Notas do Autor

1. Este texto corresponde a versão revisada após a defesa da tese.
2. O autor pode ser contactado através dos e-mails:

antonio.pedro@ieee.org

timoszczuk@uol.com.br

ANTONIO PEDRO TIMOSZCZUK

**RECONHECIMENTO AUTOMÁTICO DO LOCUTOR
COM
REDES NEURAIS PULSADAS**

Tese apresentada à Escola
Politécnica da Universidade de
São Paulo para obtenção do
Título de Doutor em Engenharia,
junto ao Departamento de
Engenharia de Telecomunicações e
Controle

São Paulo
2004

ANTONIO PEDRO TIMOSZCZUK

**RECONHECIMENTO AUTOMÁTICO DO LOCUTOR
COM
REDES NEURAIS PULSADAS**

Tese apresentada à Escola
Politécnica da Universidade de
São Paulo para obtenção do
Título de Doutor em Engenharia

Área de Concentração:
Sistemas Eletrônicos

Orientador:
Prof. Doutor
Euvaldo Ferreira Cabral Jr.

São Paulo
2004

FICHA CATALOGRÁFICA

Timoszczuk, Antonio Pedro

**Reconhecimento automático do locutor com redes neurais
Pulsadas / Antonio Pedro Timoszczuk -- São Paulo, 2004.
235 p.**

**Tese (Doutorado) – Escola Politécnica da
Universidade de São Paulo. Departamento de Engenharia de
Telecomunicações e Controle.**

**1. Reconhecimento de voz 2. Redes neurais
I. Universidade de São Paulo. Escola Politécnica. Departamento
de Engenharia de Telecomunicações e Controle II.t**

À minha família, que compartilhou esta jornada, nunca deixando de incentivar-me. Cristian, Stefanie e Dulce, obrigado, graças a vocês começa uma nova fase.

AGRADECIMENTOS

Ao amigo e orientador Prof. Dr. Euvaldo F. Cabral Júnior, pelas indicações de rumo nas horas decisivas e pelo apoio e incentivo incansável.

Aos amigos que de uma forma ou de outra colaboraram com este trabalho.

RESUMO

As Redes Neurais Pulsadas são objeto de intensa pesquisa na atualidade. Neste trabalho é avaliado o potencial de aplicação deste paradigma neural, na tarefa de reconhecimento automático do locutor.

Após uma revisão dos tópicos considerados importantes para o entendimento do reconhecimento automático do locutor e das redes neurais artificiais, é realizada a implementação e testes do modelo de neurônio com resposta por impulsos. A partir deste modelo é proposta uma nova arquitetura de rede com neurônios pulsados para a implementação de um sistema de reconhecimento automático do locutor.

Para a realização dos testes foi utilizada a base de dados Speaker Recognition v1.0, do CSLU – Center for Spoken Language Understanding do Oregon Graduate Institute - E.U.A., contendo frases gravadas a partir de linhas telefônicas digitais. Para a etapa de classificação foi utilizada uma rede neural do tipo perceptron multicamada e os testes foram realizados no modo dependente e independente do texto.

A viabilidade das Redes Neurais Pulsadas para o reconhecimento automático do locutor foi constatada, demonstrando que este paradigma neural é promissor para tratar as informações temporais do sinal de voz.

ABSTRACT

Pulsed Neural Networks have received a lot of attention from researchers. This work aims to verify the capability of this neural paradigm when applied to a speaker recognition task.

After a description of the automatic speaker recognition and artificial neural networks fundamentals, a spike response model of neurons is tested. A novel neural network architecture based on this neuron model is proposed and used in a speaker recognition system.

Text dependent and independent tests were performed using the Speaker Recognition v1.0 database from CSLU – Center for Spoken Language Understanding of Oregon Graduate Institute - U.S.A. A multilayer perceptron is used as a classifier.

The Pulsed Neural Networks demonstrated its capability to deal with temporal information and the use of this neural paradigm in a speaker recognition task is promising.

SUMÁRIO

1. INTRODUÇÃO.....	17
1.1 Objetivo e motivação.....	17
1.2 Estrutura da tese	18
1.3 Trabalhos correlatos	19
2. RECONHECIMENTO AUTOMÁTICO DO LOCUTOR (RAL)	26
2.1 Resumo histórico e conceitos	26
2.2 A representação dos sinais de voz.....	28
2.3 Métodos de análise do sinal de voz	30
2.4 Seleção de parâmetros para o reconhecimento automático do locutor	36
2.5 Sistemas de RAL.....	40
3. REDES NEURAIS ARTIFICIAIS	45
3.1 Origem e evolução das redes neurais artificiais.....	45
3.2 Algumas considerações sobre as RNAs	49
3.3 Modelos de neurônios	53
3.4 A RNA do tipo Perceptron Multicamadas ou Multi Layer Perceptron.....	58
3.5 A RNA auto-organizada de Kohonen	62
4. REDES NEURAIS PULSADAS.....	67
4.1 Funcionamento do neurônio biológico.....	67
4.1.1 Estrutura dos neurônios biológicos e potencial de ação	67
4.1.2 Acoplamento entre neurônios	70
4.2 Representação da informação nos neurônios.....	71
4.2.1 A taxa de pulsos como representação.....	71
4.2.2 A informação temporal como representação	72
4.3 Modelos de neurônios pulsados	75
4.3.1 Modelos baseados em condutância e compartimentos	75
4.3.2 Modelos de limiar de disparo.....	77
4.3.3 Modelos de taxa de disparo (rate models)	84
4.4 O modelo de resposta impulsiva discreto.....	84
5. EXPERIMENTOS PRELIMINARES	87
5.1 Metodologia	87
5.2 Base de dados utilizada.....	88
5.3 Testes com um neurônio	91
5.3.1 Resposta a um pulso único.....	92
5.3.2 Resposta a uma sequência de pulsos	98
5.4 Testes de comportamento coletivo.....	102
5.4.1 Neurônios sem realimentação	102
5.4.2 Neurônios com realimentação.....	104
5.4.3 Variação dos parâmetros de tempo	106
6. EXPERIMENTOS DE RECONHECIMENTO DO LOCUTOR.....	109
6.1 Reconhecimento no modo dependente do texto	109
6.1.1 Comportamento inter e intra locutor	109
6.1.2 Análise com mapa auto-organizável	110
6.1.3 Classificador.....	113
6.1.4 Diagrama em blocos do reconhecedor e resultados	114
6.2 Reconhecimento no modo independente do texto	118

6.2.1 Base de dados.....	118
6.2.2 Camada1 do reconhecedor.....	120
6.2.3 Camada2 do reconhecedor.....	122
6.2.4 Classificador.....	123
6.2.5 Reconhecimento utilizando a energia dos filtros	123
6.2.6 Reconhecimento utilizando os coeficientes MFCC	128
7. DISCUSSÃO DOS RESULTADOS E CONCLUSÃO.....	131
7.1 Discussão dos resultados	131
7.2 Conclusão	135
8. REFERÊNCIAS	136

LISTA DE FIGURAS

Figura 2.1 - Diagrama simplificado de um modelo de produção da voz	13
Figura 2.2 - Operações básicas para obtenção do cepstrum complexo	18
Figura 2.3 - Modelo de análise através de predição linear [30]	20
Figura 2.4 - Seqüência para geração do coeficientes <i>Mel</i> -cepstrais	24
Figura 3.1 - Exemplo de modelo de uma RNA	34
Figura 3.2 - O neurônio de McCulloch e Pitts	38
Figura 3.3 - Separação linear de dois padrões no espaço, por uma rede de dois perceptrons	40
Figura 3.4 - Diagrama de um <i>MLP</i> de três camadas clássico	42
Figura 3.5 - Configuração esquemática do <i>Self-Organizing Feature Map</i>	48
Figura 4.1 – Estrutura anatômica do neurônio biológico	52
Figura 4.2 – Potencial de ação	53
Figura 4.3 – Exemplo das possíveis codificações temporais	58
Figura 4.4 – Diagrama esquemático do modelo de Hodgkin-Huxley	60
Figura 4.5 – Conceito do modelo de integração e disparo	62
Figura 4.6 – (A) mostra o processo de transmissão de um pulso entre dois neurônios; (B) mostra o pulso do potencial de ação, que é gerado sempre que $u_j(t)$ excede o limiar v_j , seguido do período refratário dado por $\eta(t - \hat{t}_j)$; (C) mostra o potencial pós-sináptico que resulta do pulso gerado $\varepsilon_{ij}(t - \hat{t})$ ponderado pelo peso w_{ij} , sendo a linha contínua para o caso de um potencial excitatório (EPSP – <i>excitatory pos-synaptic potential</i>) e pontilhada para um potencial inibitório (IPSP – <i>inibitory pos-synaptic potential</i>)	63
Figura 4.7 – Modelo de integração e disparo representado pela equação (4.7)	67
Figura 5.1 - Seqüência de geração dos MFCCs	75
Figura 5.2 – Resposta do neurônio a um pulso único na entrada ($N_o=1, V_t = 0.1,$ $W_i = 1, T_i = 1, T_o = 1$)	77

Figura 5.3 – Resposta do neurônio a um pulso de entrada, com conexão de caráter inibitório ($N_o = 1$, $V_t = 0.1$, $W_i = -1$, $T_i = 1$, $T_o = 1$)	78
Figura 5.4 – Resposta do neurônio para um pulso com $T_i < T_o$ ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 0.1$, $T_o = 1$)	79
Figura 5.5 - Resposta do neurônio para um pulso com $T_i > T_o$ ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 10$, $T_o = 1$)	80
Figura 5.6 – Resposta do neurônio para um pulso de entrada, na condição limiar $T_i = 0.5 T_o$ ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 0.5$, $T_o = 1$)	81
Figura 5.7 - Resposta do neurônio para um pulso com $T_i > T_o$, mostrando o aumento de duração do período refratário ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 1$, $T_o = 0.1$)	82
Figura 5.8 - Resposta do neurônio para uma seqüência de pulsos com $V_{abs}=0$, mostrando a modulação da largura dos pulsos de saída ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 0.1$, $T_o = 1$)	83
Figura 5.9 - Resposta do neurônio para uma seqüência de pulsos com $V_{abs}=1$, quando ocorre a modulação da freqüência dos pulsos de saída ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 0.1$, $T_o = 1$)	84
Figura 5.10 - Resposta do neurônio para uma seqüência de pulsos para $V_{abs}=1$ e com $T_i > T_o$ ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 1$, $T_o = 0.1$) ...	84
Figura 5.11 – Resposta do neurônio com período refratário longo, para uma seqüência de entrada de amplitude variável, resultando na modulação da freqüência dos pulsos de saída	86
Figura 5.12 – Comportamento do arranjo de neurons, quando estimulado pela seqüência de pulsos obtidos a partir dos MFCCs ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 1$, $T_o = 0.1$, $V_{abs}=1$)	87
Figura 5.13 - Comportamento do arranjo de neurons, para a mesma seqüência de MFCCs como estímulo e mesmos parâmetros da Figura 5.12	88
Figura 5.14 – Comportamento do arranjo de neurons com realimentação, para a mesma seqüência de estímulo da Figura 5.12 ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 1$, $T_o = 0.1$, $V_{abs}=1$)	89
Figura 5.15 – Comportamento do arranjo de neurons com realimentação, para a mesma seqüência de estímulo da Figura 5.13	89
Figura 5.16 – Comportamento do arranjo de neurônios com resposta refratária rápida	91

Figura 5.17 – Comportamento do arranjo de neurons com resposta refratária rápida, tendo o limiar de disparo aumentado	91
Figura 5.18 – Comportamento do arranjo de neurons com resposta refratária lenta, combinado com integração da entrada ($T_i=1$, $T_o=0.03$)	92
Figura 6.1 – Distribuição da amplitude dos pesos, após o treinamento da rede SOM com seqüências de ruído branco	95
Figura 6.2 – Mapa de distribuição dos pesos obtido após o treinamento; mapa (A) corresponde ao locutor 01 e o mapa (B) corresponde ao locutor 02	95
Figura 6.3 – Exemplos de trajetórias tridimensionais de dois locutores na rede SOM; (A) corresponde ao locutor 01 e (B) corresponde ao locutor 05	97
Figura 6.4 – Diagrama em blocos do reconhecedor dependente do texto	99
Figura 6.5 – Diagrama em blocos do reconhecedor com redes pulsadas	102
Figura 6.6 – Seqüência resultante da coversão feita pela camada1 ($T_i=10$, $T_o=1$, $V_t=2.6$)	108
Figura 6.7 – Evolução dos pesos durante o treinamento da camada2 – 500 iterações	109
Figura 6.8 – Seqüência de pulsos, após mapeamento pela camada 2	110

LISTA DE TABELAS

Tabela 5.1 – Combinações de comprimento de segmento e deslocamento entre segmentos consecutivos que foram utilizados	74
Tabela 6.1 – Resultados do classificador para testes com o próprio conjunto de treino	100
Tabela 6.2 – Resultados do classificador para o conjunto de testes	101
Tabela 6.3 – Resultado de verificação do balanceamento da base de dados	104
Tabela 6.4 – Resultado de verificação dos conjuntos com classificador MLP para energia dos filtros	104
Tabela 6.5 – Melhores percentuais de reconhecimento com energia dos filtros MEL	111
Tabela 6.6 – Melhor resultado detalhado por locutor	112
Tabela 6.7 – Resultados iniciais de reconhecimento com coeficientes MFCC	113
Tabela 6.8 – Reconhecimento com ampliação do conjunto de treino	114

LISTA DE ABREVIATURAS E SIGLAS

Adaline	- Adaptive Linear Element
ADPCM	- Adaptive Diferential Pulse Code Modulation
CSLU	- Center for Spoken Language Understanding
DCT	- Discrete Cosine Transform
DTW	- Dynamic Time Warping
EPSP	- Excitatory Pos-Synaptic Potential
FFT	- Fast Fourier Transform
INSPEC	- Information Service in Physics, Electrotechnology and Control
IPSP	- Inhibitory Pos-Synaptic Potential
ITS	- Individual Trace Segmentation
LBG	- Linde, Buzo and Gray
LMS	- Least Mean Square
LPC	- Linear Predictive Coding
LTW	- Linear Time Warping ou também Linear Time Normalization
LVQ	- Learning Vector Quantization
MFCC	- Mel Frequency Cepstral Coefficients
MLP	- Multilayer Perceptron
MTI	- Minimal Temporal Information
PCM	- Pulse Code Modulation
RAL	- Reconhecimento Automático do Locutor
RBF	- Radial Basis Function
RNA	- Redes Neurais Artificiais
RNP	- Redes Neurais Pulsadas

SOM	- Self Organizing Map
STANN	- Spatio-Temporal Artificial Neural Network
STFT	- Short Time Fourier Transform
VLSI	- Very Large Scale Integration
VQ	- Vector Q

1. INTRODUÇÃO

1.1 Objetivo e motivação

Apesar de existirem métodos bem desenvolvidos para o Reconhecimento Automático do Locutor (RAL), seja utilizando redes neurais artificiais ou modelos estatísticos, o crescente interesse no paradigma de Redes Neurais Pulsadas (RNP) faz com que seja importante avaliá-lo na solução de problemas de RAL. Este paradigma neural tem sido estudado mais intensamente a partir da última década, e os trabalhos publicados têm enfatizado o desenvolvimento e o aperfeiçoamento de modelos que aproximem o funcionamento do neurônio biológico.[1][2]

No que se refere à aplicação das redes pulsadas a problemas práticos, a literatura que trata da utilização deste paradigma neural para o RAL é bastante escassa. Em recente pesquisa realizada na base de dados INSPEC de literatura científica, no período de janeiro de 1969 a março de 2003 foram indicadas apenas 4 referências, o que torna este trabalho indubitavelmente inovador.

