo OLAM é usado para construir classificadores dessa maneira, o classificador resultante, doravante chamado de classificador OLAM, pode ser entendido como uma versão *batch* do classificador Adaline.

3.4 Máquinas de Aprendizado Extremo

3.4.1 Definições Preliminares

De início, vamos assumir que existe uma lei matemática $\mathbf{F}(\cdot)$, também chamada aqui de função ou mapeamento, que relaciona um vetor de entrada qualquer, $\mathbf{x} \in \mathbb{R}^{p+1}$, com um vetor de saída, $\mathbf{d} \in \mathbb{R}^m$. Esta relação, é representada genericamente na Figura 10, pode ser descrita matematicamente da seguinte forma:

$$\mathbf{d} = \mathbf{F}[\mathbf{x}] \tag{3.22}$$

em que se assume que $\mathbf{F}(\cdot)$ é totalmente desconhecida, ou seja, não sabemos de antemão quais são as *fórmulas* usadas para associar um vetor de entrada $\mathbf{x}$ com seu vetor de saída $\mathbf{d}$ correspondente.

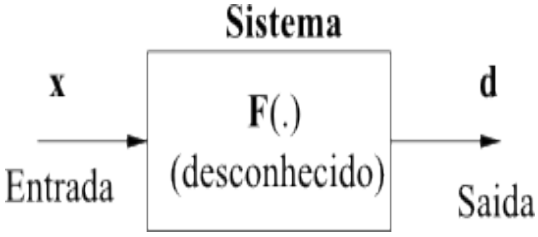


Figura 10: Representao simplificada de um mapeamento entrada-sada genrico.

O mapeamento $\mathbf{F}(\cdot)$ pode ser tão simples quanto um mapeamento linear, tal como

$$\mathbf{d} = \mathbf{M}\mathbf{x} \tag{3.23}$$

em que $\mathbf{M}$ é uma matriz de dimensão $(p+1) \times m$. Contudo, $\mathbf{F}(\cdot)$ pode ser bastante complexo, envolvendo relações não-lineares entre as variáveis de entrada e saída. É justamente o funcionamento da relação matemática $\mathbf{F}(\cdot)$ que se deseja *imitar* através do uso de algoritmos adaptativos, tais como as redes neurais.

Supondo que a única fonte de informação que nós temos a respeito de $\mathbf{F}(\cdot)$ é o conjunto finito de N pares entrada-saída observados (ou medidos), ou seja:

$$\begin{aligned} &\mathbf{x}_1, \quad \mathbf{d}_1 \\ &\mathbf{x}_2, \quad \mathbf{d}_2 \\ &\vdots \quad \vdots \\ &\mathbf{x}_N, \quad \mathbf{d}_N \end{aligned} \tag{3.24}$$

Os pares entrada-saída mostrados acima podem ser representados de maneira simplificada como $\{\mathbf{x}_\mu, \mathbf{d}_\mu\}$, em que μ é apenas um índice simbolizando o μ-ésimo par do conjunto de dados. Uma maneira de se adquirir conhecimento sobre $\mathbf{F}(\cdot)$ se dá exatamente através dos uso destes pares.

Para isto pode-se utilizar uma rede neural qualquer para implementar um mapeamento entrada-saída aproximado, representado como $\hat{\mathbf{F}}(\cdot)$, tal que:

$$\mathbf{y}_\mu = \hat{\mathbf{F}}[\mathbf{x}_\mu] \tag{3.25}$$

em que $\mathbf{y}_\mu$ é a saída gerada pela rede neural em resposta ao vetor de entrada $\mathbf{x}_\mu$. Esta saída, espera-se, seja muito próxima da saída real $\mathbf{d}_\mu$. Dá-se o nome de *Aprendizado Indutivo* ao processo de obtenção da relação matemática geral $\hat{\mathbf{F}}(\cdot)$ a partir de apenas alguns pares $\{\mathbf{x}_\mu, \mathbf{d}_\mu\}$ disponíveis.

A seguir serão mostrados duas arquiteturas de redes neurais com o propósito de obter uma representação aproximada de um mapeamento entrada-saída genérico.

3.4.2 Máquina de Aprendizado Extremo

Estamos considerando nas definições e cálculos a seguir uma arquitetura de rede neural do tipo *feedforward* (i.e. sem realimentação) com apenas uma camada de neurônios ocultos, conhecida como Máquina de Aprendizado Extremo (*Extreme Learning Machine*, ELM) (HUANG; ZHU; SIEW, 2006). Esta arquitetura de rede neural é semelhante à rede MLP, porém apresenta uma fase de aprendizado infinitamente mais rápida que a da rede MLP. Começaremos a seguir com a descrição de sua arquitetura, para em seguida escrever sobre o funcionamento e o treinamento da rede ELM.

Os neurônios da camada oculta (primeira camada de pesos sinápticos) são representados conforme mostrado na Figura 11a, enquanto os neurônios da camada de saída (segunda camada de pesos sinápticos) são representados conforme mostrado na Figura 11b.

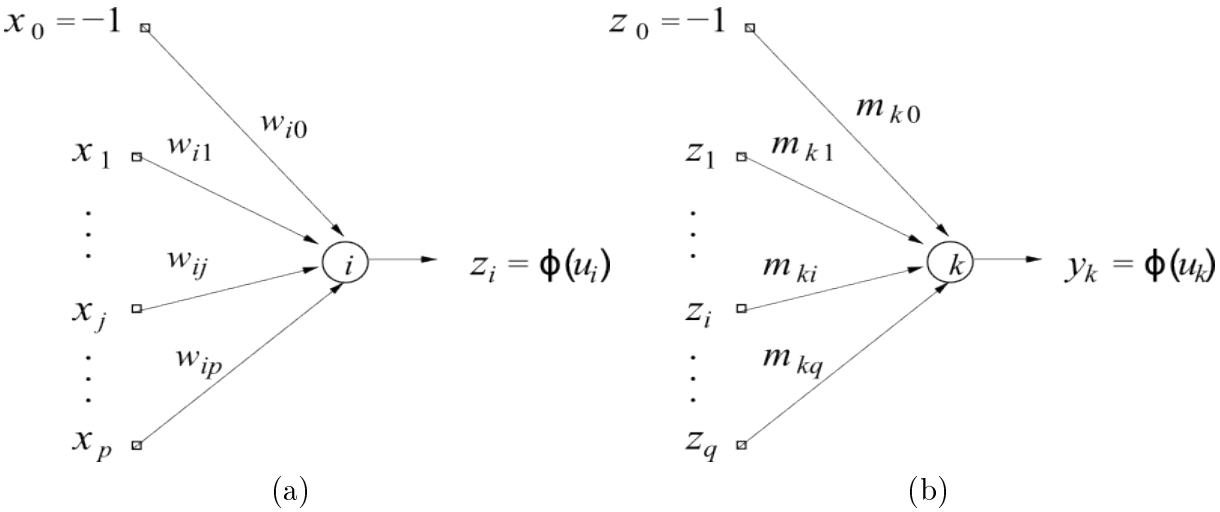


Figura 11: (a) Neurônio da camada escondida. (b) Neurônio da camada de saída.

O vetor de pesos associado a cada neurônio i da camada escondida, também chamada

de *camada oculta* ou *camada intermediária*, é representado como

$$\mathbf{w}_i = \begin{pmatrix} w_{i0} \\ \vdots \\ w_{ip} \end{pmatrix} = \begin{pmatrix} \theta_i \\ \vdots \\ w_{ip} \end{pmatrix} \tag{3.26}$$

em que θ_i é o limiar (*bias* ou *threshold*) associado ao neurônio i. Os neurônios desta camada são chamados de neurônios escondidos por não terem acesso direto saída da rede, onde são calculados os erros de aproximação.

De modo semelhante, o vetor de pesos associado a cada neurônio k da camada de saída representado como

$$\mathbf{m}_k = \begin{pmatrix} m_{k0} \\ \vdots \\ m_{kq} \end{pmatrix} = \begin{pmatrix} \theta_k \\ \vdots \\ m_{kq} \end{pmatrix} \tag{3.27}$$

em que θ_k é o limiar associado ao neurônio de saída k.

O treinamento da rede ELM se dá em duas etapas, que são descritas a seguir.

3.5 Fase 1: Inicialização Aleatória dos Pesos dos Neurônios Ocultos

Esta etapa de funcionamento da rede ELM envolve o cálculo das ativações e saídas de todos os neurônios da camada escondida e de todos os neurônios da camada de saída, uma vez que os pesos w_{ij}, $i = 1, \ldots, q$ e $j = 0, \ldots, p$, tenham sido inicializados com valores aleatórios. Formalmente, podemos escrever:

$$w_{ij} \sim U(a, b) \qquad \text{ou} \qquad w_{ij} \sim N(0, \sigma^2) \tag{3.28}$$

em que $U(a, b)$ é um número (pseudo-)aleatório uniformemente distribuído no intervalo (a, b), enquanto $N(0, \sigma^2)$ é um número (pseudo-)aleatório normalmente distribuído com média zero e variância σ^2.