Um outro aspecto motivador, para a avaliação deste paradigma neural aplicado ao RAL, é a capacidade de lidar com representações temporais a qual vem sendo revelada pelas redes pulsadas. Este fato torna este paradigma neural muito atrativo para a solução de problemas que envolvam sinais variantes no tempo, como é o caso do sinal de voz.

Assim, esta tese tem como principal objetivo original estudar a utilização do paradigma das Redes Neurais Pulsadas para a tarefa do Reconhecimento Automático do Locutor (RAL).

As etapas que fazem parte deste trabalho compreendem:

- o estudo de um modelo de neurônio pulsado, sua implementação através de procedimentos computacionais e avaliação dos parâmetros que compõem este modelo;
- a definição da arquitetura de uma rede com neurônios pulsados, para ser aplicada na tarefa do reconhecimento automático do locutor;
- testes de verificação do seu funcionamento e avaliação dos resultados alcançados.

Os testes para a verificação do funcionamento da rede proposta são desenvolvidos por meio do reconhecimento do locutor de forma dependente e independente do texto. A base de dados utilizada nos testes é a *Speaker Recognition v1.0*, do *CSLU – Center for Spoken Language Understanding do Oregon Graduate Institute* - E.U.A., que compreende locuções gravadas a partir de linha telefônica com locutores de diversas faixas etárias e sexo, proferindo diferentes frases.

Ao final deste trabalho, terá sido proporcionada uma visão de como este paradigma neural funciona e apresentado um método para a utilização das Redes Neurais Pulsadas no Reconhecimento Automático do Locutor.

1.2 Estrutura da tese

No capítulo 1, é apresentado o objetivo e a organização dos capítulos que compõem este trabalho, seguido de uma revisão dos trabalhos relacionados com o processamento de sinais de voz, utilizando as RNPs.

Os conceitos relacionados com a tarefa de reconhecimento automático do locutor, são descritos no capítulo 2, onde é apresentada uma revisão das características do sinal de voz e um resumo dos métodos de processamento usuais.

Um resumo do histórico das redes neurais artificiais (RNAs) é apresentado no capítulo 3, onde são descritos os principais aspectos relacionados com o uso das RNAs. Neste capítulo são apresentados também, alguns modelos clássicos de redes, que são utilizados ao longo deste trabalho. No capítulo 4, é feito um aprofundamento dos aspectos biológicos que deram origem às RNAs, para prover conceitos básicos necessários para a descrição do funcionamento das RNPs. Neste mesmo capítulo são apresentados os principais modelos de neurônios pulsados e feitas considerações sobre a implementação de redes utilizando estes neurônios.

O capítulo 5 é dedicado à parte experimental, onde é apresentada a análise do funcionamento do modelo de neurônio pulsado escolhido, tanto do ponto de vista de um neurônio isolado como o de arranjos de neurônios. Neste capítulo, também é

descrita a base de dados utilizada e a estrutura do reconhecedor. Os resultados dos testes realizados para o reconhecimento do locutor são apresentados no capítulo 6.

No capítulo 7 são discutidos os resultados obtidos, apresentadas as conclusões e sugeridas questões para serem estudadas em futuros trabalhos.

1.3 Trabalhos correlatos

Para ser capaz de analisar detalhadamente os trabalhos correlatos, é necessário que o leitor tenha conhecimento de diversos conceitos básicos sobre o funcionamento das redes pulsadas, os quais são apresentados mais adiante no capítulo 4 desta tese. Assim, para evitar uma exposição que seja de difícil compreensão neste ponto do trabalho, a revisão de trabalhos correlatos, apresentada a seguir, será limitada a uma apresentação da literatura considerada relevante e diretamente relacionada com o processamento de sinais de voz e o reconhecimento automático do locutor.

Durante a realização da pesquisa bibliográfica para este trabalho, foi observado que a grande maioria das publicações relacionadas com as RNPs relatam pesquisas relativas a modelagem e investigação teórica da capacidade computacional deste tipo de rede. A pesquisa sobre a aplicação prática tem se concentrado no reconhecimento de padrões visuais e poucas publicações referenciam pesquisas sobre as RNPs aplicadas ao processamento de sinais de voz. Uma possível explicação para esta concentração sobre o reconhecimento de padrões visuais seria a origem das pesquisas sobre neurônios pulsados terem como fonte trabalhos investigativos sobre o córtex visual de seres vivos.

Através da pesquisa de um modelo utilizando neurônios pulsados, para determinação da direção de uma fonte sonora, Kuronagi e Iwata [3] apresentam diversos aspectos importantes, como a conversão de um sinal analógico em pulsos e a utilização dos neurônios pulsados para medida de diferenças temporais. No primeiro estágio do modelo, demonstram um arranjo de neurônios pulsados combinado com pré-processamento, que modela o processo de transdução do sinal de áudio pelo ouvido humano. Os neurônios pulsados neste estágio atuam como codificadores, transformando os sinais analógicos de entrada em seqüências de pulsos. As

seqüências de pulsos são utilizadas como entrada para um arranjo de neurônios subsequente, que efetua um processamento baseado na diferença entre os tempos das seqüências de pulsos correspondentes ao ouvido esquerdo e direito, e no nível de sinal codificado pela seqüência de pulsos, para a avaliação da direção de origem da fonte sonora. O modelo de neurônio utilizado neste segundo artigo é semelhante ao utilizado nesta tese, com diversas simplificações no seu estágio de saída. É importante observar que todos os parâmetros da rede utilizada foram ajustados de forma experimental e interativa. Isso caracteriza a ausência de algoritmos de treinamento para a rede pulsada quando da publicação até então.

Em uma outra publicação cinco anos mais tarde, os mesmo pesquisadores lançam a proposta de um algoritmo de treinamento para as redes pulsadas, como continuidade do trabalho anterior [4]. Neste artigo, os autores demonstram o funcionamento de um algoritmo para o ajuste dos pesos das conexões de entrada, baseado na relação entre as amplitudes dos sinais de entrada e o estado interno do neurônio. Juntamente com o ajuste de pesos, é introduzida a componente de atraso na conexão de entrada e apresentado um algoritmo que permite ajustar o valor destes atrasos de forma automática. O ajuste requer um sinal de referência para o neurônio, que define quando é feita a avaliação das entradas para o treinamento. A definição da saída que está sendo treinada em um dado momento, deve ser informada para rede, caracterizando assim o treinamento como supervisionado.

A utilização das redes pulsadas como quantizador vetorial é proposta em outro artigo de Kuronayagi e Iwata, onde é apresentado um algoritmo para o ajuste de pesos da rede de forma não supervisionada [5]. Este algoritmo é muito semelhante ao utilizado nas redes de quantização vetorial, fazendo uso dos mesmos princípios utilizados no algoritmo das redes auto-organizáveis de *Kohonen* que foi adaptado para utilizar parâmetros relativos ao nível de atividade do neurônio, em lugar da distância *Euclidiana*. Neste algoritmo, todos os neurônios que são ativados têm os pesos de suas conexões ajustados. Um aperfeiçoamento deste algoritmo consistiu em introduzir o aspecto de competição entre os neurônios. A competição entre os neurônios foi introduzida através do ajuste automático do limiar de ativação dos neurônios a cada iteração, de forma que apenas um neurônio seja ativado de cada

vez. Quando é atingida esta condição, é processado o ajuste dos pesos apenas para o neurônio que foi ativado – vencedor [6].

Este conjunto de artigos, de autoria de Kuronayagi e Iwata, apresenta um resumo bastante didático das possibilidades de utilização das redes pulsadas e através de sua cronologia pode-se observar a alteração do tipo de problema abordado. Inicialmente é tratada a codificação temporal dos sinais de entrada, seguido da otimização da codificação desenvolvida através da adaptação dos métodos de quantização vetorial para as redes pulsadas, passando então para o desenvolvimento de algoritmos de aprendizado supervisionado e não supervisionado.

Ainda sobre o problema de detecção da direção de uma fonte sonora, Schauer et.al. [7] apresentam um modelo em seis camadas, onde cada uma das camadas efetua tarefas específicas. A rede possui camadas para a conversão do sinal analógico em seqüências temporais de pulsos, detecção de coincidências e seleção da direção da fonte sonora. As camadas de detecção e seleção são testadas com conexões laterais entre os neurônios vizinhos, e demonstram que para esta tarefa as conexões laterais melhoram a resposta da rede. Esta estruturação foi apresentada visando a implementação em *hardware* desta rede, para utilização em robôs, o que mostra mais uma característica das RNPs, que, devido às suas funções internas serem tipicamente do tipo exponencial ou de comparação, tornam-se extremamente promissoras para serem implementadas em *hardware*.

A segmentação de sinais sonoros utilizando RNPs, é apresentada por Smith [8], onde uma rede de neurônios pulsados é utilizada para integrar a resposta de um banco de filtros em frequência e no tempo. O trabalho de Smith é inspirado no pré-processamento auditivo dos mamíferos e faz uso de uma técnica de filtragem denominada de *onset-offset*, a qual permite obter um sinal que indica o início e o término da presença de uma determinada banda em frequência do filtro de entrada correspondente. Esta é uma técnica similar àquela utilizada para a detecção de bordas em imagens (transição preto-branco e branco-preto). As seqüências obtidas para cada banda em frequência são então integradas no tempo e na frequência, por intermédio de uma RNP. Esta rede apresenta um arranjo unidimensional, e o número de bandas utilizadas como entrada de cada neurônio provê a integração em frequência. A

integração temporal é obtida através das constantes de tempo associadas à função de ativação dos neurônios. A saída de cada um destes neurônios é realimentada para um conjunto de vizinhos, fazendo com que os neurônios interconectados sincronizem os disparos caso algum neurônio do grupo venha a disparar. O resultado desta sincronização é um mapa temporal usado para segmentar a locução que originou o mesmo. Neste trabalho, os pesos da rede foram mantidos fixos e iguais e o efeito de sincronização foi obtido adequando os valores dos períodos refratários segundo observações experimentais do sinal de voz. Este artigo mostra como uma rede composta por neurônios pulsados, pode ser utilizada para executar o processamento do espaço tempo-frequencial. A implementação desta rede em *hardware* é relatada em Glover et al [9].

Geralmente as conexões entre os neurônios, denominadas de pesos, são representadas por um fator de eficácia e são alteradas apenas durante a fase de treinamento da rede, permanecendo estáticas após esta fase. Liaw e Berger [10][11], fazendo uso de neurônios pulsados, apresentam uma modelagem sofisticada da conexão entre neurônios e sugerem que estas conexões desempenham papel vital na diversidade de representação proporcionada por uma RNP. A variação dinâmica dos pesos é explorada tendo como aplicação o reconhecimento de palavras isoladas e o reconhecimento automático do locutor. Esta variação dos pesos é denominada de “sinapse dinâmica” e é inspirada nos mecanismos biológicos que governam o funcionamento das conexões dos neurônios naturais. Com este tipo de mecanismo espera-se diminuir significativamente a quantidade de neurônios, utilizada em uma rede e dada a sua capacidade de representação, eliminar a necessidade da obtenção de parâmetros significativos no caso do reconhecimento de voz. Nos experimentos apresentados no artigo de Liaw e Berger, as redes recebem nas entradas o sinal de voz sem nenhum pré-processamento e demonstraram robustez ao ruído. A estratégia de conexões dinâmicas tem sido muito estudada e apresenta ainda várias questões a serem solucionadas, tais como o estabelecimento dos parâmetros das conexões dinâmicas, que normalmente são obtidos experimentalmente ou através de similaridade com medidas observadas em neurônios biológicos.

As conexões dinâmicas, também são usadas por Storck [12], onde as palavras “sim” e “não” são identificadas por uma rede em que a entrada corresponde a 128

neurônios que recebem a saída da transformada rápida de Fourier do sinal de voz de entrada, transformando-a em uma sequência de pulsos. As conexões dinâmicas são entre camada de entrada e a camada de saída, que é formada por 2 neurônios, cada um correspondendo a uma palavra a ser identificada. A identificação se dá quando a sequência de pulsos produzida por uma classe de entrada, apresenta uma característica temporal tal que produza um valor máximo de energia de ativação do neurônio correspondente, fazendo com que o mesmo gere um pulso ou trem de pulsos em sua saída.

Uma rede de neurônios pulsados é utilizada como codificador de padrões temporais, para gerar sequências de entrada para um classificador denominado STANN (*Spatio-Temporal Artificial Neural Network*) [13]. O sinal de voz é pré-processado por métodos clássicos, gerando vetores de características que são analisados por meio de quantização vetorial, para compor os padrões que serão submetidos à rede pulsada. O classificador executa o processamento da sequência temporal gerada pela camada de neurônios pulsados, por meio de métodos de medida de integração dos pulsos de entrada em amplitude e fase, comparando com padrões previamente armazenados.

A geração de vetores de características (*features*), que permitam identificar o locutor de forma dependente do texto, é o objetivo da RNP utilizada por Chandrasekaran et al [14]. O vetor de entrada da rede é composto pelo espectrograma da locução sob a forma de uma imagem que é apresentada para a rede de uma única vez. A rede tem todos os neurônios com os mesmos parâmetros, todos fixos, o mesmo ocorrendo com os pesos. A rede apresenta uma topologia definida pelas conexões laterais entre os neurônios, as quais também apresentam parâmetros fixos. A cada espectrograma apresentado é efetuada a leitura da saída dos neurônios durante 50 iterações, sendo calculada a quantidade de neurônios que dispara a cada iteração, compondo um vetor de 50 elementos com dimensão significativamente menor do que a entrada originalmente apresentada para a rede. Este vetor é denominado de ícone, sendo usado para a caracterização do locutor. A classificação é feita apresentando este vetor obtido, para um classificador neural clássico. No artigo de Chandrasekaran, as redes pulsadas com conexões de realimentação, demonstram possuírem uma característica de resposta própria em função dos seus parâmetros.

Na tese apresentada por Ho [105] foi proposta uma arquitetura de rede com neurônios pulsados, para a extração de características do sinal de voz. Esta rede recebe como estímulos, seqüências originadas em um banco de filtros cocleares e as transforma em seqüências de pulsos. A etapa de classificação é feita utilizando-se um classificador neural clássico.

Uma série de artigos muito interessantes de Hopfield e Wills [15][16][17] demonstra uma forma de sincronizar o disparo de neurônios pulsados, fazendo com que os mesmos desempenhem a função de detectores de coincidência para a identificação de palavras isoladas. No primeiro destes artigos sob a forma de um desafio acadêmico, Hopfield lança os resultados de uma rede, cuja arquitetura não é revelada propositalmente. Nos dois artigos seguintes quase um ano depois são apresentadas tanto a arquitetura como a descrição do funcionamento desta rede. A questão interessante neste trabalho é a demonstração de como o controle de constantes de tempo, nas conexões de entrada dos neurônios, permite associar a atividade máxima na saída de um neurônio com a presença de um padrão específico na sua entrada. A seleção adequada das constantes de tempo que governam a frequência de disparo dos neurônios, faz com que funções decrescentes monotonicamente ao longo do tempo, responsáveis pela ponderação das entradas de um neurônio, apresentem um único ponto de coincidência. Este ponto é utilizado como condição de ativação do neurônio, que apresenta um trem de pulsos na sua saída, quando da detecção desta coincidência. No experimento de Hopfield, o controle desta ativação é feito por meio de uma função que apresenta um máximo no instante da coincidência e decai linearmente até zero.