Em ambientes de programação tais como Matlab© e Octave, esta fase é facilmente implementada em uma linha apenas de código. Para isso, precisamos definir uma matriz

de pesos $\mathbf{W}$, com q linhas e $p+1$ colunas:

$$\mathbf{W} = \begin{pmatrix} w_{10} & w_{11} & \cdots & w_{1p} \\ w_{20} & w_{21} & \cdots & w_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ w_{q0} & w_{q1} & \cdots & w_{qp} \end{pmatrix}_{q\times(p+1)} = \begin{pmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_q^T \end{pmatrix} \tag{3.29}$$

em que notamos que a i-ésima linha da matrix $\mathbf{W}$ é composta pelo vetor de pesos do i-ésimo neurônio oculto. Uma vez definida a matriz $\mathbf{W}$, podemos realizar a etapa 1 através das seguintes linhas de código Octave, caso os pesos sejam inicializados com números aleatórios uniformes:

```
» a=0; b=0.1;     % define intervalo dos pesos
» W=a+(b-a).*rand(q,p+1);     % gera numeros uniformes
```

ou pelas seguintes linhas se preferirmos números aleatórios gaussianos:

```
» sig=0.1;     % define desvio-padrao dos pesos
» W=sig*randn(q,p+1);     % gera numeros gaussianos
```

3.6 Fase 2: Acúmulo das Saídas dos Neurônios Ocultos

Esta é a fase do treinamento da rede neural. O fluxo de sinais (informação) se dá dos neurônios de entrada para os neurônios de saída, passando obviamente pelos neurônios da camada escondida. Por isso, diz-se que o informação está fluindo no sentido **direto** (*forward*), ou seja:

Entrada $\rightarrow$ Camada Intermediária $\rightarrow$ Camada de Saída

Assim, após a apresentação de um vetor de entrada $\mathbf{x}$, na iteração t, o primeiro passo é calcular as ativações dos neurônios da camada escondida:

$$u_i(t) = \sum_{j=0}^{p} w_{ij}x_j(t) = \mathbf{w}_i^T\mathbf{x}(t), \quad i = 1, \ldots, q \tag{3.30}$$

em que T indica o vetor (ou matriz) transposto e q indica o número de neurônios da camada escondida.

A operação sequencial da Equação. (3.30) pode ser feita de uma única vez se utilizarmos a notação vetor-matriz. Esta notação é particularmente útil em ambientes do tipo Matlab/Octave. Neste caso, temos que o vetor de ativações $\mathbf{u}_i(t) \in \mathbb{R}^q$ do i-ésimo neurônio oculto na iteração t é calculado como

$$\mathbf{u}(t) = \mathbf{W}\mathbf{x}(t). \tag{3.31}$$

Em seguida, as saídas correspondentes são calculadas como

$$z_i(t) = \phi_i(u_i(t)) = \phi_i\left(\sum_{j=0}^{p} w_{ij}(t)x_j(t)\right) = \phi_i\left(\mathbf{w}_i^T(t)\mathbf{x}(t)\right) \tag{3.32}$$

tal que a função de ativação ϕ assume geralmente uma das seguintes formas:

$$\phi_i(u_i(t)) = \frac{1}{1+\exp[-u_i(t)]}, \qquad \text{(Logstica)} \tag{3.33}$$

$$\phi_i(u_i(t)) = \frac{1-\exp[-u_i(t)]}{1+\exp[-u_i(t)]}, \qquad \text{(Tangente Hiperblica)} \tag{3.34}$$

Em notação matriz-vetor, a Equação. (3.32) pode ser escrita como

$$\mathbf{z}(t) = \phi_i(\mathbf{u}_i(t)) = \phi_i(\mathbf{W}\mathbf{x}(t)). \tag{3.35}$$

em que a função de ativação $\phi_i(\cdot)$ é aplicada a cada um dos q componentes do vetor $\mathbf{u}(t)$. Para cada vetor de entrada $\mathbf{x}(t)$, $t = 1, \ldots, N$, tem-se um vetor $\mathbf{z}(t)$ correspondente, que deve ser organizado (disposto) como uma coluna de uma matriz $\mathbf{Z}$. Esta matriz terá q linhas por N colunas:

$$\mathbf{Z} = [\mathbf{z}(1) \mid \mathbf{z}(2) \mid \cdots \mid \mathbf{z}(N)]. \tag{3.36}$$

A matriz $\mathbf{Z}$ será usada na Fase 3 para calcular os valores do pesos dos neurônios de saída da rede ELM.

3.7 Fase 3: Cálculo dos Pesos dos Neurônios de Saída

Sabemos que para cada vetor de entrada $\mathbf{x}(t)$, $t = 1, \ldots, N$, tem-se um vetor de saídas desejadas $\mathbf{d}(t)$ correspondente. Se organizamos estes N vetores ao longo das colunas de

uma matriz $\mathbf{D}$, então temos que esta matriz terá dimensão m linhas e N colunas:

$$\mathbf{D} = [\mathbf{d}(1) \mid \mathbf{d}(2) \mid \cdots \mid \mathbf{d}(N)]. \tag{3.37}$$

Podemos entender o cálculo dos pesos da camada de saída como o cálculo dos parâmetros de um mapeamento linear entre a camada oculta e a camada de saída. O papel de vetor de "entrada" para a camada de saída na iteração t é desempenhado pelo vetor $\mathbf{z}(t)$ enquanto o vetor de "saída" é representado pelo vetor $\mathbf{d}(t)$. Assim, buscamos determinar a matriz $\mathbf{M}$ que melhor represente a transformação

$$\mathbf{d}(t) = \mathbf{M}\mathbf{z}(t). \tag{3.38}$$

Para isso, podemos usar o método dos mínimos quadrados, também conhecido como método da pseudoinversa, já discutido anteriormente no classificador OLAM. Assim, usando as matrizes $\mathbf{Z}$ e $\mathbf{D}$, a matriz de pesos $\mathbf{M}$ é calculada por meio das seguinte expressão:

$$\mathbf{M} = \mathbf{D}\mathbf{Z}^T \left(\mathbf{Z}\mathbf{Z}^T\right)^{-1}. \tag{3.39}$$

Alguns comentários sobre a matriz $\mathbf{M}$ fazem-se necessários:

- Note que para satisfazer a Equação (3.38) a matriz $\mathbf{M}$ tem dimensão $m \times q$.
- A k-ésima linha da matriz $\mathbf{M}$, denotada aqui por $\mathbf{m}_k$, $k = 1, 2, \ldots, m$, corresponde ao vetor de pesos do k-ésimo neurônio de saída.

3.8 Teste e Capacidade de Generalização da Rede ELM

Uma vez determinadas as matrizes de pesos $\mathbf{W}$ e $\mathbf{M}$ temos a rede ELM pronta para uso. Durante o uso (teste) da rede ELM, calculamos as ativações dos neurônios da camada de saída por meio da seguinte expressão:

$$a_k(t) = \sum_{i=0}^{q} m_{ki}(t) z_i(t) = \mathbf{m}_k^T \mathbf{z}(t), \quad k = 1, \ldots, m \tag{3.40}$$

em que m é o número de neurônios de saída. Note que as saídas dos neurônios da camada oculta, $z_i(t)$, fazem o papel de entrada para os neurônios da camada de saída.

Em notação vetor-matriz, as operações da Equação (3.40) podem ser executadas de

uma só vez por meio da seguinte expressão:

$$\mathbf{a}(t) = \mathbf{M}\mathbf{z}(t). \tag{3.41}$$

Para a rede ELM, assumimos que os neurônios de saída usam a função identidade como função de ativação, ou seja, as saídas destes neurônios são iguais às suas ativações: calculadas como:

$$y_k(t) = \phi_k(a_k(t)) = a_k(t). \tag{3.42}$$

Por generalização adequada entende-se a habilidade da rede em utilizar o conhecimento armazenado nos seus pesos e limiares para gerar saídas coerentes para novos vetores de entrada, ou seja, vetores que não foram utilizados durante o treinamento. A generalização considerada boa quando a rede, durante o treinamento, foi capaz de capturar (aprender) adequadamente a relação entrada-saída do mapeamento de interesse.

O bom treinamento de uma rede ELM, de modo que a mesma seja capaz de lidar com novos vetores de entrada, depende de uma série de fatores, dentre os quais podemos listar os seguintes

1. Excesso de graus de liberdade de uma rede ELM, na forma de elevado número de parâmetros ajustáveis (pesos e limiares).

2. Excesso de parâmetros de treinamento, tais como taxa de aprendizagem, fator de momento, número de camadas ocultas, critério de parada, dimensão da entrada, dimensão da saída, método de treinamento, separação dos conjuntos de treinamento e teste na proporção adequada, critério de validação, dentre outros.

Em particular, no que tange ao número de parâmetros ajustáveis, uma das principais consequências de um treinamento inadequado é a ocorrência de um subdimensionamento ou sobredimensionamento da rede ELM, o que pode levar, respectivamente, ocorrência de *underfitting* (subajustamento) ou *overfitting* (sobreajustamento) da rede aos dados de treinamento. Em ambos os casos, a capacidade de generalizao é ruím.

Dito de maneira simples, o subajuste da rede aos dados ocorre quando a rede no tem poder computacional (i.e. neurônios na camada oculta) suficiente para aprender o mapeamento de interesse. No outro extremo está o sobreajuste, que ocorre quando a rede tem neurônios ocultos demais (dispostos em uma ou duas camadas ocultas) e passa a memorizar os dados de treinamento. O ajuste ideal é obtido para um número de camadas

ocultas e neurônios nestas camadas que confere à rede um bom desempenho durante a fase de teste, quando sua generalização é avaliada.

3.9 Dicas para um Bom Desempenho da Rede ELM

O projeto de uma rede neural envolve a especificação de diversos itens, cujos valores influenciam consideravelmente o funcionamento do algoritmo. A seguir especificaremos a lista destes itens juntamente com as faixas de valores que os mesmos podem assumir:

Dimensão do vetor de Entrada (p): Este item pode assumir em tese valores entre 1 e ∞. Porém, existe um limite superior que depende da aplicação de interesse e do custo de se medir (observar) as variáveis x_j. É importante ter em mente que um valor alto para p não indica necessariamente um melhor desempenho para a rede neural, pois pode haver redundância no processo de medição. Neste caso, uma certa medida é, na verdade, a combinação linear de outras medidas, podendo ser descartada sem prejuízo ao desempenho da rede. Quando muito caro, ou até impossível, medir um elevado número de variáveis x_j, deve-se escolher aquelas que o especialista da área considera como mais relevante ou representativas para o problema. O ideal seria que cada varivel x_j, $j = 1, \ldots, p$, "carregasse" informação que somente ela contivesse. Do ponto de vista estatístico, isto equivale a dizer que as variáveis são *independentes* ou *não-correlacionadas* entre si.

Dimensão do vetor de saída (M): Assim como o primeiro item, este também depende da aplicação. Se o interesse está em problemas de aproximação de funções, $\mathbf{y} = F(\mathbf{x})$, o número de neurônios deve refletir diretamente a quantidades de funções de saída desejadas (ou seja, a dimensão de $\mathbf{y}$).

Se o interesse está em problemas de classificação de padrões, a coisa muda um pouco de figura. Neste caso, o número de neurônios deve codificar o número de classes desejadas. É importante perceber que estamos chamando as classes às quais pertencem os vetores de dados de uma forma bastante genérica: classe 1, classe 2, ..., etc. Contudo, cada classe pode estar associado um rótulo (e.g. classe dos empregados, classe dos desempregados, classe dos trabalhadores informais, etc.), cujo significado depende da interpretação que o especialista na aplicação dá a cada uma delas. Estes rótulos normalmente no estão na forma numérica, de modo que para serem utilizados para treinar a rede ELM eles devem ser convertidos para a forma numérica. A este procedimento dá-se o nome de codificação da saída da rede.

A codificação mais comum define como vetor de saídas desejadas um vetor binário de comprimento unitário; ou seja, apenas uma componente deste vetor terá o valor "1", enquanto as outras terão o valor "0" (ou -1). A dimensão do vetor de saídas desejadas corresponde ao número de classes do problema em questão. Usando esta codificação define-se automaticamente um neurônio de saída para cada classe. Por exemplo, se existem três classes possíveis, existirão três neurnios de saída, cada um representando uma classe. Como um vetor de entrada não pode pertencer a mais de uma classe ao mesmo tempo, o vetor de saídas desejadas terá valor 1 (um) na componente correspondente classe deste vetor, e 0 (ou -1) para as outras componentes. Por exemplo, se o vetor de entrada $\mathbf{x}(t)$ pertence classe 1, então seu vetor de saídas desejadas $\mathbf{d}(t) = [1 \;\; 0 \;\; 0]^T$. Se o vetor $\mathbf{x}(t)$ pertence classe 2, então seu vetor de saídas desejadas $\mathbf{d}(t) = [0 \;\; 1 \;\; 0]^T$ e assim por diante para cada exemplo de treinamento.

número de neurônios na camada escondida (q): Encontrar o número ideal de neurônios da camada escondida não é uma tarefa fácil porque depende de uma série de fatores, muito dos quais não temos controle total. Entre os fatores mais importantes podemos destacar os seguintes:

1. Quantidade de dados disponíveis para treinar e testar a rede.
2. Qualidade dos dados disponíveis (ruidosos, com elementos faltantes, etc.)
3. Número de parâmetros ajustáveis (pesos e limiares) da rede.
4. Nível de complexidade do problema (não-linear, discontínuo, etc.).

O valor de q geralmente encontrado por tentativa-e-erro, em função da capacidade de *generalização* da rede (ver definição logo abaixo). Grosso modo, esta propriedade avalia o desempenho da rede neural ante situações não-previstas, ou seja, que resposta ela dá quando novos dados de entrada forem apresentados. Se muitos neurônios existirem na camada escondida, o desempenho será muito bom para os dados de treinamento, mas tende a ser ruim para os novos dados. Se existirem poucos neurônios, o desempenho será ruim também para os dados de treinamento. O valor ideal aquele que permite atingir as especificações de desempenho adequadas tanto para os dados de treinamento, quanto para os novos dados.

Existem algumas fórmulas heurísticas (*ad hoc*) que sugerem valores para o número de neurônios na camada escondida da rede ELM, porém estas regras devem ser usadas apenas para dar um valor inicial para q. O projetista deve sempre treinar

e testar várias vezes uma dada rede ELM para diferentes valores de q, a fim de se certificar que a rede neural generaliza bem para dados novos, ou seja, não usados durante a fase de treinamento.

Dentre a regras heurísticas citamos a seguir três, que são comumente encontradas na literatura especializada:

Regra do valor médio - De acordo com esta fórmula o número de neurônios da camada escondida igual ao valor médio do número de entradas e o número de saídas da rede, ou seja:

$$q = \frac{p + M}{2} \tag{3.43}$$

Regra da raiz quadrada - De acordo com esta fórmula o número de neurônios da camada escondida igual a raiz quadrada do produto do número de entradas pelo número de saídas da rede, ou seja:

$$q = \sqrt{p \cdot M} \tag{3.44}$$

Regra de Kolmogorov - De acordo com esta fórmula o número de neurônios da camada escondida é igual a duas vezes o número de entradas da rede adicionado de 1, ou seja:

$$q = 2p + 1 \tag{3.45}$$

Perceba que as regras só levam em consideração características da rede em si, como número de entradas e número de sadas, desprezando informações úteis, tais como número de dados disponíveis para treinar/testar a rede e o erro de generalização máximo aceitável.

Uma regra que define um valor inferior para q levando em consideração o número de dados de treinamento/teste dada por:

$$q \geq \frac{N - 1}{p + 2} \tag{3.46}$$

A regra geral que se deve sempre ter em mente é a seguinte: *devemos sempre ter muito mais dados que parâmetros ajustáveis.* Assim, se o número total de parâmetros (pesos + limiares) da rede dado por $Z = (p + 1) \cdot q + (q + 1) \cdot M$, então devemos

sempre tentar obedecer à seguinte relação:

$$N \gg Z \tag{3.47}$$

Um refinamento da Equação (3.47), é proposto por Baum & Haussler (1991), sugere que a relação entre o número total de parâmetros da rede (Z) e a quantidade de dados disponíveis (N) deve obedecer seguinte relação:

$$N > \frac{Z}{\varepsilon} \tag{3.48}$$

em que $\varepsilon > 0$ é o erro percentual máximo aceitável durante o teste da rede; ou seja, se o erro aceitável é 10%, então $\varepsilon =$ é 0,1. Para o desenvolvimento desta equação, os autores assumem que o erro percentual durante o treinamento não dever ser maior que $\varepsilon/2$.

Para exemplificar, assumindo que $\varepsilon =$ é 0,1, então temos que $N > 10Z$. Isto significa que para uma rede de Z parâmetros ajustáveis, devemos ter uma quantidade dez vezes maior de padrões de treinamento.

Note que se substituirmos Z na Equação (3.48) e isolarmos para q, chegaremos à seguinte expressão que fornece o valor aproximado do número de neurônios na camada oculta:

$$q \approx \left\lceil \frac{\varepsilon N - M}{p + M + 1} \right\rceil \tag{3.49}$$

em que $\lceil u \rceil$ denota o menor inteiro maior que u.

A Equação (3.49) é bastante completa, visto que leva em consideração não só aspectos estruturais da rede ELM (número de entradas e de saídas), mas também o erro máximo tolerado para teste e o número de dados disponíveis. Portanto, seu uso é bastante recomendado.

Funções de ativação (ϕ_i) e (ϕ_k): - Em tese, cada neurônio pode ter a sua própria função de ativação, diferente de todos os outros neurônios. Contudo, para simplificar o projeto da rede é comum adotar a mesma para todos os neurônios. Em geral, escolhe-se a função logística ou a tangente hiperbólica para os neurônios da camada escondida. Aquela que for escolhida para estes neurônios será adotada também para os neurônios da camada de saída. Em algumas aplicações é comum adotar uma função de ativação linear para os neurônios da camada de saída, ou seja, $\phi_k(u_k(t)) = C_k \cdot u_k(t)$, onde C_k é uma constante (ganho) positiva. Neste caso, tem-se que $\phi'_k(u_k(t)) = C_k$. O fato de $\phi_k(u_k(t))$ ser linear não altera o poder computacio-

nal da rede, o que devemos lembrar sempre que os neurônios da camada escondida devem ter uma função de ativação não-linear, obrigatoriamente.

Avaliao de Desempenho: - O desempenho da rede ELM é, em geral, avaliada com base nos valores do erro quadrático médio (ε_{teste}) por padrão de teste:

$$\varepsilon_{teste} = \frac{1}{N}\sum_{t=1}^{N}\varepsilon(t) = \frac{1}{2N}\sum_{t=1}^{N}\sum_{k=1}^{n}e_k^2(t) \tag{3.50}$$

em que $e_k(t) = d_k(t) - y_k(t)$ é o erro do k-ésimo neurônio de saída na iteração t.