Dentre os trabalhos analisados até aqui, a maioria tem aplicação na área de voz para o reconhecimento de palavras isoladas e pré-tratamento para segmentação de locuções. Também são significativos os resultados obtidos na detecção da direção de uma fonte sonora. Poucos são os trabalhos que mencionam o reconhecimento automático do locutor ou tratam especificamente desta aplicação [11][14][16].

O que pôde ser verificado, através da revisão dos artigos mencionados, é que as redes pulsadas apresentam-se como fortes candidatas ao processamento de sinais de voz, pois conseguem extrair informações a partir do dinamismo e variabilidade do sinal

de entrada. Em especial as redes com conexões dinâmicas apresentam-se como poderosos classificadores que podem ser implementados em uma quantidade relativamente pequena de neurônios. Porém ainda não está totalmente clara a melhor forma de implementar este tipo de rede, que deve ser objeto de intensa pesquisa nos próximos anos.

Outro ponto importante a ser destacado, é o fato de que as redes com neurônios pulsados não requerem algoritmos sofisticados de pré-processamento para obtenção de resultados satisfatórios. É notada também a falta de algoritmos de treinamento amplamente formalizados e aceitos, tal como ocorre com os modelos de redes neurais artificiais considerados clássicos. Os algoritmos utilizados para o aprendizado nas redes pulsadas são inspirados nos algoritmos clássicos e certamente a busca por formas de treinamento mais adequadas para este novo tipo de rede, faz destas redes um vasto campo para a pesquisa. Outra constatação é que os modelos de neurônios utilizados sugerem a sua implementação em hardware, pelo fato de serem baseados em circuitos capacitivos e comparadores relativamente simples.

2. RECONHECIMENTO AUTOMÁTICO DO LOCUTOR (RAL)

2.1 Resumo histórico e conceitos

O Reconhecimento Automático do Locutor (RAL) consiste em reconhecer automaticamente a identidade de um indivíduo, através da comparação de características extraídas do seu sinal de voz. Relatos históricos mostram que, por volta de 1660, uma testemunha afirmou, em uma das sessões de investigação das circunstâncias que levaram a morte Charles I, que poderia reconhecer um réu por meio de sua voz. Não houve interesse científico no reconhecimento do locutor até que a telefonia e a tecnologia de gravação tornaram-se disponíveis permitindo o reconhecimento de forma independente da distância e em tempo não real. Desde então, foram desenvolvidas as primeiras técnicas de reconhecimento do locutor baseadas na audição e nos espectrogramas (*voice print*). A partir de 1966, a corte na Inglaterra passou a admitir as análises baseadas em espectrogramas do sinal de voz como provas válidas para a identificação de indivíduos. Assim, a primeira aplicação prática do reconhecimento do locutor foi de natureza forense, no apoio à solução de casos na justiça [18].

Todas as técnicas para o reconhecimento do locutor até então existentes eram subjetivas, pois dependiam de especialistas muito bem treinados para a interpretação auditiva de trechos de fala ou interpretação visual dos espectrogramas. Cabe notar atualmente que no Brasil, a maioria dos casos em que é solicitada uma perícia para a identificação de um indivíduo por meio da fala, esta é realizada por especialistas através destes métodos subjetivos.

A evolução dos sistemas computacionais permitiu que as técnicas de reconhecimento automático do locutor até então desenvolvidas apenas do ponto de vista teórico, pudessem ser implementadas de forma eficiente, fazendo com que os sistemas de RAL atingissem níveis adequados de desempenho e proporcionando sua utilização em aplicações reais. A partir de então pôde ser observada uma ampliação nas aplicações do RAL. Inicialmente o reconhecimento do locutor foi utilizado em sistemas de segurança, com o propósito de restrição de acesso, seja a informações ou

instalações físicas e mais recentemente passou a ser utilizado em equipamentos comerciais, dentre eles sistemas embarcados automotivos e brinquedos [19].

Os sistemas de RAL podem ser classificados segundo duas tarefas a serem executadas: **a identificação** do locutor e **a verificação** do locutor. Na verificação, a locução de entrada é comparada apenas contra o padrão da identidade informado pelo indivíduo que fornece a locução de entrada, fornecendo uma decisão binária do tipo SIM / NÃO. Na identificação, é assumido um conjunto de elementos cada qual com o seu padrão, e a locução de entrada é comparada contra todos os elementos do conjunto fornecendo uma indicação de qual padrão melhor representa a locução de entrada. Neste caso, a decisão não é do tipo binário e pode resultar em nenhum dos elementos do conjunto. De uma forma resumida, a verificação e a identificação são problemas de reconhecimento de padrões que compartilham as mesmas etapas de processamento, diferindo essencialmente no estágio final de classificação [20].

Outra forma de classificar os sistemas de RAL é através do texto pronunciado pelo locutor, durante o reconhecimento. Os sistemas que trabalham com textos fixos, ou seja, para o reconhecimento é necessário que o locutor pronuncie um texto pré-definido, são denominados **dependentes do texto**. Por outro lado, os sistemas nos quais não existe a necessidade da utilização de textos pré-definidos, são denominados **independentes do texto**.

Revisões abrangentes sobre a tarefa do RAL podem ser encontradas em Atal [21], Corsi [22], Gish [23] e Campbell [24].

Uma vez que o RAL consiste basicamente na comparação de padrões, é necessário que seja definida uma forma de representação do sinal de voz e quais são as características mais adequadas que devem ser utilizadas para a montagem dos padrões a serem utilizados no reconhecimento. Nas seções seguintes serão apresentadas as formas mais utilizadas de representação do sinal de voz, os métodos de análise e a seleção das características mais adequadas do sinal de voz, para o reconhecimento automático do locutor.

2.2 A representação dos sinais de voz

A escolha da representação dos sinais é fundamental para o sucesso de qualquer sistema de processamento de voz, entendendo-se aqui não só o reconhecimento da fala e do locutor, mas também qualquer sistema de comunicação que utilize a voz como principal meio de transmissão das informações. Atualmente, a grande maioria das aplicações como transmissão de voz, sistemas de resposta automática, sistemas de síntese de voz e sistemas de reconhecimento do locutor vêm utilizando a representação digital do sinal de voz. Dois motivos para este fato são apontados em Schafer [25]: a disponibilidade de computadores digitais rápidos permitindo a aplicação de diversas técnicas de processamento digital de sinais e o fato deste tipo de abordagem facilitar a implementação de soluções diretamente em circuitos integrados levando a sistemas comerciais economicamente factíveis. Neste mesmo artigo, é apresentada uma análise das principais formas de representação digital dos sinais de voz, abrangendo desde o modelo de produção da voz humana, a codificação do sinal de voz e os diversos métodos de análise.

Para estabelecer uma representação dos sinais de voz, é necessário considerar um modelo para o aparelho fonador humano, que é a fonte destes sinais. Uma representação física do aparelho fonador humano muito utilizada é aquela onde o trato vocal é representado por um tubo acústico, que contém em uma de suas extremidades as cordas vocais e na outra os lábios; o trato nasal é representado por um tubo adjacente cuja conexão é determinada pelo velum. A partir deste modelo, uma representação digital para a geração dos sinais de voz pode ser equacionada, por meio de um filtro digital variável no tempo controlado por coeficientes que representam os parâmetros do trato vocal, conforme mostrado na Fig. 2.1.

A excitação deste filtro é feita através de um trem de impulsos quase periódico representando os sons vozeados* ou através de uma fonte de ruído branco responsável pelos sons não vozeados. Este modelo é a base para uma série de representações dos sinais de voz e sua função de transferência é apresentada a seguir.

* Em lugar dos termos vozeados e não vozeados uma alternativa seria o uso dos termos **sonoros** e **surdos** respectivamente, uma vez que os mesmos são utilizados pelos foneticistas (ou fonetistas).

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^p a_k z^{-k}} \quad (2.1)$$

Fig. 2.1 - Diagrama simplificado de um modelo de produção da voz.

Com relação a equação (2.1), cabe comentar que o modelo do trato vocal representado por esta equação é simplificado (modelo só de pólos), de forma que sua solução pode ser feita por meio de um sistema de equações lineares. O modelo ideal do trato vocal inclui pólos e zeros, onde estes últimos correspondem a parcela do modelo relativa aos fonemas nasais e fricativos. A simplificação adotada para obter a equação (2.1), utiliza a informação de magnitude da densidade espectral de potência, o que é considerado aceitável para a grande maioria das aplicações uma vez que o ouvido humano não percebe a informação de fase, e no caso do modelo só de pólos tem uma resposta de fase mínima.

A forma mais simples de representar o sinal de voz é a sua representação no domínio do tempo, fazendo uso do teorema da amostragem (Teorema de Nyquist- Shannon), o qual afirma que qualquer sinal limitado em banda pode ser reconstituído a partir de amostras tomadas periodicamente no tempo, desde que esta periodicidade seja pelo menos duas vezes maior que a maior frequência do sinal amostrado.

Cabe esclarecer que a representação do sinal de voz no meio digital é denominada digitalização e envolve as seguintes etapas: a *amostragem* que consiste em tomar amostras periodicamente no tempo, a *codificação* que determina um código com número de bits que pode ser variável para representar cada uma das amostras obtidas e a *quantização* que fixa o valor representado pelos bits que codificam cada uma das amostras.

Dois tipos de codificação utilizados em processamento de voz são o PCM (*Pulse Code Modulation*) e o ADPCM (*Adaptive Differential Pulse Code Modulation*). No caso do PCM, as amostras são quantizadas em um número N de bits, representando a amplitude do sinal ao longo do tempo. No ADPCM o número de bits utilizados para representar um sinal é reduzido, aproveitando a correlação existente entre amostras sucessivas do sinal. Desta forma apenas a diferença entre as amostras sucessivas é codificada, podendo ser feita através de uma combinação linear das amostras prévias, resultando em uma representação mais compacta.

O objetivo principal da codificação do sinal de voz é permitir sua transmissão de forma eficiente e possibilitar a sua reconstrução da forma mais fiel possível, após sua recepção. Em várias aplicações, dentre elas o RAL, a codificação é utilizada apenas como uma etapa intermediária do processamento para a obtenção de características representativas do sinal de voz e seus respectivos parâmetros, através de diversos métodos de análise. É importante observar neste ponto, que os parâmetros são responsáveis pela quantificação das características do sinal de voz. Cada característica é representada por um conjunto de parâmetros.

2.3 Métodos de análise do sinal de voz

- **Análise em tempo curto (*short time analysis*)**

Um conceito fundamental para a análise do sinal de voz é a propriedade de estacionariedade, a qual é obtida através do conceito de análise em tempo curto. Este método compreende a análise de segmentos curtos (de 10 a 30 mseg) nos quais os parâmetros do sinal de voz podem ser considerados invariantes no tempo, fato esse que constitui uma característica do sinal de voz [25].

- **Análise de picos**

A análise no domínio do tempo pode ser efetuada de diversas formas. Uma destas formas é a análise de picos, uma vez que o sinal de voz apresenta picos que tem sua periodicidade condicionada a frequência fundamental (*pitch period*) e as frequências formadoras^(*) que são características do trato.

- **Análise da energia**

Um outro método é a análise da energia, a qual provê um bom meio para a discriminação de trechos de sinal que contém voz daqueles que contém silêncio. No entanto alguns cuidados devem ser tomados neste tipo de análise: primeiro é a escolha adequada de uma janela, em segundo lugar a escolha adequada do tamanho do intervalo a ser analisado (comprimento da janela), pois se esta for menor do que o intervalo da frequência fundamental, perdem-se as características lentamente variáveis do sinal e por outro lado se o comprimento da janela for muito grande a função de energia apresentará variação lenta perdendo as características variáveis do sinal de voz. Entende-se por janela uma função que é multiplicada pelo segmento de voz a ser analisado. Esta função apresenta valor nulo nos extremos do segmento de voz e a transição do valor máximo no centro do segmento para os extremos é suave e tem por finalidade evitar que o segmento analisado tenha seus extremos cortados de forma abrupta o que provocaria a introdução de componentes indesejáveis no domínio da frequência (*spectral leakage*).

- **Taxa de cruzamento de zero**

A taxa de cruzamento de zero (*zero-crossing rate*) consiste de um método de análise no domínio do tempo, que corresponde a medida da quantidade de mudanças da amplitude do sinal de voz de positivo para negativo (ou vice-versa). Se a taxa de cruzamento de zero é alta, indica um segmento não vozeado do sinal; caso a taxa de cruzamento de zero seja baixa, indica grande probabilidade do segmento ser vozeado. Cuidados devem ser tomados na utilização deste método, pois níveis DC presentes no sinal e interferências de 60Hz podem provocar desvios neste tipo de medida.

^(*) O termo frequências formadoras equivale ao termo formantes que é usualmente encontrado na literatura. O mesmo foi introduzido por Cabral[30] em sua dissertação de mestrado e é utilizado ao longo deste trabalho.

- **Autocorrelação**

A autocorrelação é uma forma útil em determinar a estrutura de qualquer forma de onda, inclusive sinais de voz. A periodicidade da função de autocorrelação indica periodicidade no sinal analisado, e o contorno da função de auto-correlação pode indicar a periodicidade do sinal analisado.

- **Análise espectral**

A análise do espectro em frequência, feita a partir de uma análise em tempo curto, tem sido tradicionalmente um dos mais importantes métodos de processamento do sinal de voz. A principal ferramenta utilizada neste tipo de análise é a transformada de Fourier de segmentos curtos (STFT), consecutivos e sobrepostos do sinal de voz, a qual provê uma boa representação para o espectro em frequência do sinal analisado. Dois métodos têm sido usualmente empregados para a obtenção de parâmetros provenientes da análise de Fourier em tempo curto: bancos de filtros e o espectro de magnitude obtido com o algoritmo *FFT* (*Fast Fourier Transform*).

No primeiro caso, a implementação é feita utilizando um banco de filtros passa-banda sintonizados ao longo da banda espectral do sinal de voz e a saída dos mesmos pode ser considerada como uma integração de vários componentes de Fourier do sinal de entrada. No caso do algoritmo FFT, a solução é essencialmente digital o que facilita a sua implementação em computadores digitais, sendo superior ao método que utiliza banco de filtros.

A representação espectral obtida, pode ser usada diretamente como uma representação do sinal de voz. Em muitos casos no entanto, o espectro obtido é utilizado como uma etapa intermediária para a obtenção de outros parâmetros do sinal de voz, tais como a frequência fundamental, frequências formadoras do trato, dentre outros [26]. Uma das formas mais utilizadas para a análise do sinal de voz a partir de sua representação espectral é o espectrograma que consiste na representação da variação das frequências de um trecho do sinal de voz ao longo do tempo.

- **Processamento homomórfico do sinal de voz**

Um outro método de análise do sinal de voz é o processamento homomórfico, que faz parte das técnicas não-lineares de processamento de sinal, sendo baseado na generalização do princípio de superposição que define os sistemas lineares [27]. Sistemas que atendam a generalização do princípio de superposição são denominados **sistemas homomórficos**, uma vez que podem ser representados através de um mapeamento linear (homomórfico, ou que preserva a forma) da entrada para a saída.

A partir do modelo utilizado para a geração do sinal de voz descrito anteriormente, o problema de análise do sinal de voz, em segmentos curtos o suficiente no tempo, pode ser encarado como um problema de separação das componentes de uma convolução da função excitação (pulsos quase periódicos ou ruído branco) com a função de resposta impulsiva do trato vocal, ou seja, um problema de deconvolução proveniente do modelo da Fig. 2.1.