Por outro lado, quando se utiliza a rede para classificar padrões, o desempenho da mesma é avaliado pela *taxa de acerto na classificação*, definida como:

$$P_{acerto} = \frac{\text{número de vetores classificados corretamente}}{\text{número de total de vetores}} \tag{3.51}$$

Outras métricas de avaliação do desempenho da rede ELM em tarefas de reconhecimento de padrões são a matriz de confusão e os valores de sensibilidade e especificidade para o caso de problemas de classificação binária.

Para validar a rede treinada, ou seja, dizer que ela está apta para ser utilizada, importante testar a sua resposta (saída) para dados de entrada diferentes daqueles vistos durante o treinamento. Estes novos dados podem ser obtidos através de novas medições, o que nem sempre viável. Durante o teste os pesos de saída da rede, em geral, não são ajustados.

Para contornar este obstáculo, o procedimento mais comum consiste em treinar a rede apenas com uma parte dos dados selecionados *aleatoriamente*, guardando a parte restante para ser usada para testar o desempenho da rede. Assim, ter-se-á dois conjuntos de dados, um para treinamento, de tamanho $N_1 < N$, e outro de tamanho $N_2 = N - N_1$. Em geral, escolhe-se N_1 tal que a razão N_1/N esteja na faixa de $0,75$ a $0,90$.

Em outras palavras, se $N_1/N \approx 0,75$ tem-se que 75% dos vetores de dados devem ser selecionados aleatoriamente, sem reposição, para serem utilizados durante o treinamento. Os 25% restantes serão usados para testar a rede. O valor de ε_{teste} é calculado com os dados de teste chamado de *erro de generalização* da rede, pois testa a capacidade da mesma em "extrapolar" o conhecimento aprendido durante o treinamento para novas situações. importante ressaltar que, geralmente, o erro de generalização é maior do que o erro de treinamento, pois trata-se de um novo conjunto de dados.

3.10 Dicas para um Bom Projeto da Rede ELM

A seguir são dadas algumas sugestões para aumentar a chance de ser bem-sucedido no projeto de uma rede neural artificial.

Pré-processamento dos pares entrada-saída Antes de apresentar os exemplos de treinamento para a rede ELM é comum mudar a escala original das componentes dos vetores $\mathbf{x}$ e $\mathbf{d}$ para a escala das funções de ativação logística (0 e 1) ou da tangente hiperbólica (-1 e 1). As duas maneiras mais comuns de se fazer esta mudança de escala são apresentadas a seguir:

Procedimento 1: - Indicado para quando as componentes x_j do vetor de entrada só assumem valores positivos e a função de ativação, $\phi(u)$, é a função logstica. Neste caso, aplicar a seguinte transformação a cada componente de $\mathbf{x}$:

$$x_j^* = \frac{x_j}{x_j^{max}} \tag{3.52}$$

em que, ao dividir cada x_j pelo seu maior valor $x_j^{max} = \max_{\forall t}\{x_j(t)\}$, tem-se que $x_j^* \in [0, 1]$.

Procedimento 2: - Indicado para quando as componentes x_j do vetor de entrada assumem valores positivos e negativos, e a função de ativação, $\phi(u)$, é a função tangente hiperbólica. Neste caso, aplicar a seguinte transformação a cada componente de $\mathbf{x}$:

$$x_j^* = 2\left(\frac{x_j - x_j^{min}}{x_j^{max} - x_j^{min}}\right) - 1 \tag{3.53}$$

em que $x_j^{min} = \min_{\forall t}\{x_j(t)\}$ é o menor valor de x_j. Neste caso, tem-se que $x_j^* \in [-1, +1]$.

Os dois procedimentos descritos acima também devem ser igualmente aplicados às componentes d_k dos vetores de saída, $\mathbf{d}$, caso estes possuam amplitudes fora da faixa definida pelas funções de ativação.

Funo Tangente Hiperbólica: - Tem sido demonstrado empiricamente, ou seja, através de simulação computacional que o processo de treinamento converge mais rápido quando se utiliza a função de ativação tangente hiperbólica do que quando se usa a função logística. A justificativa para isto está no fato da tangente hiperbólica ser uma função par, ou seja, $\phi(-u_i) = -\phi(u_i)$. Daí sugere-se utilizar a função tangente hiperbólica sempre que o problema permitir.

Classificação de padrões: - Quando se treina a rede ELM para classificar padrões é comum usar a codificação de saída descrita na Seção 3.9, em que na especificação do vetor de saídas desejadas assume-se o valor de saída unitrio (1) para o neurônio que representa a classe e nulo (0) para os outros neurônios. Conforme dito no item anterior este valores são assintóticos e portanto, dificilmente serão observados durante a fase de teste.

Assim para evitar ambigidades durante o cálculo da taxa de acerto P_{acerto} durante as fases de treinamento e teste define-se como a classe do vetor de entrada atual, $\mathbf{x}(t)$, como sendo a classe representada pelo neurônio que tiver maior valor de saída. Em palavras, podemos afirmar que se o índice do neurnôio de maior sada é c, ou seja

$$y_c(t) = \max_{\forall k}\{y_k(t)\} \tag{3.54}$$

então a Classe de $\mathbf{x}(t)$ é a Classe c.

4 Metodologia e Resultados

Neste capítulo são descritos os resultados da realização deste trabalho. Inicialmente, serão descritos os blocos que compõem um sistema de reconhecimento de comando de voz desenvolvido e os algoritmos e técnicas empregados. Em seguida, será analisado o desempenho do sistema para conjuntos de parâmetros de entrada diferentes. Após isso, serão comparados os resultados para duas redes neurais diferentes: ELM e OLAM.

4.1 Sistema de Reconhecimento de Comandos Voz

Para este trabalho, foi desenvolvido um Sistema de Reconhecimento de Comandos de Voz - apartir de agora chamado de SRCV - que identifica um entre quatro comandos do conjunto frente, trás, esquerda, direita. O conjuto de comandos pode ser facilmente extendido através do cadastro de um novo comando na interface do sistema. Os comandos testados limitam-se apenas à palavras. Frases inteiras não serão testadas. A figura abaixo mostra os sistemas que compõe o SRCV:

Figura 3.1: Diagrama de blocos dos SRCV. A seguir, será descrita a implementação de cada um desses blocos e os algoritmos empregados.

4.1.1 Aquisição dos Comandos de Voz

O bloco de Aquisição dos Comandos de Voz é responsável por capturar as elocuções através de um microfone inserido na entrada de áudio de um PC. Este bloco basicamente é uma classe Java que modela um gravador de voz e tem como saída um vetor de bytes cujos níveis representam a intensidade do sinal de voz. Os parâmetros utilizados para esse gravador são:

- Taxa de amostragem: 8KHz
- Número de bits por amostra: 8

- Número de canais: 1
- Sinalização: Sim
- Endianess: Big-endian

Vale ressaltar que o hardware utilizado foi a placa de áudio onboard Pebble High Definition Smart Audio da Conexant e um fone de ouvido com microfone da CLONE.

4.1.2 Pré-processamento

O bloco de pré-processamento consiste de um filtro de pré-ênfase que quando aplicado ao sinal de voz digitalizado o torna menos suscetível aos efeitos de precisão finita nas etapas de processamento de sinal posteriores e planifica o seu espectro. A função de transferência deste filtro é:

$$H(z) = 1 - \tilde{a}z^{-1}, 0.9 \leq a \leq 1.0 \tag{4.1}$$

Através da aplicação da transformada Z inversa, foi obtida a equação à diferenças que representa o sistema no domínio do tempo:

$$\tilde{s}(n) = s(n) - \tilde{a}s(n-1) \tag{4.2}$$

Segundo (RABINER; JUANG, 1993), existem duas abordagem para se definir o valor de ã: a dinâmica e a estática. Na abordagem dinâmica, o parâmetro ã pode variar de acordo com algum critério de avaliação. Na abordagem estática o valor do coeficiente ã é fixo. Neste trabalho, a abordagem estática foi empregada e um valor de 0,90 foi utilizado.