A partir destas considerações, a deconvolução homomórfica pode ser aplicada para separar estes sinais. O processo consiste em tomar o resultado da transformada de Fourier do sinal de voz e calcular a transformada inversa de Fourier do logaritmo deste sinal. O logaritmo é aplicado com o objetivo de transformar o produto entre a função de excitação e o trato, em uma soma.

$$\log(S) = \log(EV) = \log(E) + \log(V) \quad (2.2)$$

onde: S = transformada de Fourier do sinal de voz

E = parcela correspondente a função de excitação

V = parcela correspondente a função de transferência do trato vocal

$$c(n) = \mathfrak{F}^{-1} \log(S) = \mathfrak{F}^{-1} \log(E) + \mathfrak{F}^{-1} \log(V) \quad (2.3)$$

O resultado desta transformação, em função das partes real e imaginária da transformada de Fourier, é o chamado **cepstrum complexo** (Fig. 2.2).

Fig. 2.2 - Operações básicas para obtenção do cepstrum complexo.

A forma de cepstrum comumente utilizada e aplicada neste trabalho é o chamado **cepstrum real** ou simplesmente **cepstrum**. A diferença básica entre estas duas formas de cepstrum é descrita em [28], e consiste em que o cepstrum descarta a informação de fase. Embora o cepstrum complexo seja mais preciso, apresentando informações que são perdidas no cepstrum real, sua aplicação prática é mais onerosa em termos computacionais. Contudo uma das formas de cepstrum mais utilizadas contemporaneamente é aquela derivada de coeficientes de predição linear, que assume o sinal de voz parametrizado como sendo de fase mínima, situação na qual as representações complexa e real do cepstrum se equivalem. O cepstrum complexo é normalmente utilizado em aplicações que sejam sensíveis a fase, tais como codificadores de voz do tipo vocoders, que não são objeto deste trabalho.

A obtenção do cepstrum real ou cepstrum é feita através da mesma sequência descrita anteriormente, sendo que o logaritmo é aplicado apenas à magnitude da transformada de Fourier, correspondente ao produto entre a excitação e a resposta ao impulso do trato vocal. Obtém-se assim um sinal composto da soma de duas componentes, o cepstrum da excitação e da resposta do trato vocal.

O cepstrum apresenta-se como uma excelente ferramenta para estimar a frequência fundamental de um trecho de fala, ou para determinar a presença do sinal de voz. A função de transferência do trato vocal, correspondente ao contorno do espectro em frequência, pode ser obtida removendo-se as componentes rápidas do sinal obtido pela transformação acima descrita. Uma das formas de obter-se esta função de transferência é calcular a transformada discreta de Fourier do cepstrum obtido. A

análise dos picos desta função obtida é uma boa maneira de estimar a *pitch* bem como as frequências formadoras.

Uma vez que o cepstrum retém toda a informação contida no espectro de tempo-curto, o mesmo pode ser encarado como uma forma alternativa de representação para o sinal de voz. Assumindo que o modelo de produção da voz é o descrito anteriormente, e considerando que a função de transferência do trato vocal tenha apenas pólos (*all-pole model*) existe uma fórmula apresentada nas equações (2.4) e (2.5), que relaciona a resposta impulsiva do trato vocal ao cepstrum, permitindo o cálculo de coeficientes cepstrais para a representação do sinal de voz [25].

- **Análise por predição linear**

Um dos métodos mais conhecidos para a codificação dos sinais de voz, é baseado no princípio de predição linear. A idéia básica na codificação através da predição linear - LPC (*Linear Predictive Coding*) - é a de que uma dada amostra do sinal de voz pode ser aproximada através de uma combinação linear de amostras anteriores. Através da minimização do quadrado da diferença entre a amostra atual e as amostras obtidas pela predição linear, podem ser determinados os coeficientes da combinação linear ou os coeficientes do preditor.

Assumindo que o polinômio de predição represente o denominador da função de transferência do trato vocal, que conforme mostrado anteriormente pode ser modelado por um filtro só de pólos, a respectiva resposta em frequência pode ser obtida. Existem os seguintes métodos para a obtenção dos coeficientes de predição linear [29]:

1. método da covariância;
2. método da autocorrelação;
3. método *lattice*;
4. método da filtragem inversa;
5. método de estimação espectral;
6. método da máxima verossimilhança;
7. método do produto interno.

Fig. 2.3 - Modelo de análise através de previsão linear [30].

De acordo com Cabral [30], os quatro últimos métodos são equivalentes aos três primeiros. O método *lattice* é uma combinação dos dois primeiros e esses diferem apenas nos limites das equações.

A partir da resposta em frequência, utilizando o método de detecção de picos, é possível estimar as frequências formadoras do trato vocal. Através do erro de previsão podem ser estimadas propriedades importantes da excitação, tais como a frequência fundamental e determinação de segmentos vozeados ou não. Outro ponto importante na previsão linear é a existência de uma fórmula recorrente apresentada a seguir, a qual permite relacionar os coeficientes de previsão linear com o cepstrum da resposta do trato vocal.

$$c_1 = a_1 \quad (2.4)$$

$$c_n = \sum_{k=1}^{n-1} \left(1 - \frac{k}{n}\right) a_k c_{n-k} + a_n \quad , \quad 1 < n \leq p \quad (2.5)$$

Em Furui [31] é apresentado um sistema para verificação do locutor que utiliza coeficientes cepstrais obtidos a partir de coeficientes LPC.

2.4 Seleção de parâmetros para o reconhecimento automático do locutor

Considerando o RAL como um problema de reconhecimento de padrões (O'Shaughnessy [32]) as principais etapas envolvidas no processo de reconhecimento podem ser consideradas como sendo:

- normalização;
- parametrização;
- montagem dos vetores de parâmetros (padrões);
- medida de similaridade e decisão.

Cabe notar que a seleção eficiente de parâmetros proporciona benefícios óbvios para o sistema de reconhecimento. Dentre estes benefícios podem ser destacados o menor número de dados armazenados, o uso de procedimentos de classificação mais rápidos e menos complexos, além de erros finais de classificação menores.

Uma análise para escolha eficiente de parâmetros acústicos é efetuada por Wolf [33], onde são definidas as características ideais dos parâmetros para o reconhecimento do locutor, a saber:

- ocorrência natural e freqüente na fala;
- ser facilmente mensurável;
- apresentar a maior variação possível entre locutores, e a menor possível para um mesmo locutor;
- não variar ao longo do tempo, ou ser afetada pela condição física do locutor;
- ser na medida do possível imune ao ruído ambiente e ao meio de transmissão; e
- não ser facilmente descaracterizada conscientemente pelo próprio locutor.

Uma forma de comparar a efetividade de parâmetros do locutor muito citada na literatura, porém não a mais efetiva, é a razão-F (*F-ratio*). A razão-F é definida como:

$$F = \frac{\text{variância média inter - locutores}}{\text{variância média intra - locutores}} \quad (2.6)$$

É importante destacar que altos valores de F não garantem que um determinado conjunto de parâmetros seja efetivo, pois a razão-F tende a ser alta para parâmetros onde um ou dois locutores são muito diferentes dos outros. A razão-F também não proporciona nenhuma informação sobre possíveis dependências entre os parâmetros, em Goldstein [34] é sugerido que a média das razões-F de diversos parâmetros (*features*) seja utilizada como forma de eliminar aqueles menos promissores. Uma

forma de considerar os parâmetros de forma conjunta, incluindo os efeitos de interdependência, é generalizar a razão-F por meio da medida de divergência de Kullback (Atal [21]).

Na busca de um conjunto de características que melhorem o reconhecimento do locutor, uma avaliação extensa dos parâmetros acústicos durante a produção de consoantes nasais e estridentes é relatada por Sambur [35]. Um ponto importante destacado por este trabalho é que a efetividade de qualquer parâmetro está intimamente ligada ao classificador que o utiliza. Algumas conclusões importantes apontadas neste trabalho são:

- a precisão da análise por predição linear depende da separação, largura de banda e amplitude das frequências formadoras adjacentes, especialmente da segunda e terceira frequências formadoras;
- os segmentos de fala nasais são muito bons para a obtenção de parâmetros para o reconhecimento do locutor;
- a frequência fundamental apresenta variações ao longo das seções de gravação, que feitas com intervalos de meses e anos, diminuem sua efetividade no reconhecimento do locutor.

Um outro estudo de parâmetros é apresentado em Goldstein [34], onde a trajetória e a frequência das quatro primeiras frequências formadoras são investigadas para um conjunto específico de sons. As conclusões obtidas, apontam como características para o reconhecimento do locutor os valores mínimos e máximos das frequências formadoras, porém as diferenças caracterizadas por estes parâmetros dependem mais dos hábitos de fala do locutor do que das características do trato vocal em si.

Uma avaliação importante dos parâmetros mais utilizados nos sistemas de processamento de voz atuais é apresentada em Picone [36], onde as representações utilizadas compartilham algumas importantes características. Em primeiro lugar são constituídas por informações de origem temporal e em segundo lugar tendem a ser perceptualmente compatíveis com o sistema auditivo humano, por exemplo, o uso do logaritmo da energia (potência em decibéis) ou o uso de bancos de filtros distribuídos segundo escalas perceptuais (escala Bark ou Mel) na análise espectral, procuram emular a resposta em frequência do ouvido humano. As expressões a seguir definem

o mapeamento das frequências acústicas f para as escalas perceptuais Bark, definida como *critical band rate*, e Mel.

$$Bark = 13 \operatorname{atan}\left(\frac{0.76f}{1000}\right) + 3.5 \operatorname{atan}\left(\frac{f^2}{7500^2}\right) \quad (2.7)$$

$$Mel = 2595 \log_{10}\left(1 + \frac{f}{700.0}\right) \quad (2.8)$$

Finalmente, os parâmetros tendem a ser normalizados de alguma forma, especialmente de forma estatística. Na análise apresentada por Picone, são avaliados 31 sistemas de reconhecimento do locutor disponíveis na época, e 21 utilizam os coeficientes cepstrais ou alguma variação dos mesmos, indicando estes como parâmetros dominantes. Os coeficientes Mel-cepstrais são os mais utilizados, colocando os coeficientes de predição linear LPC em segundo plano. O vetor de parâmetros mais utilizado consiste de coeficientes cepstrais, coeficientes delta-cepstrais, a energia e a delta-energia. A técnica de normalização mais utilizada é a da normalização pela variância [37]. Coeficientes delta-cepstrais são os coeficientes gerados a partir da diferença entre dois coeficientes cepstrais consecutivos. Desta forma é criada uma representação para a variação dos coeficientes cepstrais ao longo da locução. Da mesma maneira os coeficientes delta-delta-cepstrais, muitas vezes citados na literatura, codificam a variação dos delta-cepstrais. Este mesmo conceito é aplicável à energia ou qualquer outro coeficiente que apresente variação ao longo da locução.

Outro estudo que ratifica esta conclusão é apresentado em Runstein [38], onde diversos parâmetros são analisados para aplicação em um sistema classificador baseado em RNAs. Os parâmetros que apresentam melhor desempenho são os Mel-cepstrais, destacando que os coeficientes gerados utilizando a DCT apresentam os melhores resultados. Isto provavelmente ocorre devido a uma propriedade da DCT, que é ordenar os coeficientes obtidos por ordem de variância, colocando aqueles que apresentam maior variância em primeiro lugar; assim os coeficientes obtidos via DCT provavelmente carreguem mais informação do que os seus correspondentes obtidos através da FFT inversa. Outros estudos que também confirmam a eficiência dos coeficientes Mel-cepstrais podem ser verificados em Jankowski [39] e [40]. Um

diagrama bastante objetivo de como gerar os coeficientes Mel-cepstrais utilizando a DCT é apresentado em Reynolds [41] e aqui reproduzido na Fig. 2.4 .

Fig. 2.4 - Seqüência para geração dos coeficientes *Mel*-cepstrais.

2.5 Sistemas de RAL

Algumas abordagens utilizadas para o reconhecimento automático do locutor são apresentadas a seguir com objetivo de destacar pontos considerados importantes para o projeto de um sistema de RAL.

O trabalho de Rosenberg [20] aborda o problema da verificação do locutor, sendo avaliados alguns fatores que afetam a implementação prática desses sistemas, a saber:

- tamanho da população;
- variação dos parâmetros entre as seções de gravação;
- características físicas do locutor;
- ambiente de gravação e o meio de transmissão;
- resistência a mímicos.

Rosenberg analisa dois sistemas práticos para a verificação do locutor, sendo o primeiro baseado na análise espectral através de um banco de filtros (*Texas Instruments Entry Control System*) com a comparação do número de coincidências. No segundo sistema do *Bell Labs* a decisão é efetuada através da comparação dos contornos da frequência fundamental, frequências formadoras e coeficientes de

predição linear, por meio de uma medida de distância entre o padrão previamente armazenado e o conjunto sob verificação.

Parâmetros tipicamente fonéticos foram utilizados no trabalho de Glenn [42], onde é feito um estudo da identificação do locutor através de sons nasalizados. Com populações pequenas da ordem de 10 a 30 locutores e utilizando o método da correlação, foram atingidos de 93% a 97% de acertos durante a identificação, indicando que os sons nasalizados apresentam muitas informações que caracterizam o locutor. Esta conclusão é importante, sendo levada em consideração para a escolha das frases utilizadas neste trabalho.

Uma visão de procedimentos considerados importantes, no processo de identificação do locutor, é dada em BRICKER [43]. Neste trabalho são utilizadas duas bases de dados, uma com dez locutores e outra com 172 locutores; a representação utilizada para o sinal de voz é a densidade espectral de energia obtida através de um banco de filtros, que é representada digitalmente através de matrizes (para cada locução). A representação assim obtida consiste de uma massa muito grande de dados, indicando a necessidade de **redução da dimensionalidade** dos dados, permitindo que estes sejam tratados de forma eficiente. Esta redução é feita através da utilização das seguintes representações:

- a somatória da energia nas diversas bandas ao longo do tempo, eliminando a informação da distribuição espectral;
- a somatória da energia ao longo do tempo para cada banda, eliminando a informação de variação temporal do espectro;
- a distribuição da energia ao longo do tempo, para cada banda (*frequency slices*);
- representações combinadas.

É importante observar o aspecto de normalização da energia, como forma de compensar a variação entre as diversas repetições. Uma vez estabelecida a redução da dimensionalidade dos dados, é necessário determinar uma representação espacial para estes dados através de uma transformação/métrica, de maneira a constituir os agrupamentos de interesse (*clusters* [39]) que serão associados a cada locutor. Para cada agrupamento de interesse é determinado um elemento representante (centróide) que será utilizado como padrão de cada locutor. Este conjunto transformação/métrica

deve ser tal que maximize a distância inter-locutores e minimize a distância intra-locutores, propiciando assim uma discriminação eficiente evitando ao máximo a superposição dos *clusters*. Uma vez estabelecida a representação e a métrica [45] a ser utilizada, pode-se passar à fase de classificação que neste caso é dividida em duas estratégias:

- no caso de um pequeno grupo de locutores, a estratégia consiste em calcular a distância entre o locutor desconhecido e cada um dos centróides, que representam os diversos locutores, atribuindo a identificação àquela classe representada pelo centróide mais próximo;
- no caso de um grupo muito grande de locutores, a estratégia acima torna-se pouco prática do ponto de vista computacional. Neste caso deve ser estabelecida uma forma de restrição do conjunto de locutores que será comparado, para a identificação do locutor desconhecido. Uma maneira de efetuar esta restrição é aplicar uma análise discriminante produzindo um sub-espço no qual os locutores formam os agrupamentos de interesse, considerando apenas as características mais relevantes, o locutor desconhecido é então mapeado sobre este sub-espço e a comparação é feita apenas com os locutores do agrupamento de interesse mais próximo. Caso o sub-espço obtido não apresente formação de agrupamentos, este pode ser dividido arbitrariamente formando “agrupamentos artificiais”, que serão utilizados para comparação.

Outro ponto importante a ser considerado em sistemas de RAL é o alinhamento temporal das locuções comparadas, uma vez que as mesmas não apresentam a mesma duração devido a variação na velocidade com que são pronunciadas. No trabalho de Furui [31], é descrita a técnica de *Dynamic Time Warping (DTW)*, que consiste em criar uma função de mapeamento entre o contorno de duas locuções (*warping function*) com determinadas restrições. A função ótima é determinada através da distância mínima calculada sobre a função de mapeamento, esta mesma distância é utilizada para a determinação do padrão mais próximo do padrão sob teste. Em Cabral [46], é apresentada uma forma alternativa para a normalização no tempo que é o método de *Individual Trace Segmentation (ITS)* o qual conduz a um desempenho comparável a DTW, porém com um custo computacional sensivelmente menor. Em Linford [47] é destacada a técnica de normalização linear (*Linear Time*

Normalization - LTW), demonstrando que a mesma é uma alternativa simples e robusta para a normalização de locuções. Um método de organização dos parâmetros obtidos de uma locução, para a apresentação dos mesmos à um classificador, dispensando a normalização no tempo, é apresentado por Timoszczuk [48]. A reorganização consiste em agrupar os parâmetros resultantes de janelas sequenciais, em conjuntos temporais denominados de ^{**}*Minimal Temporal Information – MTI*, que são apresentados para um classificador baseados em RNAs.

Soluções atualmente utilizadas para a identificação automática do locutor, de forma independente do texto, são descritas em Gish [23] e consistem em métodos de modelamento probabilístico das características do locutor. Ao invés de efetuar a classificação baseando-se em medidas de distância sobre valores médios das características, são utilizadas distribuições de probabilidade para caracterização dos parâmetros e a decisão é tomada baseada em probabilidades ou critérios de verossimilhança.

Outra abordagem muito utilizada no reconhecimento do locutor (e de voz de uma maneira geral), é a quantização vetorial ou *VQ (vector quantization)*. Consiste em montar um conjunto de padrões para cada locutor (*codebook*), sendo estes padrões adaptados de forma a minimizar o erro de quantização. O reconhecimento se dá codificando a locução sob teste, segundo o conjunto de padrões, e tomando-se como resultado aquele que apresente o menor erro de quantização dentro de certos limites pré-estabelecidos (*threshold*) [28]. O sucesso desta abordagem depende fundamentalmente da geração adequada do conjunto de padrões; para isso duas abordagens muito utilizadas são o algoritmo *K-means* [49] [50] (ou *Lloyd's algorithm*) e o LBG (Linde, Buzo and Gray) [51] que é uma expansão do caso anterior. Um exemplo de aplicação das técnicas de quantização vetorial ao RAL é apresentada em Burton [52]. A abordagem consiste em utilizar uma sequência de vetores representando uma locução, em lugar de um único vetor por locução. Na verdade com esta abordagem foi tentado destacar as informações fonéticas e de coarticulação, sendo que a melhoria no reconhecimento utilizando estas informações adicionais foi apenas marginal.

^{**} MTI – *Minimal Temporal Information* é um termo criado por Timoszczuk e Cabral [48].

No sistema desenvolvido neste trabalho, um ponto importante tem sido deixado a parte embora seja de fundamental importância nos sistemas práticos, é o ruído nas suas diversas formas: ruído periódico, ruído impulsivo, ruído em banda larga e interferência devido a presença de vozes estranhas. Tais ruídos podem corromper de tal forma o sinal de voz, fazendo com que a taxa de reconhecimento seja drasticamente reduzida. Algumas das principais técnicas para a eliminação de ruído são:

- filtros estacionários;
- transformação espectral (*spectral shapping*);
- interpolação;
- processamento não linear;
- subtração espectral; e
- cancelamento adaptativo.

Uma descrição mais detalhada destes métodos pode ser encontrada em Kurashima [53].

3. REDES NEURAIS ARTIFICIAIS

3.1 Origem e evolução das redes neurais artificiais

O desenvolvimento das RNAs pode ser creditado a diversos motivos, dentre os quais tem-se:

- entender o funcionamento do sistema nervoso e particularmente o cérebro humano;
- simular aspectos de funcionamento do cérebro humano, para desenvolver métodos que possibilitem aos computadores desempenhar bem funções que são cotidianas para o cérebro, tais como reconhecimento de imagens, reconhecimento de voz, inteligência, aprendizado e compreensão.

O cérebro é formado por unidades funcionais aparentemente simples, denominadas de neurônios, que emitem impulsos elétricos em resposta a estímulos. Estas unidades são densamente interconectadas, o que resulta em uma arquitetura altamente complexa e com uma capacidade computacional ainda não igualada por nenhum computador. Inspirados nesta arquitetura complexa, vários modelos matemáticos foram desenvolvidos para os neurônios, cada qual refletindo o conhecimento biológico sobre o funcionamento do cérebro humano disponível na época. Métodos para a interconexão dos modelos de neurônios foram criados, ou desenvolvidos a partir da observação de unidades funcionais biológicas, surgindo então os modelos de redes neurais artificiais (RNAs).

Um modelo de neurônio artificial, pode ser descrito de forma resumida, como uma unidade de processamento que possui uma ou várias entradas e uma saída que é acionada por meio de uma função de transferência da entrada para a saída. A interconexão destas unidades simples é feita por meio de conexões, denominadas sinapses, sendo estas conexões ponderadas por pesos. Esta interconexão dos neurônios, permite obter funções matemáticas altamente complexas, aplicáveis para a solução de diversos problemas práticos.

O primeiro modelo de neurônio surgiu em 1943, a partir dos estudos de como representar eventos do sistema nervoso [54]. McCulloch e Pitts, um neurofisiologista

e um matemático respectivamente, formularam um dos primeiros modelos de neurônio. O próximo grande passo veio em 1949 através de HEBB [55], com o livro *The Organization of Behaviour*. Neste livro, foi relatada pela primeira vez uma regra de aprendizado para um conjunto de neurônios interligados, a partir da modificação dos pesos das sinapses: a regra de Hebb. A regra de treinamento de Hebb foi elaborada em função da hipótese de que as conexões internas do cérebro, estão continuamente sendo alteradas na medida em que diferentes tarefas vão sendo aprendidas pelo organismo. Esta regra determina que a conexão entre dois neurônios é reforçada, na medida em que o primeiro neurônio ativa o segundo repetidamente através desta conexão. Vários pesquisadores trabalharam sobre a idéia de Hebb, dentre eles ASHBY(1952) [56], MINSKY(1954) [57] e UTLEY(1956) [58], culminando com a criação do perceptron por ROSENBLATT em 1958 [59].

Widrow e Hoff em 1960 [60], na Universidade de Stanford, desenvolveram uma importante variação do algoritmo de aprendizado do perceptron. Através da aplicação do conceito de erro médio quadrático (*Least Mean Square algorithm - LMS*), formularam o *Adaline (Adaptive Linear Element)* ou elemento linear adaptivo. Utilizando esta teoria, em 1962 Widrow definiu uma das primeiras redes construída com elementos adaptativos, o *Madaline (Multiple-adaline)*. Durante os anos 60 foram desenvolvidos vários estudos de aplicação do perceptron, indicando que as RNAs poderiam solucionar qualquer tipo de problema.

Em 1969, Minsky e Papert [61] lançaram o polêmico livro *Perceptrons*, onde foi demonstrado matematicamente, que o teorema de convergência do perceptron desenvolvido por Rosenblatt só era aplicável à classe de problemas linearmente separáveis e não era capaz de resolver problemas elementares como o OU-exclusivo. Ainda neste mesmo trabalho, Minsky e Papert colocaram que, não havia motivo aparente para acreditar que redes formadas por múltiplas camadas de perceptrons, pudessem herdar as boas qualidades do perceptron simples. Estas colocações desestimularam sobremaneira os pesquisadores que atuavam na área, fazendo com que as pesquisas sobre RNAs sofressem uma quase que completa paralização durante a década de 70. Uma das poucas atividades importantes neste período foi o surgimento das redes auto-organizáveis utilizando algoritmos competitivos. Von der Malsburg em 1973 [62] foi o primeiro pesquisador a demonstrar o princípio da auto-

organização e, juntamente com Willshaw em 1976 [63], publicou um artigo pioneiro sobre a formação das redes auto-organizáveis, motivado pelo mapeamento topológico do cérebro. Paralelamente a este trabalho Kohonen em 1977 [64] desenvolveu seus modelos de memórias associativas.

Nos anos 80, foram apresentadas algumas contribuições importantes, as quais despertaram novamente o interesse pelas RNAs. Em 1980 Grossberg [65], baseado em seus trabalhos prévios sobre o aprendizado competitivo, estabeleceu os princípios para uma nova classe de redes neurais denominadas de ART ou *Adaptive Resonance Theory*. Em 1982 Hopfield [66] utilizou a idéia de minimização da função de energia, para um enfoque no princípio de funcionamento das redes recorrentes com conexões sinápticas simétricas, tendo dado origem às *Hopfield Networks*. Em 1982 Kohonen [67] publicou seu artigo sobre os mapas auto-organizáveis, utilizando estruturas do tipo *lattice*, diferindo ligeiramente do trabalho de Willshaw e von der Malsburg. Em 1983, Kirkpatrick descreveu um novo procedimento para a otimização de redes [68], baseado nos princípios da termodinâmica estatística, denominado *Simulated Annealing*. Em 1983 Cohen e Grossberg [69] estabeleceram um princípio geral para as memórias de conteúdo endereçável e também em 1983, Barto [70] publicou um artigo sobre o controle do pêndulo invertido, baseado no aprendizado por reforço (*reinforcement learning*), inicialmente considerado por Minsky em 1954.

Um grande marco para o desenvolvimento das RNAs aparece em 1986 com desenvolvimento do algoritmo de retro-propagação (*back-propagation*) por Rumelhart, Hinton e Williams (embora este algoritmo tenha sido proposto anteriormente em 1974 por Werbos em sua tese de doutorado [71]), e com o lançamento do livro denominado *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* por Rumelhart e McClelland [72], o qual apresentava pela primeira vez de uma forma organizada todo o progresso das redes neurais até então. A partir daí, uma série de trabalhos foram efetuados, dentre eles Broomhead e Lowe em 1988 que descreveram o uso das funções de base radial (*radial basis functions*), muito usadas em problemas práticos [73].

De uma maneira geral, pode-se dizer que os trabalhos de Hopfield em 1982 e o livro de Rumelhart e McClelland em 1986, foram os grandes responsáveis pelo ressurgimento do interesse pelas RNAs.

Observando a evolução dos modelos de neurônios, estes podem ser divididos em três gerações consecutivas até os dias de hoje [1].

A primeira geração compreende os modelos que apresentam função de transferência do tipo degrau e são ativados por limiar de disparo; estes modelos são denominados de *threshold-gates*. Nestes modelos, o neurônio funciona como uma chave ligada-desligada, que tem o seu estado comutado em função do valor apresentado na entrada. O modelo característico desta geração é o *perceptron* o qual serviu e ainda serve com ponto de partida de diversos modelos que surgiram posteriormente.

Os modelos de neurônios da segunda geração apresentam como principal característica, a utilização de funções de transferência contínuas. Nestes neurônios, ao invés do comportamento como chave, tem-se uma saída contínua e proporcional ao valor de entrada do neurônio. Relembrando o neurônio biológico, que gera impulsos elétricos em resposta a estímulos, o valor contínuo da saída dos modelos de segunda geração, pode ser associado a uma frequência média dos impulsos elétricos gerados pelo neurônio. Assim, estes modelos têm maior plausibilidade biológica em relação a geração anterior, e uma capacidade de representação ampliada. O principal representante desta geração é o *perceptron* com função de transferência *sigmoidal*.

Na última década, na medida em que os estudos sobre o funcionamento do neurônio biológico foram sendo aprofundados, aumentou o questionamento sobre a precisão dos modelos utilizados até então [74][75][76][77]. Este questionamento deve-se especialmente a dois fatores: o primeiro é o tempo de resposta proporcionado pelo cérebro que é da ordem de 100 mseg ou menos para determinadas ações funcionais, o que inviabiliza a idéia de um cálculo de uma frequência média de disparo. O segundo, apoia-se em evidências de que a relação temporal entre os impulsos de excitação, tem influência determinante na resposta dos neurônios biológicos. Este aumento de interesse pelos modelos de maior plausibilidade biológica criou a denominada “Terceira Geração” de modelos neuronais, os modelos pulsados [1][2][78].

Os modelos pulsados, diferem dos modelos da primeira e segunda gerações que aqui serão denominados de clássicos, pois codificam a informação recebida através da “observação” das diferenças temporais e coincidências entre os impulsos recebidos. Nestes modelos a informação é codificada e transmitida para os demais neurônios, por intermédio de impulsos.

3.2 Algumas considerações sobre as RNAs

Apesar de biologicamente inspiradas, as RNAs na sua maioria não apresentam plausibilidade biológica, fato que ocorre com a maioria das redes utilizadas comercialmente nos dias atuais. Na verdade as RNAs procuram explorar os princípios adotados pelo cérebro, especialmente o paralelismo, na busca de implementar sistemas que possam solucionar problemas da mesma forma que este último.

Assim uma RNA é constituída por neurônios ou elementos processadores, conexões e pesos associados a estas, sendo estes últimos ponderadores das conexões. De uma forma geral, na literatura, é citado que os pesos são os responsáveis pelo armazenamento do conhecimento da rede. Esta afirmação não é errônea, porém existem arquiteturas nas quais os neurônios, por meio de sua função de ativação também armazenam parte do conhecimento da rede, como por exemplo a rede do tipo *RBF* e especialmente as redes pulsadas, partilhando esta responsabilidade com as conexões.

A construção de uma RNA envolve os seguintes passos [79]:

1. determinação das propriedades da rede: a topologia (ou conectividade), o tipo de conexões, a ordem das conexões e a faixa dinâmica dos pesos;
2. determinação das propriedades dos neurônios que compõem a rede: a função de neurônio (ou função de transferência) e a faixa dinâmica de ativação dos neurônios;
3. determinação da dinâmica do sistema: envolve o esquema de inicialização dos parâmetros, o cálculo da função de cada neurônio e a regra de aprendizado.

A seguir é apresentado um comentário sobre estas definições e suas opções mais comuns.

- **Topologia da rede**

Um exemplo de modelo de RNA é apresentado na Fig. 3.1. Este modelo é baseado em camadas, embora existam redes que não apresentem este tipo de estrutura. Do ponto de vista de topologia, a forma mais comum de rede, é aquela organizada em camadas, as quais incluem:

- a camada de entrada;
- a camada oculta (uma ou mais de uma);
- a camada de saída;

Com relação a conectividade as redes podem ser:

- propagação para adiante (*feedforward*): todas as conexões apontam em uma única direção, da entrada para a saída;
- recorrentes: quando existem conexões de realimentação.

com relação ao tipo de conexão, podem ser do tipo simétrico ou assimétrico.

Fig. 3.1 - Exemplo de modelo de uma RNA.

A ordem das conexões é determinada pelo número de saídas de neurônios, que são combinadas em uma única conexão; tipicamente todas as RNAs são de primeira ordem.

Com relação aos pesos, estes podem ser números reais ou inteiros, e podem ou não ser confinados em uma determinada faixa. Após o treinamento os valores dos pesos não são, em geral, mais alterados.