4.1.3 Sistema de Recorte

O bloco de recorte, ou detector de atividade de voz (DAV), é responsável por detectar os extremos de uma elocução, ou seja, extrair o silêncio do início e do fim das amostras do sinal voz de uma elocução (RABINER; SAMBUR, 1975) e (RABINER; JUANG, 1993). Através deste bloco, foi possível obter a parte relevante do sinal de voz mesmo na presença de um ruído de fundo. Com isso, a quantidade de amostras que são passadas para os próximos blocos é bem menor, reduzindo assim a utilização dos recursos computacionais e melhorando o desempenho do bloco classificador. As figuras abaixo mostram a forma de onda original da elocução do comando frente e a forma de onda após o sinal ter

passado pelo bloco de recorte respectivamente, para janelas de 10ms de tamanho e 1ms de deslocamento:

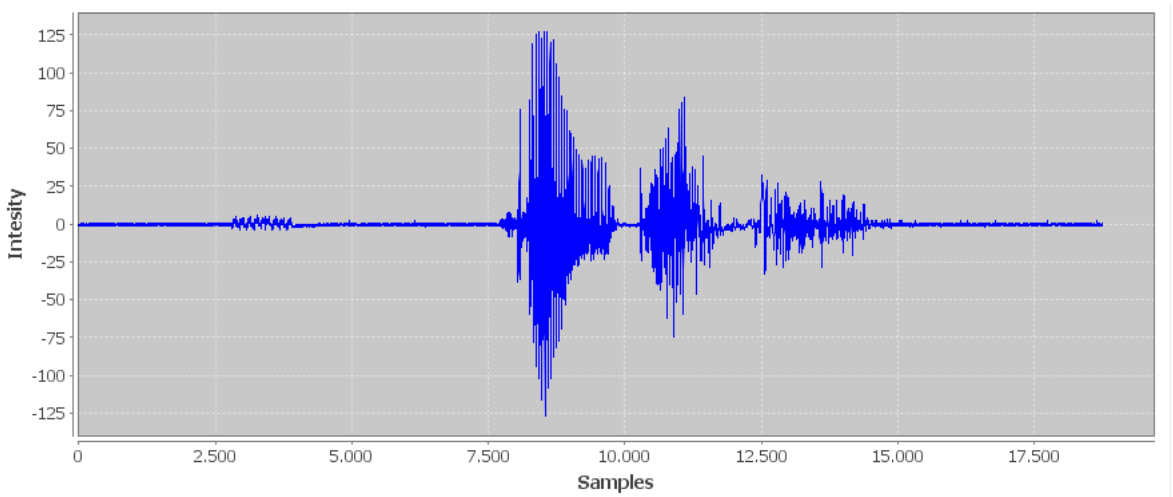


Figura 12: Elocução original do comando frente.

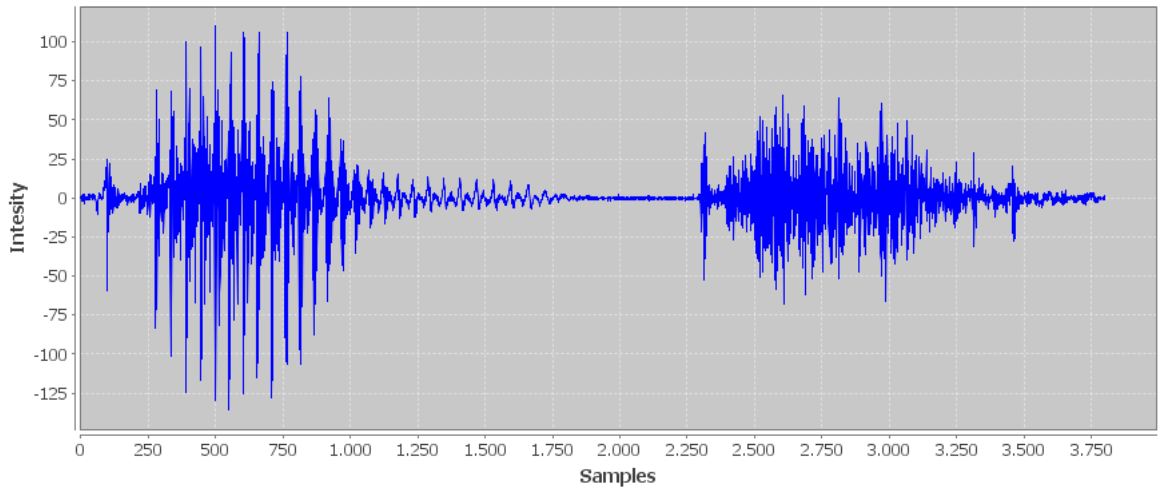


Figura 13: Elocução do comando frente após pré-ênfase e extração de silêncio.

O algoritmo implementado foi aquele proposto em Rabiner e Sambur (1975) devido a sua simplicidade, relativo baixo custo computacional e fácil compreensão. Este algoritmo utiliza duas métricas que são extraídas do sinal de voz quadro à quadro: energia e taxa de cruzamentos por zero. Essas duas métricas possuem valores bastantes distintos quando tratamos de um quadro de silêncio ou de um quadro de voz. É baseado nesta distinção que o algoritmo define limiares para ambas as métricas, duas para energia e uma para cruzamentos por zero. O limiar de cruzamento por zero é chamado de IZTC e os limiares inferior e superior de energia são chamados de ITL e ITU, respectivamente.

Os limiares são obtidos através de uma modelagem estatística das duas métricas nos 100ms iniciais, que geralmente consistem apenas de silêncio e ruído de fundo. Através

desses limiares é possível distinguir entre um quadro de silêncio e um quadro de voz. Após a definição dos limiares, o algoritmo percorre a elocução em ambos os sentidos para encontrar os quadros de início de e de fim da parte relevante da elocução. Com a utilação do DAV foi obtida uma redução média de 71,6% nas amostras da elocução original. As taxas máxima e mínima de redução de amostras foram respectivamente 88,36% e 40,03%.

4.1.4 Representação do Sinal de Voz

Este bloco é responsável pela extração dos coeficientes LPC do sinal de voz. A representação através de LPC foi empregada pois utiliza pouco parâmetros para representar o sinal e é amplamente utilizada na literatura. Todo o processamento é feito em curta duração, ou seja, o sinal de voz é divido em quadros e apartir destes são extraídos p coeficientes LPC, onde p é a ordem do preditor linear. Para este trabalho, a ordem p do preditor foi definida empiricamente como sendo igual a 10.

A extração dos coeficientes LPC em cada quadro foi realizada através do algoritmo de Levison-Durbin. Este algoritmo é utilizado para resolver o sistema linear da Equação 2.12, exposto na Seção 2.3.3 deste trabalho, através da inversão da matrix R_s. Pode-se notar que quase todos os valores da matrix R_s estão presentes na matrix r_s. Devido a essa estrutura particular, o algoritmo aplica recursão para obter os coeficientes ótimos do preditor linear Rabiner e Juang (1993). Segue o pseudo código do algoritmo:

Algorithm 4.1.1: LEVISON-DURBIN(,)

$E^0 \leftarrow \phi[0]$

for $i \leftarrow 1$ **to** p

do
- $k_i = (\phi[i] - \sum_{j=1}^{i-1} \alpha_j^{i-1}\phi[i-j])/E^{i-1}$
- $\alpha_i^{(i)} = k_i$;
- **if** $i > 1$
  - **then**
    - **for** $j \leftarrow 1$ **to** $i-1$
      - **do** $\{\alpha_j^i = \alpha_j^{i-1} - k_i\alpha_{i-j}^{i-1}$
    - **endfor**
- **endif**
- $E^i = (1 - k_i^2)E^{i-1}$

endfor

$\alpha_j = \alpha_j^{(p)} \qquad j = 1, 2, 3..., p$

Após a extração dos coeficientes LPC, estes podem ser submetidos a duas fases distintas dependendo do modo em que o SRCV se encontra. Se o SRCV estiver no modo de aquisição de dados, os LPC são armazenados em um banco de dados para treinar posteriormente o bloco Reconhecedor de Comandos de Voz. Cada coeficiente LPC é armazenado no banco de dados juntamente com um identificador do comando selecionado previamente pelo usuário. Este identificador de comando será utilizado posteriormente para rótular os vetores de entrada da rede neural. Porém, se o SCRV estiver no modo de teste de comandos, os LPC são submetidos ao bloco de Reconhecedor de Comandos de Voz para avaliar qual comando foi emitido.

4.1.5 Reconhecedor de Comandos de Voz

O bloco Reconhecedor de Comandos de Voz consiste de uma rede neural do tipo ELM ou OLAM. Essas duas arquiteturas de redes neurais foram utilizadas para que um estudo comparativo pudesse ser realizado e demonstrar o quão eficiente a rede neural ELM, que é o foco deste trabalho, se comportou. A escolha da rede neural é feita pelo usuário através da interface do SCRV. Para realizar a validação de ambas as arquiteturas de redes neurais, alguns conjuntos de dados, obtidos do UCI Machine Learning Repository, foram empregados. Esses conjuntos de dados, depois de carregados pelo SVR, têm seus vetores de dados embaralhados e divididos em dois conjuntos menores: conjunto de treinamento e conjunto de teste. O tamanho do conjunto de treinamento foi fixado em 80% do tamanho do conjunto original. Os 20% restantes formam o conjunto de teste.

4.1.5.1 Conjunto de Validação 01: Wine

O primeiro conjunto de dados é chamado Wine e possui valores de treze propriedades químicas de vinhos provindos de uma mesma região da Itália, porém de variedades diferentes. De acordo com os fornecedores do conjunto, ele pode ser considerado um conjunto de dados pouco desafiador, ou seja, o conjunto de dados seria facilmente aprendido pelas redes neurais com uma taxa de acerto elevada. Para o caso da rede neural OLAM essa taxa média de acerto foi de 98%.

No caso da rede neural ELM, podemos variar a quantidade q de neurônios da primeira camada (camada oculta). Para cada valor de q obtém-se uma nova taxa de acerto. Os gráficos abaixo mostram respectivamente a evolução da taxa de acerto na rede neural ELM, variando-se a quantidade de neurônios da primeira camada e tempo médio gasto no treinamento da rede neural utilizando o conjunto Wine. A rede é treinada 50 vezes

e a os pesos obtidos no treinamento são aqueles de maior taxa de acerto. A unidade de tempo utilizada nos gráficos é o milissegundo.