- **Funcionamento dos neurônios**

A função de cada neurônio determina a ativação do mesmo. Esta função deve ser ajustável de forma a ficar compatível com a faixa dinâmica dos dados de entrada (dependente do problema), caso contrário poderá ocorrer do neurônio nunca ser ativado, ou permanecer sempre ativado.

- **Dinâmica do sistema**

A forma de inicialização dos parâmetros, depende do tipo de rede utilizado. No caso dos pesos das conexões, na maioria das vezes esta inicialização é feita com números pseudo-aleatórios pequenos. A regra de aprendizado é uma das características mais importantes de uma RNA. É a regra de aprendizado que determina como é feita a atualização dos parâmetros durante o treinamento, de forma a otimizar o desempenho da rede. Do ponto de vista de aprendizado, as RNAs podem ser apresentadas como de **treinamento supervisionado** ou **não supervisionado**.

Entende-se por treinamento supervisionado, aquele no qual pares entrada/saída, representando a excitação e a resposta desejada (*target*) são sucessivamente apresentados à rede. O treinamento não supervisionado, por sua vez, consiste na apresentação apenas dos dados entrada para a rede, ficando a cargo desta a determinação das classes existentes.

- **Por quê utilizar as RNAs?**

Uma potencial vantagem das RNAs é **o processamento altamente paralelo** que ocorre em sua estrutura. Existe um outro aspecto porém, o qual coloca as RNAs como uma classe muito especial de sistemas; é **a capacidade de generalização** exibida pelas RNAs. Por capacidade de generalização, entende-se a capacidade de gerar respostas adequadas, para entradas que nunca foram apresentadas durante o treinamento. Estes dois aspectos fazem com as RNAs sejam capazes de solucionar problemas altamente complexos.

Contudo, é importante observar que na prática, as redes neurais artificiais não solucionam problemas sozinhas. Sua aplicação deve ser feita de forma criteriosa, decompondo o problema em questão em partes que possam ser tratadas eficientemente por uma ou várias RNAs, explorando ao máximo as habilidades de cada rede.

A utilização das RNAs, proporciona algumas características muito úteis aos sistemas que as utilizam [79], a saber:

Não linearidade

A não linearidade é uma propriedade inerente ao neurônio, sendo extremamente importante nos casos onde o sinal (ou entrada) analisado é gerado por um dispositivo não linear, como é a maioria dos sistemas na natureza.

Mapeamento entre a entrada e a saída

Um dos paradigmas de aprendizado mais utilizado é o aprendizado supervisionado, o qual promove a modificação dos pesos das conexões da rede, aplicando um conjunto de entradas com as respectivas saídas. Para cada uma destas entradas escolhida aleatoriamente, é determinada a modificação dos pesos de modo a minimizar a diferença entre a resposta atual da rede, e a resposta desejada para a respectiva entrada. Desta forma a rede aprende a partir de exemplos, construindo um mapeamento entre a entrada e a saída.

Adaptatividade

As RNAs têm a capacidade natural de adaptar os seus parâmetros, em resposta a mudanças no ambiente a sua volta, através de um retreinamento. Contudo, se o ambiente é do tipo não estacionário, a RNA pode ser construída de forma a adaptar os parâmetros de suas conexões em tempo real, em resposta às mudanças no ambiente. Um cuidado porém deve ser observado quanto a esta adaptabilidade, uma vez que pode levar a perda de robustez da rede, ou seja, a RNA pode passar a responder a estímulos (ou entradas) às quais não deveria responder, como por exemplo ruídos espúrios. Este problema é conhecido como o dilema da **estabilidade-plasticidade**.

Tratamento de informação contextual

Dependendo da arquitetura adotada, na fase de treinamento, a saída da rede é afetada pela atividade de todos os neurônios, e a correção dos parâmetros atinge potencialmente todas as conexões. Por exemplo, no caso dos pesos, este efeito faz com que a informação armazenada na rede seja influenciada por todo o conjunto de dados apresentado, considerando não apenas cada padrão apresentado mas também a correlação entre os mesmos, produzindo uma forma de informação contextual.

Tolerância a falhas

Em vários modelos de rede, a saída recebe contribuições de todos os neurônios. Caso haja falha em um neurônio, ou em um conjunto de conexões, a saída será apenas degradada. As RNAs tendem a não apresentar falhas catastróficas com a perda de alguns neurônios, ou seja, são bastante tolerantes a falhas.

Possibilidade de implementação em VLSI

Como consequência de sua natureza paralela e estrutura regular, as RNAs são ótimas candidatas para implementação em circuitos integrados do tipo *VLSI* (*Very Large Scale Integration*).

3.3 Modelos de neurônios

A seguir é feita uma breve descrição de alguns modelos de neurônios clássicos mais conhecidos.

- **O neurônio de McCulloch e Pitts**

Os modelos de neurônios tiveram início com o trabalho de McCulloch e Pitts [81], que propuseram um modelo de neurônio conforme apresentado na Fig. 3.2. Neste modelo o neurônio atua como uma unidade de decisão binária, que é ativada quando a soma ponderada de suas entradas i_i , excede um certo limiar θ (*threshold*), produzindo saídas correspondentes a zero ou um. O valor θ , pode ser interpretado como uma polarização (*bias*), correspondendo a uma entrada com valor fixo e com

sua conexão correspondente apresentando peso igual a unidade. A seguir são apresentadas as expressões para o cálculo da saída (o) e da função do neurônio (F):

$$o = F\left(\sum_N w_i i_i(t) - \theta\right) \quad (3.1)$$

$$F(x) = \begin{cases} 1 & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases}$$

McCulloch e Pitts [81], mostraram que com a escolha adequada dos pesos (w_i), um conjunto destes neurônios interligados poderia realizar cálculos computacionais, como por exemplo funções lógicas do tipo E, OU ou funções mais complexas.

Fig. 3.2 - O neurônio de McCulloch e Pitts.

- **O Perceptron**

Um dos modelos de neurônio mais utilizado atualmente como bloco básico para a construção de redes mais complexas é o do perceptron [82]. A sua arquitetura é essencialmente a mesma do neurônio de McCulloch e Pitts, apresentado na Fig. 3.2, e a grande diferença com relação ao modelo anterior é o estabelecimento de um algoritmo de aprendizado para o ajuste dos pesos [83], onde a saída é dada pela equação (3.2) e a atualização dos pesos é dada pela equação (3.3).

$$o(t) = F\left(\sum_0^{N-1} w_i i_i(t) - \theta\right) \quad (3.2)$$

$$w_i(t+1) = w_i(t) + \alpha e(t) i_i(t), \quad 0 \leq i \leq N-1 \quad (3.3)$$

onde:

$e(t)$, corresponde ao erro (no caso de Rosenblatt $e(t) = d(t) - o(t)$)

α = ganho positivo < 1 , corresponde ao coeficiente de aprendizado

Um exemplo bastante ilustrativo de como uma rede neural funciona como separador de padrões é a classificação linear feita por uma rede de *perceptrons*, que é ilustrada no exemplo que a seguir.

Exemplo:

Supondo um *perceptron* com duas entradas (i_1 e i_2), deseja-se separar duas classes distintas A e B, descritas em função de i_1 e i_2 . A equação que descreve a saída fica:

$$o(t) = w_1 i_1(t) + w_2 i_2(t) + \theta \quad (3.4)$$

a região limite de separação das classes, ocorre quando $o(t) = 0$, ou

$$w_1 i_1(t) + w_2 i_2(t) + \theta = 0 \quad (3.5)$$

ou

$$i_2(t) = -\frac{w_1}{w_2} i_1(t) - \frac{\theta}{w_2} \quad (3.6)$$

que é a equação de uma reta e a Fig. 3.3, ilustra uma possível separação destes dois padrões.

Deve ser observado que até aqui, a função do neurônio corresponde ao degrau unitário (*hard limiting*) conforme descrito originalmente por Rosenblatt. Contudo na prática, a função *sigmoidal* do tipo:

$$o(t) = \frac{1}{1 + e^{-\beta x(t)}} \quad (3.7)$$

é mais utilizada, como não linearidade do *perceptron*, porque a ativação se “suaviza”. A função da equação (3.7) é contínua e monotônica, com $0 \leq o(t) \leq 1$, para $-\infty \leq x(t) \leq +\infty$. O fator β , corresponde a inclinação da *sigmóide*, e para valores de β muito grandes, a função tende para um degrau unitário (a função tangente hiperbólica também é muito utilizada).

Fig. 3.3 - Separação linear de dois padrões no espaço, por uma rede de dois perceptrons.

Algumas vantagens que justificam a utilização da função *sigmóide*, como função de ativação do neurônio, são apresentadas a seguir.[82]

- é uma função diferenciável, que permite a implementação de algoritmos de aprendizado do tipo gradiente descendente (*gradient descent*), para redes de múltiplas camadas, conforme será visto mais adiante neste trabalho;
- é adequada para aplicações que necessitem de uma saída com valores contínuos;
- produz uma saída com valores entre 0 e 1, que pode ser interpretada como uma probabilidade estimada, dependendo do tipo de rede e algoritmo de treinamento utilizado.

- **O Adaline (*Adaptive Linear Element*)**

Outro modelo de neurônio importante é o *Adaline*, a diferença básica com relação ao *Perceptron* está no fato da função binária de saída apresentar a seguinte forma:

$$F(x) = \begin{cases} +1 & \text{se } x \geq 0 ; \\ -1 & \text{se } x < 0 \end{cases} \quad (3.8)$$

e na adoção do **erro médio quadrático** (*LMS - Least Mean Square*), como procedimento para o cálculo do erro. A equação para o cálculo do erro tem a forma dada a seguir:

$$e(t) = \left[\sum_i [o_i(t) - d_i(t)]^2 \right]^{1/2} \quad (3.9)$$

o que para neurônios simples, conduz à mesma equação para adaptação dos pesos utilizada no *perceptron*. A regra de aprendizado é conhecida como **regra de Widrow, Hoff** ou ainda **regra delta** e utiliza o método do gradiente descendente para a minimização do erro, sendo que os pesos são ajustados de forma proporcional ao negativo da derivada do erro com relação aos pesos anteriores [54][72].

- **O neurônio RBF**

Conforme descrito no trabalho de Cover, a capacidade de separar classes aumenta com a complexidade da função de separação [84]. Em Broomhead é examinada uma classe de funções que apresenta simetria radial, isto é, um caso particular de funções quadráticas que apresentem um eixo de simetria no ponto de inflexão [73]. Utilizando como função do neurônio esta classe particular de funções, pode ser contruído um neurônio cuja saída (*o*) seja proporcional à menor distância entre a entrada (*i*) e este eixo de simetria da função do neurônio no espaço, definindo assim uma região de ativação.

As regiões com mesmo valor da saída (*o*) formam assim superfícies de decisão paralelas no espaço sob a forma de elipsóides concêntricas, este é o conceito do classificador por região também chamado de *kernel classifier* [83]. Uma função de

neurônio muito utilizada para este tipo de classificador é a função Gaussiana, conforme abaixo:

$$O = e^{-\frac{\|I - P\|^2}{2\sigma^2}} \quad (3.10)$$

onde $\|I - P\|$ é a distância Euclidiana entre o vetor de entrada I e o centro da Gaussiana P , e σ corresponde a abertura da Gaussiana (ou raio da função).

3.4 A RNA do tipo Perceptron Multicamadas ou Multi Layer Perceptron

A rede do tipo *Perceptron Multicamadas* ou *Multi Layer Perceptron* (MLP) é talvez a mais utilizada comercialmente, e é utilizada nesta tese como comparativo (*benchmark*) para a validação de alguns resultados. Desta forma, é importante descrever o seu funcionamento.

Conforme verificado anteriormente, o *perceptron* simples, só é útil para a separação de classes linearmente separáveis, o que limita em muito a sua aplicação prática. A forma de contornar este problema é usar múltiplos *perceptrons* distribuídos em camadas totalmente conectadas. Na Fig. 3.4 pode ser observado um exemplo de *MLP* com três camadas totalmente conectadas, utilizando a nomenclatura criada por Cabral [85].

Fig. 3.4 - Diagrama de um *MLP* de três camadas clássico.

Esta solução somente foi viabilizada, com o desenvolvimento de um algoritmo de treinamento para redes com múltiplas camadas. O primeiro pesquisador a formalizar esta regra de aprendizado foi Werbos em 1974 [71], porém, somente em 1986 com a divulgação do trabalho de Rumelhart [72], é que o algoritmo de retropropagação ou *backpropagation* tornou-se efetivamente conhecido.

- **O algoritmo de retropropagação (*backpropagation*)**

O algoritmo de *backpropagation* consiste na propagação do erro da saída da rede para a sua entrada. Este algoritmo foi desenvolvido a partir da generalização da regra delta, citada anteriormente neste trabalho, utilizando a técnica do gradiente descendente (*gradient descent*) para efetuar a correção dos pesos em função da derivada do erro com relação aos pesos anteriores [72] [80] [82].

As expressões genéricas para a atualização dos pesos, são as seguintes:

$$w(t+1) = w(t) + \Delta w(t) \quad (3.11)$$

para neurônios pertencentes à qualquer camada:

$$\Delta w(t) = \alpha \delta(t) i(t) \quad (3.12)$$

para um neurônio de saída:

$$\delta(t) = (d(t) - o(t)) \partial f(nf) \quad (3.13)$$

para um neurônio que não pertença a saída:

$$\delta(t) = \partial f(nf) \sum_k \delta_k(t) w_k(t) \quad (3.14)$$

onde:

Δw é o valor da correção aplicada aos pesos;

$\partial f(nf)$, é a derivada da função de ativação do neurônio; e

$\delta_k(t)$ e $w_k(t)$ são respectivamente a derivada da função de erro com relação aos pesos de um neurônio da camada subsequente e o peso da ligação entre o neurônio atual e o seu subsequente.

O termo $\delta(t)$ corresponde a derivada da função de erro com relação aos pesos, sendo responsável pela propagação do erro da saída da rede para as suas camadas intermediárias, através de um cálculo recursivo.

- **O coeficiente de aprendizado α (learning rate)**

Algumas considerações devem ser feitas sobre o coeficiente de aprendizado (α). Quando é aplicada a técnica do gradiente descendente, através do algoritmo de retropropagação, está sendo calculada uma trajetória no espaço de pesos em busca de um erro mínimo. Caso o valor adotado para α seja pequeno, a trajetória calculada será suave, porém o tempo gasto para atingir-se o erro mínimo não será ótimo. Uma forma de compensar este fato é aumentar o valor de α de modo a acelerar o aprendizado da rede. Entretanto, este aumento no valor de α pode fazer com que a rede oscile e os valores dos pesos irão variar de forma abrupta, fazendo com que estes oscilem em torno do ponto de mínimo erro da rede.

Uma maneira de contornar este problema é a adoção de um termo adicional para o cálculo da atualização dos pesos denominado **momento (m)** [80]. Este termo corresponde a um fator multiplicador, que determina um percentual da correção dos pesos da interação anterior que deve ser aplicado à correção atual, fazendo com que as oscilações sejam amortecidas pelo fator de inércia dos pesos assim criado. A aplicação deste recurso, altera a expressão para correção dos pesos para a seguinte forma:

$$\Delta w(t) = \alpha \delta(t) i(t) + m \Delta w(t-1) \quad (3.15)$$

A utilização do momento porém, deve ser feita com critério, pois pode provocar o mesmo efeito de um coeficiente de aprendizado com valor muito alto.

Um exemplo de outra solução, que é muito utilizada na prática, é o cálculo adaptativo do coeficiente de aprendizado e do termo correspondente ao momento. Em Yu [86] estes são calculados de forma adaptativa durante o treinamento da rede, fazendo o uso de informações da primeira e segunda derivada da função de erro.

- **O problema do mínimo local**

Um problema que ocorre com certa frequência, durante o treinamento da rede, é a convergência para um mínimo local da superfície de erro. Algumas formas para evitar este tipo de mínimo, são apresentadas em Hertz [87], as quais são citadas brevemente a seguir:

- a) utilização do termo de momento, descrito anteriormente, fazendo com que sempre existam alterações mais adequadas nos pesos;
- b) inicialização adequada dos pesos, evitando a inicialização com valores altos, o que faz com que o sistema fique “preso” em um mínimo local próximo do início do treinamento (*flat plateau*);
- c) uma solução alternativa para o item anterior é ajustar a função de transferência dos neurônios para a faixa dinâmica dos dados de treinamento, através de uma análise prévia;
- d) adicionar ruído aos dados de treinamento, uma vez que estes mínimos em geral não são muito profundos, a correção nos pesos gerada por este ruído faz com que o sistema deixe o mínimo local. Isto pode ser conseguido através dos seguintes métodos:
 - aplicação dos dados em seqüência aleatória;
 - adicionar valores pequenos aleatoriamente aos pesos;
 - somar valores pequenos aos dados de entrada.

A quantidade de ruído adicionada, é um dado experimental, e seu uso incorreto pode causar instabilidade na rede.

- **Otimização do *MLP***

A otimização do *MLP* é um tópico complexo [82]. A seguir, são destacados os pontos considerados mais importantes:

- a) A busca do tamanho ótimo da rede: este parâmetro tem influência direta na capacidade da rede construir um modelo adequado para um dado problema. Uma maneira de determinar o tamanho adequado da rede é considerar o número de exemplos de treinamento ou a dimensionalidade dos dados de entrada. Outra forma é utilizar técnicas para o crescimento ou diminuição seletiva do número de neurônios e conexões.

- b) O tempo de treinamento da rede: uma vez determinado um tamanho adequado para a mesma, a minimização do tempo de treinamento pode ser feita através da escolha de coeficientes de aprendizado e técnicas para evitar a permanência da rede em mínimos locais durante o treinamento.
- c) A capacidade de generalização da rede: consiste basicamente na habilidade da rede produzir saídas significativas quando apresentada à padrões de entrada novos. A solução para este problema está intimamente ligada aos dois aspectos anteriores, sendo atingida através da utilização de um número suficiente de padrões de treinamento e da adequação da arquitetura da rede de forma a minimizar a representação de redundâncias e aspectos pouco significativos dos padrões representados.

3.5 A RNA auto-organizada de Kohonen

Da mesma forma que o MLP, a rede de Kohonen foi utilizada neste trabalho para estudos iniciais dos resultados obtidos com as redes pulsadas. Esta rede também serviu como ponto de partida para a organização de uma das camadas do reconhecedor, e por isso os seus princípios são descritos.

Uma rede RNA auto-organizada possui uma característica que se destaca em relação as propriedades das redes com treinamento supervisionado em geral. Os pesos sinápticos dos neurônios que a compõem se ajustam ao longo da apresentação dos dados de entrada, sem necessidade de supervisão externa, ou seja, de um treinamento cujo resultado desejado já exista *a priori*.

Os princípios da rede auto-organizada de Kohonen, também conhecida como *Self Organizing Map – SOM* -, são baseados na regra de Hebb com uma variação que considera os neurônios competitivamente. Trata-se da regra chamada *winner-takes-all*, ou seja, os pesos relativos às sinapses dos neurônios que mais respondem a um estímulo são modificados na direção dos padrões mais comuns apresentados à rede. Assim, a rede descobre por ela mesma padrões, regularidades, correlações ou categorias. Por esses motivos, a rede de Kohonen pode ser usada para criar agrupamento de padrões (*clustering*), gerar representantes das classes de dados

apresentadas para a rede (*prototyping*), reduzir a dimensionalidade dos dados de entrada (*codificação*) e mapeamento de atributos (*feature mapping*) [67][88][89].

- **Descrição da rede**

O paradigma neural *SOM*, se baseia no aprendizado competitivo e os neurônios formam um *reticulado*, cuja dimensão pode ser qualquer uma, dependendo de como ocorre o treinamento. Geralmente são utilizados reticulados neurais unidimensionais, para classificar vetores pertencentes a um espaço apresentados à rede, ou bidimensionais, formando um mapa discreto de atributos do espaço de entrada.

A cada vetor de entrada, que pode ser visto como um estímulo aplicado à rede, corresponde um nó vencedor do reticulado, que também pode ser visto como sendo um neurônio com maior resposta ao estímulo dado. Ou seja, a rede faz um papel de mapeador de um conjunto de pontos de um espaço n -dimensional num reticulado discreto, daí o nome *SOM*.

Ao longo do treinamento, os neurônios progressivamente se ajustam aos vários padrões ou classes de vetores de entrada, de maneira tal que as relações topológicas entre vetores de entrada se tornam compatíveis às relações dos neurônios que os correspondem no reticulado. Assim, as localizações dos neurônios vencedores tendem a ser organizadas em relação aos outros de tal maneira que se pode considerar válido um sistema de coordenadas sobre o reticulado [89]. Dessa forma, para diferentes atributos de entrada correspondem diferentes localizações no reticulado, daí a razão para o termo *SOFM*.

Após o treinamento, a rede é capaz de selecionar um número fixo de vetores (*prototypes*) de maneira ótima num espaço de entrada de grande dimensão e facilita a compressão de dados. Posto em comparação com métodos estatísticos voltados à classificação de padrões, como *clustering*, o mapa de Kohonen se mostra a altura em situações de baixo nível de dispersão e mais eficiente em situações de grande dispersibilidade [90].

- **Arquitetura**

Em outros modelos de rede neural, como o caso do *Multilayer Perceptron*, por exemplo, as conexões entre os neurônios são essenciais. Porém, a posição de cada

neurônio em sua estrutura não tem muita influência devido ao fato de os neurônios serem organizados em camadas bem determinadas e de cada neurônio ser ligado a todos os outros da camada posterior. No mapa auto-organizado, por outro lado, a ordem dos neurônios é importante. A localização espacial de um neurônio de saída no mapa topográfico corresponde a um domínio particular ou atributo do dado de entrada.

A estrutura da rede de Kohonen corresponde a um reticulado n-dimensional, geralmente uni ou bidimensional, de neurônios de saída, totalmente conectado às entradas, provido de uma topologia que garanta a existência de vizinhos para um determinado neurônio. Este reticulado, depois do treinamento, representa uma função discriminante (matematicamente, as variâncias ordenadas) dos sinais n-dimensionais de entrada.

A figura 3.5 ilustra a configuração de um reticulado neural, com dimensão do espaço de entrada igual a 3. De modo a que o reticulado seja considerado bidimensional, a vizinhança entre os neurônios deve ser definida convenientemente. A partir daí, pode-se considerar este como um reticulado de 3x3 neurônios ($j = 1, \dots, 9$). Ou seja, não há nada na estrutura da rede que diga qual a topologia do mapa neural, sendo esta característica determinada durante o treinamento, a qual depende do tipo de vizinhança utilizado [91].

Figura 3.5 - Configuração esquemática do *Self-Organizing Feature Map*.

- **Aprendizado**

Na rede de Kohonen, o neurônio com maior resposta à uma entrada tem os pesos de suas sinapses atualizadas de maneira a aumentar ainda mais sua resposta diante daquele padrão de entrada. Este procedimento caracteriza o treinamento competitivo da rede.

O que existe de particularidade na rede é que os pesos dos neurônios vizinhos do vencedor também são atualizados. No início do treinamento, as vizinhanças de atualização são grandes a ponto de abranger todo o reticulado, o que torna as direções das atualizações dos pesos sinápticos correlatadas em todo o reticulado. Ao longo do treinamento, a redução do tamanho das vizinhanças favorece uma seletividade entre os padrões de entrada [90] e a formação de regiões de maior atividade em torno do neurônio vencedor. Esse é o processo responsável pela formação de mapas que conservam a topologia.

Considerando o reticulado bidimensional, vetor de entrada:

$$I = [i_1 \dots i_p]^T \quad (3.16)$$

vetor sináptico do neurônio j :

$$W_j = [w_{j1} \dots w_{jp}]^T \quad \text{onde } j = 1, \dots, N \quad (3.17)$$

sendo p a dimensão dos vetores de entrada e N o número de neurônios do reticulado.

Pode-se considerar o neurônio vencedor como o que tem maior $W_j^T I$, sendo este o centro da vizinhança de atividade. Apesar de haver outras maneiras de se medir a proximidade dos vetores de entrada com os vetores sinápticos [91], adota-se como resposta do neurônio, a distância Euclidiana entre vetores. Assim, dado um vetor de entrada I , o neurônio vencedor será o de índice $n(i)$, que apresentar:

$$n(i) = \arg \left(\min_j \|i - w_j\| \right) \quad (3.18)$$

A topologia da rede, determina a vizinhança de atualização $V(n(i), r)$ de um neurônio $n(i)$ de raio r , medido segundo a métrica utilizada sobre o reticulado, que é reduzido ao longo do treinamento [80].

Dada a estrutura da rede neural e estabelecido um treinamento competitivo, a expressão que descreve como os pesos sinápticos devem ser atualizados ao longo do tempo discreto n é dada pela equação 3.19.

$$w_j(n+1) = \begin{cases} w_j(n) + \alpha(n)(i - w_j(n)), & j \in V(n(i), r(n)) \\ w_j(n), & \text{caso contrário} \end{cases} \quad (3.19)$$

4. REDES NEURAIS PULSADAS

4.1 Funcionamento do neurônio biológico

Para o entendimento das redes neurais pulsadas, faz-se necessário aprofundar os conhecimentos sobre o funcionamento dos neurônios biológicos e entender alguns conceitos básicos relacionados com o seu funcionamento de forma a se poder traçar um paralelo entre a biologia e as RNAs.

4.1.1 Estrutura dos neurônios biológicos e potencial de ação

Os neurônios biológicos podem ser divididos topologicamente, de uma forma simplificada, em três partes: os dendritos (ou árvore dendrítica) responsáveis por receber os estímulos provenientes de outros neurônios, o corpo ou soma onde os estímulos são integrados resultando nas condições para geração ou não da resposta do neurônio e o axônio que propaga a resposta para outros neurônios. A Fig. 4.1 ilustra um neurônio biológico.

A resposta de um neurônio se dá por meio de impulsos elétricos, denominados de potenciais de ação, sendo estes a chave para a propagação das informações de um neurônio para outro. A geração do potencial de ação ocorre no início do axônio, junto ao corpo do neurônio, em uma região denominada de “*axon hillock*”. A partir deste ponto o potencial de ação se propaga ao longo do axônio, até atingir os neurônios subsequentes.

Os neurônios biológicos, na ausência de estímulos em sua árvore dendrítica, apresentam um potencial de equilíbrio, o qual é estabelecido por meio de mecanismos de troca de íons entre o interior e exterior da célula, sendo de especial importância os íons de sódio e potássio. Esta troca se dá através da membrana celular por meio de processos especializados na condução de cada um destes íons para dentro e para fora da célula (bombas ativas e canais iônicos). O potencial resultante deste equilíbrio é denominado de potencial de membrana e tem aproximadamente – 50mV.

Fig. 4.1 – Estrutura anatômica do neurônio biológico.

Se o neurônio receber estímulos externos, que provoquem uma corrente elétrica através da membrana celular, o potencial de equilíbrio da membrana celular pode

variar, tanto aumentando (efeito resultante de sinapses excitatórias) como diminuindo (efeito resultante de sinapses inibitórias). Este fenômeno de alteração do potencial de equilíbrio, é denominado despolarização da membrana celular.

Caso o potencial de equilíbrio aumente acima de um determinado limiar, mecanismos de condução iônica na membrana celular são ativados, e como resultado ocorre um aumento brusco no potencial de membrana, gerando um impulso elétrico que é denominado de potencial de ação. Este impulso tem uma duração finita de alguns milisegundos e é extinto por meio do próprio aumento do potencial de membrana, que faz com que os mecanismos de restabelecimento sejam ativados conduzindo este potencial à um nível inferior ao de equilíbrio inicial, estabilizando após alguns milisegundos novamente no valor do potencial de repouso, caso o estímulo tenha sido retirado. A Fig. 4.2 ilustra o potencial de ação, permitindo observar as suas diversas fases. É importante observar que, caso o estímulo permaneça, novo potencial de ação (impulso) será gerado, porém somente após o potencial de repouso ter sido restabelecido. Este restabelecimento é denominado de período refratário e limita a frequência dos impulsos de resposta de um neurônio.

Fig. 4.2 – Potencial de ação.

4.1.2 Acoplamento entre neurônios

Outro aspecto importante a ser entendido consiste no mecanismo de acoplamento dos estímulos de um neurônio para outro. Os estímulos são recebidos através da árvore dendrítica por meio das sinapses, que são os pontos de acoplamento entre as terminações de axônios de neurônios anteriores com os dendritos do neurônios subsequentes. Nestes pontos de acoplamento são recebidos os potenciais de ação dos neurônios anteriores - potenciais pré-sinápticos - e por meio de mecanismos químicos que utilizam os assim chamados neurotransmissores, são transmitidos aos respectivos dendritos - potenciais pós-sinápticos.

Um ponto importante a ser observado é a natureza destes acoplamentos ou sinapses; estas podem ser de caráter excitatório, inibitório ou modulador. As sinapses de caráter excitatório, são aquelas que produzirão correntes elétricas que contribuirão para o aumento do potencial de repouso da membrana celular, propiciando a geração do potencial de ação. As sinapses inibitórias são aquelas que irão gerar correntes que diminuirão ainda mais o potencial de membrana, tornando-o mais negativo e dificultando a geração do potencial de ação. Finalmente, as sinapses de caráter modulador são aquelas que influenciam as correntes através do corpo celular de forma a controlar a intensidade das mesmas. Este aspectos são explorados por Liaw e Berger, na definição dos modelos de neurônios com sinapses dinâmicas [10][11].

Conforme observado anteriormente, a árvore dendrítica não apresenta potenciais de ação propagando-se mas sim correntes geradas pelos estímulos. Estas correntes elétricas circulam pelo corpo (ou soma) do neurônio receptor, e, ao passarem pela região de origem do axônio – *axon hillock*, compõem-se numa corrente resultante a qual será responsável pela eventual despolarização ou não da membrana celular e geração do potencial de ação no neurônio receptor. Um ponto importante sobre a composição de correntes é que a posição da sinapse na árvore dendrítica, faz com que a distância percorrida pelas correntes até chegar ao corpo do neurônio seja variada. Isto introduz uma componente temporal na somatória de correntes, pois o estímulo deve ser mantido por tempo suficiente para que seu efeito seja sentido na região do “*axon hillock*”.

Uma vez gerado o potencial de ação, este propaga-se através do axônio e suas derivações, até atingir as sinapses dos neurônios subseqüentes. Este mecanismo de propagação se dá por condução e por regeneração do potencial de ação em pontos bem determinados do axônio. Para minimizar as perdas durante a propagação, o axônio dispõe de um mecanismo de revestimento para diminuir sensivelmente a condução entre o meio interno e externo da membrana celular, composto de mielina (bainha mielínica). Nestes trechos a perda no sinal elétrico é mínima e de trechos em trechos do axônio, a bainha mielínica apresenta nós desmielinizados (nós de Ranvier) onde ocorre a regeneração do potencial de ação. Desta forma mesmo com várias ramificações, a forma do potencial de ação chega praticamente inalterada a cada uma das sinapses. Este mecanismo de geração e propagação do potencial de ação foi descrito por Hodgkin e Huxley através de um conjunto de equações diferenciais, conhecido como modelo de Hodgkin/Huxley, o qual permite simular computacionalmente o mecanismo de geração do potencial de ação e é a base dos modelos computacionais usados para simular as células neuronais [93].