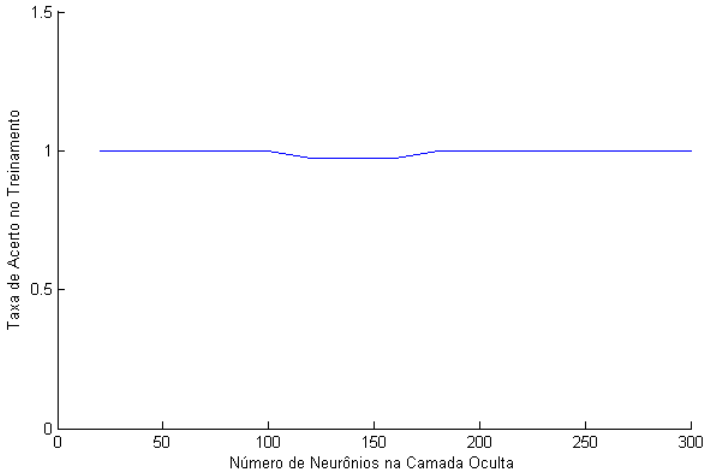


Figura 14: Gráfico da taxa máxima de acerto no teste versus o número de neurônios da camada oculta.

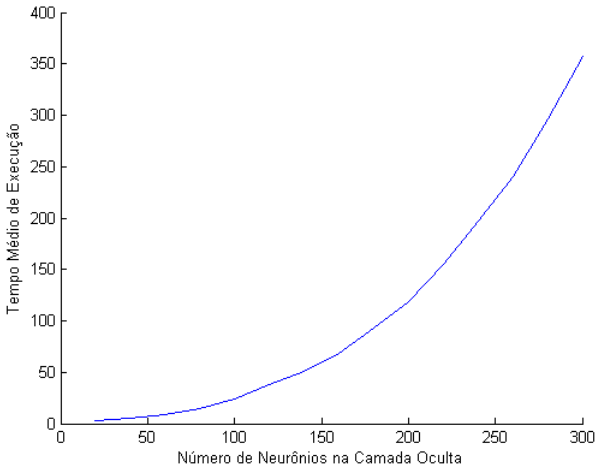


Figura 15: Gráfico do tempo médio de execução (inicialização, treinamento e teste) versus o número de neurônios da camada oculta. A unidade de tempo utilizada neste gráfico é o milissegundo.

4.1.5.2 Conjunto de Validação 02: Wall-Following Robot Navigation Data

O segundo conjunto de dados utilizado é chamado de Wall-Following Robot Navigation Data. Este conjunto de dados possui valores de leituras de sensores de ultrassom dispostos circularmente no corpo de um robô móvel. Ele é subdivido em três partes: um arquivo com vetores de 24 leituras correspondendo a 24 sensores, um arquivo com vetores 4 leituras correspondendo a 4 sensores e outro arquivo com vetores de 2 leituras correspondendo a

2 sensores. De acordo com os fornecedores do conjunto, ele pode ser considerado um conjunto de dados desafiador, impondo assim uma maior dificuldade no seu aprendizado por uma rede neural quando se comparado ao conjunto Wine.

Com o uso desse conjunto é esperada uma diferença mais significativa nas taxas de acerto das redes neurais ELM e OLAM. Através de simulações, uma taxa média de 65,5% foi obtida para a rede OLAM.

Os gráficos abaixo mostram respectivamente a evolução da taxa de acerto na rede neural ELM, variando-se a quantidade de neurônios da primeira camada (camada oculta) e tempo médio gasto no treinamento da rede neural utilizando o conjunto Wall-Following Robot Navigation Data. A rede é treinada 50 vezes e os pesos obtidos no treinamento são aqueles de maior taxa de acerto:

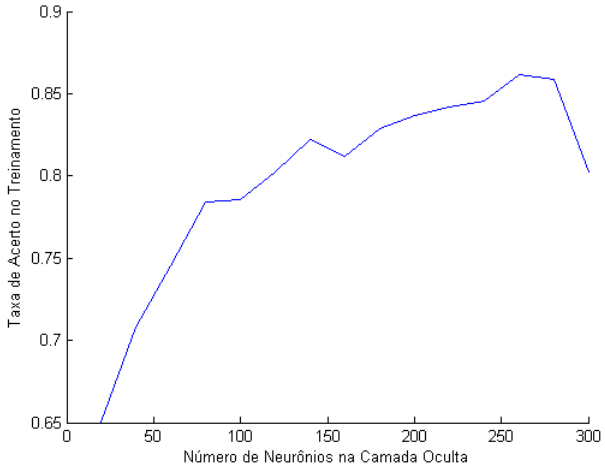


Figura 16: Gráfico da taxa máxima de acerto no teste versus o número de neurônios da camada oculta.

4.1.5.3 Conjunto de Dados do Sinal Voz

O conjunto de dados de voz utilizado para treinamento e teste das redes neurais estudadas neste trabalho foram armazenados num banco de dados Oracle MySQL. Este gerenciador de banco de dados foi escolhido devido sua ampla adoção na comunidade de software livre, ampla documentação, robustez e velocidade. Além disso, ele possui um excelente conjunto de ferramentas de gerenciamento que facilita o desenvolvimento e a manutenção dos dados. Antes de mostrar como os dados de sinais de voz foram utilizados, faz-se necessário entender como estes foram armazenados no banco de dados intitulado vcldb3.

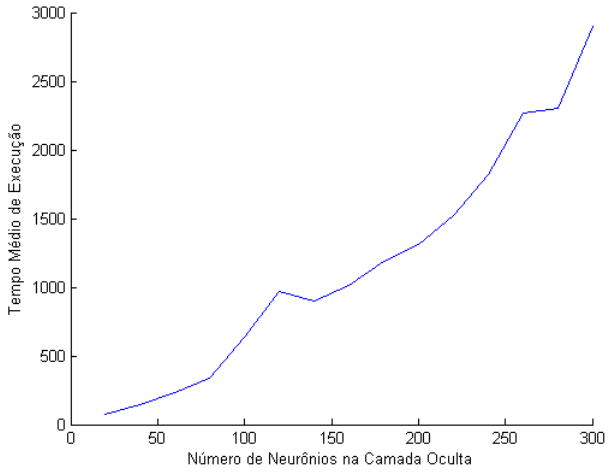


Figura 17: Gráfico do tempo médio de execução (inicialização, treinamento e teste) versus o número de neurônios da camada oculta. A unidade de tempo utilizada neste gráfico é o milissegundo.

4.1.5.4 Modelo de Dados do Sistema de Reconhecimento de Comandos de Voz

O banco de dados vcldb3 é um banco de dados relacional constituído de sete tabelas. Essas tabelas e os seus relacionamentos podem ser observados através do modelo de dados relacional abaixo:

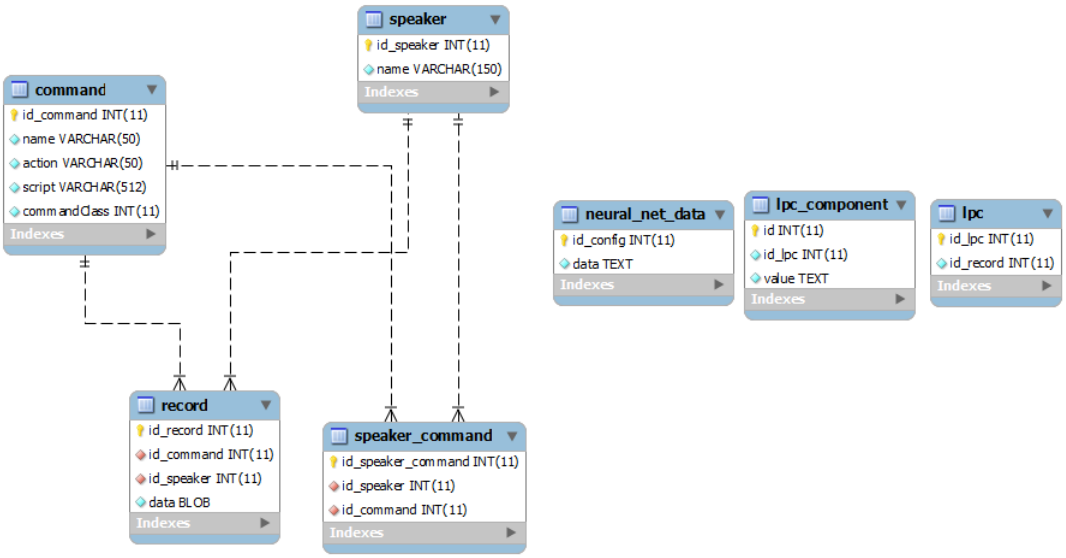


Figura 18: Modelo de dados relacional utilizado no SRCV.

O acesso a banco de dados e a outros tipos de meios persistentes é uma tarefa bastante corriqueira em sistemas que armazenem e/ou acessem uma quantidade relevante de dados. Porém, em alguns casos o tipo da fonte de dados pode mudar deixando de ser um banco

de dados relacional e passando a ser um banco de dados orientado a objetos ou até mesmo um simples arquivo, por exemplo. É ideal que toda a lógica que manipula esses dados não se modifique, mas mude apenas a forma como eles são obtidos e/ou persistidos. Deve-se haver uma transparência entre o código que utiliza os dados em relação como esses dados são obtidos (ALUR; CRUPI; MALKS,). A partir deste fato, devem surgir soluções que, no contexto das linguagens orientadas a objeto, proporcionem reuso, modularidade e uma maior separação entre camadas de software. Uma solução que possui essas características é denominada padrão de projeto.

Neste trabalho, para realizar o acesso aos dados de forma transparente em relação a fonte, foi implementado um padrão de projeto chamado Data Access Object (DAO). Uma classe DAO encapsula métodos de escrita e leitura, e gerencia sua conexão com a fonte de dados (ALUR; CRUPI; MALKS,). A figura abaixo, extraída de Alur, Crupi e Malks (), mostra o diagrama de classes que representa o padrão de projeto DAO:

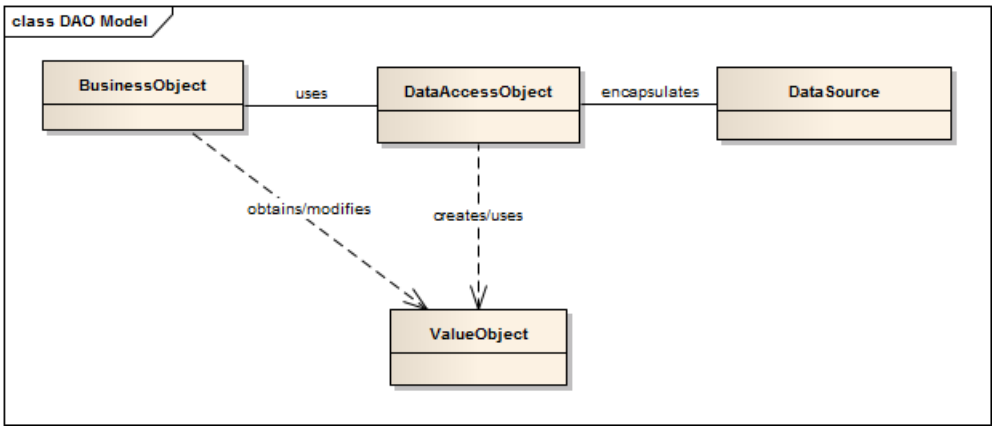


Figura 19: Modelo de classes representando os relacionamentos presentes no padrão de projeto DAO.

Na figura acima podemos notar a presença de algumas classes participantes. A classe BusinessObject representa de forma geral as diversas classes do sistema que precisem manipular ou recuperar os dados para realizar alguma tarefa. Já a classe DataSource modela a fonte de dados seja ela qual for. A classe ValueObject são as entidades que representam dos dados presentes da fonte. Por ultimo, temos a classe DataAccessObject que modela a camada que torna a fonte de dados transparente para a lógica da aplicação. No SRCV, a fonte de dados é um banco de dados MySQL que é abstraído através várias classes DAO, uma para cada entidade. Por exemplo, as elocuções gravadas pelos usuários são modeladas pela entidade chamada Record. Para realizar a persistência de elocuções, existe uma classe RecordDao que torna o processo transparente. Se a fonte de dados mudar, toda lógica que manipula as elocuções permanecerá a mesma. As figuras abaixo mostram as

classes DAO presentes no sistema e suas entidades correspondentes, respectivamente:

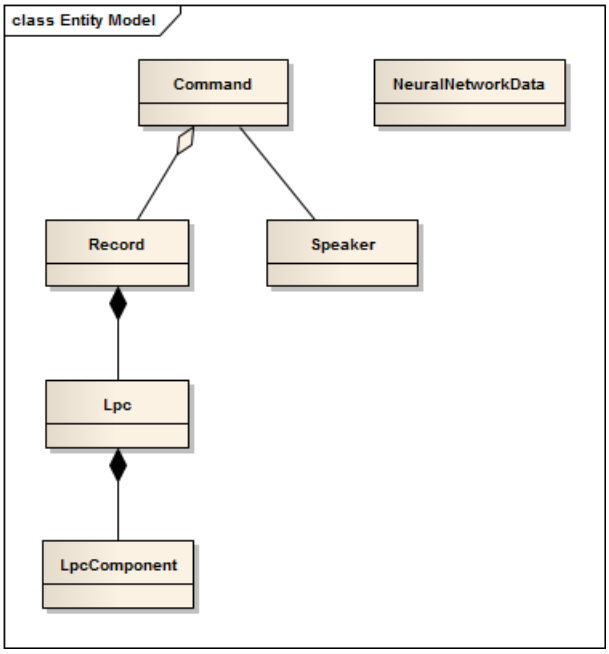


Figura 20: Modelo de entidades utilizado pelos objetos de acesso a dados.

Tendo sido entendido como os conjuntos de dados são gravados, lidos e modificados, pode-se agora mostrar como eles são manipulados pelo bloco Reconhecedor de Comandos de Voz.

4.1.6 Primeiro Cenário: Rede Neural ELM

Para realizar o reconhecimento automático de comandos de voz, foi definida uma arquitetura com 10 entradas, N neurônios da camada intermediária e 4 neurônios de saída. A quantidade de neurônios da camada intermediária foi selecionada empiricamente avaliando-se a taxa de acerto no momento do treinamento.

Para realizar o treinamento da rede neural ELM, foram gravadas 7 elocuções para cada um dos quatro comandos do conjunto frente, trás, esquerda, direita. Vetores de 10 LPCs são armazenados para cada uma das janelas de cada uma das elocuções. Tendo isso em mente, a matriz de treinamento para rede neural ELM é composta pela concatenação em colunas de 100% dos vetores LPC persistidos no banco de dados. Nesta fase, adicionamos uma linha extra acima da matriz dos vetores LPC que foi preenchida com os rótulos dos comandos correspondentes. Esta linha de rótulos é posteriormente extraída da matriz e armazenada internamente para construir os vetores de saídas desejadas. Supondo que

existam L vetores LPC no banco de dados, a matriz de treinamento tem a seguinte forma:

$$\begin{bmatrix} R_0 & ... & R_{L-2} & R_{L-1} \\ C_{00} & ... & C_{0L-2} & C_{0L-1} \\ C_{10} & ... & C_{1L-2} & C_{1L-1} \\ C_{20} & ... & C_{2L-2} & C_{2L-1} \\ \vdots & ... & \vdots & \vdots \\ C_{90} & ... & C_{9L-2} & C_{9L-1} \end{bmatrix}$$

Os gráficos abaixo mostram respectivamente a evolução da taxa de acerto na rede neural ELM, variando-se a quantidade de neurônios da camada intermediária, e tempo médio gasto no treinamento da rede neural utilizando o conjunto de dados de voz. A rede é treinada 50 vezes e a os pesos obtidos no treinamento são aqueles de maior taxa de acerto:

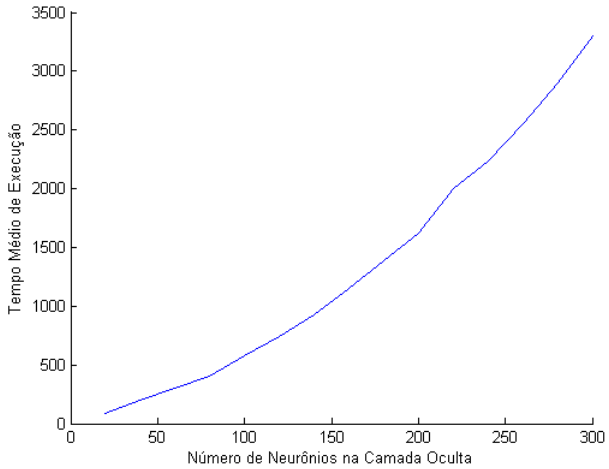


Figura 21: Gráfico da taxa de acerto máxima no treinamento versus o número de neurônios da camada oculta. A unidade de tempo nos gráficos é o milissegundo.

Após a fase de treinamento, o usuário do SRCV poderá selecionar o Modo de Teste de Comando para testar o bloco Reconhecedor de Comandos de Voz. Neste modo, toda elocução gravada gera uma matriz similar a matriz de treinamento, porém com todos os rótulos preenchidos com o valor -1. Cada um dos vetores coluna dessa matriz é submetido à rede neural ELM treinada e os valores presentes nos neurônios de saída são avaliados. O neurônio que possuir maior saída terá seu contador acumulado. Após a avaliação de todos os vetores de entrada, o SCRV usará uma operação de voto majoritário sobre os contadores para identificar o comando, ou seja, indicará ao usuário que o comando reconhecido por ele é aquele de maior contador. A matriz de confusão abaixo demonstra as taxas de acerto

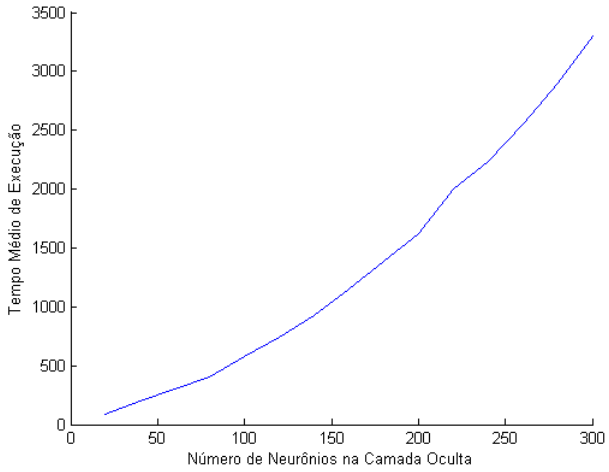


Figura 22: Gráfico do tempo médio de execução (inicialização, treinamento e teste) versus o número de neurônios da camada oculta. A unidade de tempo é o milisegundo.

	Frente	Trás	Esquerda	Direita
Frente	0.91	0	0.09	0
Trás	0	0.83	0.17	0
Esquerda	0	0.05	0.85	0.1
Direita	0.03	0.03	0.22	0.72

Tabela 1: Matriz de confusão no teste da rede neural ELM para os dados de voz.

obtidas:

Como o reconhecimento do comando pelo SRCV é feito através de voto majoritário, faz-se necessário definir uma métrica que indique o desempenho do SRCV em relação a sua precisão. Foi definida uma métrica m_1 que é a razão entre a soma dos três menores contadores e o maior contador quando a rede neural acerta que comando o usuário emitiu. Quanto melhor for a precisão do SRCV, mais próximo a métrica m_1 se aproximará de 0. A tabela resume os resultados relativos à métrica m_1:

4.1.7 Segundo Cenário: Rede Neural OLAM

Neste segundo cenário todo processo se desencadeia da mesma maneira, tendo a rede neural OLAM uma fase de inicialização dos dados praticamente igual da rede ELM. É

	Mínimo	Máximo	Médio
	0.09	0.86	0.45

Tabela 2: Valores mínimo, máximo e médio da métrica m_1 para a rede ELM.

	Frente	Trás	Esquerda	Direita
Frente	0.93	0	0	0.07
Trás	0	0.84	0.16	0
Esquerda	0.045	0	0.91	0.045
Direita	0.35	0.06	0.23	0.33

Tabela 3: Matriz de confusão no teste da rede neural OLAM para os dados de voz.

	Mínimo	Máximo	Médio
	0.29	0.83	0.5

Tabela 4: Valores mínimo, máximo e médio da métrica m_1 para a rede OLAM.

no momento do treinamento que o cenário se diferencia do anterior. Apenas um estágio linear é utilizado pela rede neural, ou seja, existe uma única camada de neurônio cujos parâmetros são obtidos através da operação pseudo-inversa. A taxa máxima de acerto no treinamento e o tempo médio de execução (inicialização, treinamento e teste) foram 55

Assim como na rede neural ELM, esta rede neural se baseia no voto majoritário da avaliação dos contadores de saída da rede neural. A matriz de confusão abaixo demonstra as taxas de acerto obtidas:

Uma avaliação da precisão da rede neural OLAM também foi obtida através da métrica m_1. A tabela resume os resultados relativos à métrica m_1:

5 Conclusão

Através deste trabalho, foi possível observar a estrutura básica de um sistema classificador de padrões e suas peculiaridades. Foram também apresentadas boas práticas que, quando seguidas corretamente, levam a sistemas de classificação de padrões mais robustos e precisos.

Para a realização deste estudo, foi desenvolvido um software que captura e processa os sinais de voz. Tal software seguiu alguns padrões e soluções de Engenharia de Software que trazem modularidade e reuso ao desenvolvimento. Este software foi utilizado como simulador para o processo de reconhecimento das elocuções.

A partir das simulações e dos dados obtidos nas mesmas, pôde-se constatar que a rede neural ELM apresentou uma maior capacidade de generalização do que a rede neural OLAM. Isso pôde proporcionar a ela uma maior precisão, evidenciada pela métrica m_1, no reconhecimento dos comandos de voz atráves da acumulação em contadores, um para cada neurônio de saída da rede neural.

A taxa de acerto no momento do treinamento da rede neural ELM foi superior a da rede neural OLAM demonstrando que o problema de classificação de comandos de voz se enquadra na classe de problemas de classificação não-linear.

Outro aspecto a ser observado, é que a taxa de acerto da rede neural ELM satura a medida em que aumentamos a quantidade de neurônios da camanda oculta. Isto é útil pois pode-se obter um compromisso entre taxa de acerto e tempo de execução da rede neural, ou seja, o tempo nescessário para inicializar, treinar e realizar uma fase de teste na rede neural. Foi constatado que este tempo cresce de forma quase exponencial a medida que se aumenta os neurônios da camada oculta.

5.1 Trabalhos Futuros

Após a simulação SRCV de forma controlada em um computador pessoal, pode-se realizar estudos e adaptações de implementação de forma a executá-lo em um sistema embarcado, onde se deve levar em consideração limitações de processamento, memória disponível e interface com o usuário. Como exemplo de sistemas embarcados, temos o robô móvel SCITOS G5 que poderia ter seu movimento controlado através de comandos de voz. Outra possível extensão deste trabalho seria o reconhecimento de usuários através da fala, onde um usuário seria identificado dentro de um banco de dados de indivíduos.

Referências

ALUR, D.; CRUPI, J.; MALKS, D. *Core J2EE Patterns Best Practices and Design Estrategies*. [S.l.]: Sun Press.

BISHOP, C. M. *Pattern Recognition and Machine Learning*. [S.l.]: Springer, 2006.

CAMPBELL JR, J. P. Speaker recognition: A tutorial. In: *Proceedings of the IEEE*. [S.l.: s.n.], 1997. v. 85, p. 1947–1462.

CHEN, C. H.; PAU, L. F.; WANG, P. S. P. *Handbook of Pattern Recognition and Computer Vision*. [S.l.]: World Scientific Pub Co Inc, 2005.

DELLER, J. R.; HANSEN, J. H. L.; PROAKIS, J. G. *Discrete-Time Processing of Speech Signals*. [S.l.]: John Wiley & Sons, 2000.

HAIGH, J.; MASON, J. Robust voice activity detection using cepstral features. In: *Proceedings of the IEEE Conference on Computer, Communication, Control and Power Engineering (TENCON'93)*. [S.l.: s.n.], 1993. v. 3, p. 321–324.

HAYKIN, S. *Adaptive filter theory*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 1ª. ed. [S.l.]: Prentice Hall, 1994.

HENRIQUE, L. L. *Acústica Musical*. [S.l.]: Fundacao Caloust Gulbenkian, 2002.

HUANG, G. B.; ZHU, Q. Y.; SIEW, C. K. Extreme learning machine: Theory and applications. *Neurocomputing*, v. 70, n. 1–3, p. 489–501, 2006.

HUANG, X.; ACERO, A.; HON, H.-W. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

KOHONEN, T. *Self-Organization and Associative Memory*. [S.l.]: Springer-Verlag, 1989.

LAMEL, L. et al. An improved endpoint detector for isolated word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, v. 29, n. 4, p. 777–785, 1981.

MAKHOUL, J. Linear prediction: A tutorial review. In: *Proceedings of the IEEE*. [S.l.: s.n.], 1975. v. 63, n. 4, p. 561–580.

MARTIN, A.; CHARLET, D.; MAUUARY, L. Robust speech/non-speech detection using lda applied to mfcc. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'01)*. [S.l.: s.n.], 2001. v. 1, p. 237–240.

MARTINS, J. V. de M. L. et al. Deteccao automatica de patologias da laringe usando codificacao por predicao linear e redes neurais mlp. *Congresso Brasileiro de Redes Neurais e Inteligencia Computacional*, 2009.

MORETTIN, P. A.; TOLOI, C. M. C. *Anlise de Sries Temporais*. [S.l.]: Edgard Blcher, 2004.

OPPENHEIM, A. V.; WILLSKY, A. S.; NAWAB, S. H. *Signals & systems (2nd ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

PRINCIPE, J.; EULIANO, N.; LEFEBVRE, W. *Neural and Adaptative Systems: Fundamentals through simulation*. [S.l.]: John Wiley and Sons, INC, 2000.

RABINER, L.; JUANG, B.-H. *Fundamentals of speech recognition*. [S.l.]: Prentice-Hall International, 1993.

RABINER, L.; SAMBUR, M. R. An algorithm for determining the endpoints of isolated utterances. *Bell System Technical Jornal*, v. 54, p. 297–315, 1975.

RABINER, R. W. S. L. R. *Digital Processing of Speech Signals*. New Jersey: Prenctice-Hall, 1978.

RAMIREZ, J. et al. An effective subband OSF-based VAD with noise reduction for robust speech recognition. *IEEE Transactions on Speech and Audio Processing*, v. 13, n. 6, p. 1119–1129, 2005.

RAMREZ, J. et al. Efficient voice activity detection algorithms using long-term speech information. *Speech Comunication*, v. 42, p. 271–287, 2004.

RODRIGUES, F. F.; REIS, M. d. A. A. J. R. Identificador neural de comandos de vocais para o acionamento sem fios de um robô lego mindstorms. *Congresso Brasileiro de Redes Neurais e Inteligencia Computacional*, 2009.

TANYER, S.; OZER, H. Voice activity detection in nonstationary noise. *IEEE Transactions on Speech and Audio Processing*, v. 8, n. 4, p. 478–482, 2000.

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern Recognition & Matlab Intro: Pattern Recognition*. [S.l.]: Academic Press, 2008.

TUONONEN, M.; HAUTAMKI, R. G.; FRANTI, P. Automatic voice activity detection in different speech applications. In: *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia*. [S.l.: s.n.], 2008. p. 1–6.

WANG, J.-F. et al. The design of a speech interactivity embedded module and its applications for mobile consumer devices. *IEEE Transactions on Consumer Electronics*, v. 54, n. 2, p. 870–875, 2008.