4.2 Representação da informação nos neurônios

Até este ponto, foram estudados a estrutura do neurônio biológico e os mecanismos responsáveis pelo seu funcionamento, para gerar o potencial de ação e transmiti-lo para os neurônios subseqüentes. A resposta de um neurônio a estímulos, é feita sob a forma de uma seqüência de impulsos elétricos (ou simplesmente pulsos) e nos dias de hoje ainda não existe um consenso de qual é o tipo de representação utilizado pelos neurônios biológicos, para codificar informações. A seguir, são apresentadas algumas formas de representação.

4.2.1 A taxa de pulsos como representação

Conforme verificado na literatura, não existe um conceito único sobre o significado da taxa dos pulsos. Existem pelo menos três formas propostas de interpretação, as quais são usadas simultaneamente e muitas vezes confundidas. Excelentes discussões sobre a contagem de pulsos como representação são encontradas em [94][95].

- **taxa de contagem de pulsos**

A taxa de pulsos tem sido definida usualmente, como uma média temporal. A partir da definição de uma janela de tempo Δt , é contado o número de pulsos que ocorrem dentro desta janela. Dividindo o número de pulsos contado $n(\Delta)$, pelo comprimento da janela Δt , obtém-se a taxa média de disparo. Tradicionalmente esta medida tem sido adotada nos últimos 80 anos, como a grandeza mais importante para codificar a informação transmitida por um neurônio.

- **taxa de densidade de pulsos**

Este método é muito semelhante ao anterior. Neste caso o intervalo de tempo Δt , para a contagem dos pulso é pequeno, de forma que o tempo total de um experimento é dividido nestes pequenos intervalos e a quantidade de pulsos em cada Δt é contada. São feitas diversas repetições do mesmo experimento, sendo calculada a média de disparos em cada intervalo de tempo Δt , que resulta em um gráfico de densidade de pulsos ao longo do experimento. Obviamente esta não é a abordagem adotada pelo cérebro, que toma decisões instantâneas e com isso inviabiliza a teoria da média a partir de várias repetições.

- **taxa de atividade de um conjunto de neurônios**

Esta é uma abordagem muito semelhante a anterior, porém é considerado um grupo de neurônios para a contagem dos pulsos nos intervalos de tempo.

Nestas formas de representação da informação, está sendo desprezada qualquer informação de caráter temporal que possa estar contida na sequência de pulsos na saída do neurônio, e na verdade passa-se a considerar apenas a informação média da excitação (que tem origem na variabilidade de padrões de fase de disparo). O que contraria resultados de experimentos realizados e descritos na literatura, os quais mostram que a resposta de um neurônio varia de acordo com o padrão temporal da excitação [96].

4.2.2 A informação temporal como representação

A seguir, serão descritas algumas formas de codificação da atividade dos neurônios que consideram as informações temporais. Mais detalhes sobre estas formas de codificação, podem ser encontradas na literatura [94][97].

- **instante do primeiro pulso**

Supondo que um neurônio receba um estímulo do tipo degrau unitário no instante t_0 . Este neurônio irá apresentar uma resposta, no decorrer de um intervalo de tempo Δt . Pode-se propor uma codificação onde o intervalo de tempo para o início da resposta do neurônio, varie em função da intensidade do estímulo. Desta forma uma resposta $\Delta t_1 < \Delta t_2$, daria a indicação de que a excitação que gerou a resposta em Δt_1 tempo é mais intensa do que aquela que gerou uma resposta em Δt_2 .

- **codificação em fase**

Este método é semelhante ao anterior, porém em vez de utilizar um único evento como referência temporal para a medida do tempo de resposta, utiliza-se um sinal periódico de referência.

- **correlação e sincronismo de atividade**

Neste caso, pulsos de outros neurônios podem ser usados com referência temporal. Por exemplo a sincronia ou a diferença temporal entre as respostas de um par de neurônios, pode ser usada para indicar a ocorrência de algum estímulo específico. Esta abordagem permite com que sejam montadas redes de neurônios com capacidade para a detecção de padrões utilizando um detector de coincidências.

A Fig.4.3 ilustra estas possibilidades de codificação temporal. Um exemplo de como as informações de padrões temporais podem ser codificadas por meio de pulsos, é apresentada no artigo de Hopfield [98]. Neste artigo é sugerido que atrasos temporais nas respostas obtidas a um sinal de sondagem (no caso eco acústico do sonar da coruja) podem ser detectados por uma rede de neurônios e serem utilizados para determinar o padrão de objeto diante do animal. Outro exemplo de interesse, que ilustra a detecção de padrões temporais por meio da sincronização de populações de neurônios, é apresentado por Singer [99], onde é demonstrada a forma pela qual os cérebros de mamíferos desenvolvem as representações do ambiente a sua volta.

Fig.4.3 – Exemplo das possíveis codificações temporais.

Devido a diversidade de representações possíveis, conclui-se que a utilização de modelos de neurônios que utilizem as informações de contagem de pulsos em conjunto com a informação temporal, certamente irá proporcionar uma nova classe de modelos de neurônios muito mais poderosos computacionalmente.

4.3 Modelos de neurônios pulsados

A partir da descrição feita sobre o funcionamento do neurônio biológico, modelos que descrevam os neurônios considerando o fator temporal, podem ser construídos em diferentes níveis de abstração. Do ponto de vista microscópico, existe um grande número de canais iônicos, que são ativados e desativados dependendo do potencial de membrana e da presença moléculas denominadas de neurotransmissores. Os modelos que tentam descrever estes processos, levam em consideração aspectos espaciais e da fisiologia do neurônio, e são os denominados de modelos baseados em condutância e os modelos em compartimentos.

Em um segundo nível de abstração, deixando de considerar a estrutura espacial e o funcionamento preciso dos mecanismos iônicos, surgem os modelos denominados de “limiar de disparo”. Nestes modelos o neurônio é considerado como uma unidade que gera seqüências de pulsos, caso a excitação atinja um determinado limiar. A seguir é apresentada uma descrição geral destes modelos.

4.3.1 Modelos baseados em condutância e compartimentos

Os modelos baseados em condutância e compartimentos buscam descrever em detalhes o funcionamento dos neurônios, concentrando-se na geração e na transmissão do potencial de ação pelo axônio. Estes modelos, tiveram origem nos resultados de extensos estudos com neurônio da lula gigante, realizados por Hodgkin-Huxley e que resultaram em um modelo descrito por quatro equações diferenciais [93].

A primeira equação descreve a conservação de carga elétrica em um trecho da membrana, sob a influência de algumas correntes e é mostrada na equação (4.1).

$$C \frac{du}{dt} = - \sum_k I_k + I(t) \quad (4.1)$$

Na equação (4.1) u é a tensão, $I(t)$ é uma corrente externa e $\sum_k I_k$ representa a somatória de correntes iônicas através da membrana celular. No caso do modelo de Hodgkin-Huxley, existem três tipos de correntes iônicas causadas pelos canais de sódio, potássio e um canal de descarga (*leakage*) que são identificadas por Na, K e L. A equação 4.2 apresenta esta somatória.

$$\sum_k I_k = g_{Na} m^3 h (u - V_{Na}) + g_K n^4 (u - V_K) + g_L (u - V_L) \quad (4.2)$$

Os parâmetros g são constantes empíricas que determinam a máxima condutância de cada tipo de canal. As variáveis m , n e h são descritas pelas três equações diferenciais restantes [93], não apresentadas aqui, e controlam a remoção do bloqueio dos canais Na e K. O canal L é independente de controle por tensão e apresenta um condutância g_L . Os parâmetros V_{Na} , V_K e V_L , são constantes empíricas relativas aos canais.

A Fig. 4.4 representa as propriedades da membrana, por meio de C e R , e os canais iônicos mencionados nas equações. Devido as concentrações de iônicas serem diferentes no interior e exterior da célula, mesmo na ausência de estímulos externos, surge um potencial e a membrana funciona com um capacitor que é carregado por uma bateria. Se uma corrente I for injetada na célula, provocará a carga do capacitor e fluirá pelos canais iônicos.

Fig. 4.4 – Diagrama esquemático do modelo de Hodgkin-Huxley.

Quando o potencial de membrana aumenta o suficiente, os canais iônicos abrem-se primeiro Na e depois K, fazendo com que o efeito seja um pulso conforme mostrado na Fig. 4.2, um pico positivo devido a súbita abertura e fechamento dos canais de sódio rápidos, seguida de um pico negativo devido aos canais K que são mais lentos.

O modelo de Hodgkin-Huxley, descrito até aqui, não inclui nenhuma informação espacial da estrutura do neurônio. O neurônio é considerado como um elemento pontual e o principal objetivo do modelo é a geração dos pulsos de potencial de ação. Os modelos de compartimentos procuram prover uma descrição mais detalhada da dinâmica neuronal, levando em consideração aspectos espaciais da estrutura da árvore dendrítica, modelando assim a transmissão sináptica com maior detalhe. O modelo de compartimentos, é derivado do modelo de Hodgkin-Huxley, onde são acrescentados mais alguns canais iônicos e cada trecho da árvore dendrítica é modelado por um circuito equivalente ao da Fig. 4.4, que são conectados por meio de uma resistência longitudinal r .

4.3.2 Modelos de limiar de disparo

Estes modelos tem o seu funcionamento inspirado no fato de que um pulso ocorre na saída do neurônio, quando o potencial de membrana excede o valor de um limiar. Nestes modelos a variável que descreve o estado do neurônio, corresponde ao potencial de membrana. Dois modelos deste tipo são amplamente utilizados na literatura: o “modelo de integração e disparo” (*integrate and fire model*) e o “modelo de resposta impulsiva” (*spike response model*).

- **modelo de integração e disparo**

A idéia básica deste modelo consiste de um capacitor C em paralelo com um resistor R , sendo estes alimentados por uma corrente $I(t)$, conforme indicado na Fig. 4.5.

A corrente de entrada pode ser dividida em duas componentes, I_R que corresponde a parcela que passa pelo resistor linear R e I_C que corresponde a corrente de carga do capacitor C . A equação (4.3), apresenta estas componentes em função de $I(t)$.

$$I(t) = \frac{u(t)}{R} + C \frac{du}{dt} \quad (4.3)$$

Fig. 4.5 – Conceito do modelo de integração e disparo.

Multiplicando a equação (4.3) por R , e introduzindo a constante de tempo $\tau_m = RC$, é obtida a equação (4.4), que define o modelo básico de integração e disparo de Stein [100].

$$\tau_m \frac{du}{dt} = -u(t) + RI(t) \quad (4.4)$$

Onde, u é o potencial de membrana e τ_m refere-se a constante de tempo da membrana do neurônio. No modelo de integração e disparo, o potencial de ação (pulso de saída do neurônio), não é descrito explicitamente. Os pulsos são na verdade eventos caracterizados pelo instante de disparo do neurônio $t^{(f)}$, que é o momento em que $u(t)$ excede a tensão de limiar v . Imediatamente após $t^{(f)}$, o potencial de membrana é reduzido para um novo valor $u_r < v$. Para $t > t^{(f)}$ a dinâmica é dada novamente pela equação 4.4. Pela sua simplicidade e analogia direta com circuitos elétricos, o modelo de integração e disparo têm sido o modelo escolhido para a implementação em hardware e em circuitos integrados.

- **modelo de resposta impulsiva**

O modelo de resposta impulsiva, foi introduzido por Gerstner em 1991 [101] e uma boa revisão deste modelo pode ser encontrada em Gerstner [97].

Da mesma maneira que no modelo de integração e disparo, o estado interno de um neurônio j , é descrito por uma única variável $u_j(t)$, que corresponde ao potencial de membrana do neurônio j no instante t . Os pulsos de saída são gerados, quando

$u_j(t)$ ultrapassa um determinado limiar v e o momento em que isto ocorre, pode ser usado para definir formalmente os instantes de disparo $t_j^{(f)} \in \mathfrak{R}^+$ do neurônio j . A Fig. 4.6 ilustra a conexão entre dois neurônios e o detalhe temporal do potencial de membrana, e do potencial pós-sináptico.

Fig. 4.6 – (A) mostra o processo de transmissão de um pulso entre dois neurônios; (B) mostra o pulso do potencial de ação, que é gerado sempre que $u_j(t)$ excede o limiar v_j , seguido do período refratário dado por $\eta(t - \hat{t}_j)$; (C) mostra o potencial pós-sináptico que resulta do pulso gerado $\epsilon_{ij}(t - \hat{t}_j)$ ponderado pelo peso w_{ij} , sendo a linha contínua para o caso de um potencial excitatório (EPSP – *excitatory postsynaptic potential*) e pontilhada para um potencial inibitório (IPSP – *inhibitory postsynaptic potential*).

No modelo de resposta impulsiva, a dinâmica do potencial de membrana é descrita por uma função $\eta_j(s)$ onde $s = t - \hat{t}_j$ é o período de tempo desde o tempo em que o

limiar v_j foi excedido ou o instante de tempo do último disparo do neurônio j . A duração do pulso emitido pelo neurônio δ_{abs} , não carrega nenhuma informação e por isso a forma do potencial de ação durante o disparo não é relevante, permitindo considerá-lo um impulso simples. Um ponto importante neste modelo, é que logo após a geração de um potencial de ação, a emissão de um novo pulso é impossível. Este período é denominado de “período refratário absoluto”, sendo modelado por $\eta_j(s)$ com valor $-\infty$ para o intervalo $s \leq \delta_{abs}$. Após esse intervalo, o potencial de membrana segue a forma $\eta(s) = -v_j \exp(-(s - \delta_{abs})/\tau_{ref})$, conforme mostra a Fig. 4.6(B).

Esta informação, gerada pelo neurônio j no instante $\hat{t}_j$, é transmitida para neurônios subseqüentes conforme mostrado na Fig.4.6(A). Neste modelo, os detalhes do processo de transmissão são desprezados e a modelagem concentra-se no efeito que o pulso gerado em j tem sobre o potencial de membrana do neurônio i . As formas típicas do potencial pós-sináptico são apresentadas na Fig.4.6(C). Deve ser observado que para $s \leq d_{ij}$, a função que modela o potencial pós-sináptico apresenta valor nulo ($\varepsilon_{ij}(s) = 0$), o que corresponde ao atraso de transmissão entre o neurônio j e o neurônio i . Para $s > d_{ij}$ a função de resposta corresponde aquela descrita na equação (4.5), onde τ_p corresponde ao instante de pico da função.

$$\varepsilon_{ij}(s) = xe^{1-x} \quad \text{onde} \quad x = \frac{s - d_{ij}}{\tau_p} \quad (4.5)$$

A análise feita até aqui, ficou restrita a dois neurônios interligados. Na realidade cada neurônio pós-sináptico i recebe na entrada, pulsos provenientes de diversos neurônios $j \in \Gamma_i$, onde Γ_i corresponde ao conjunto de todos os neurônios pré-sinápticos ao neurônio i . Para incluir o efeito dos demais neurônios, é considerado que o potencial de membrana $u_i(t)$ é uma superposição linear de todas as contribuições na entrada do neurônio i , conforme mostra a equação 4.6.

$$u_i(t) = \sum_{t_i \in F_i} \eta_i(t - t_i) + \sum_{j \in \Gamma_i} \sum_{t_j \in F_j} w_{ij} \varepsilon_{ij}(t - t_j) \quad (4.6)$$

Na equação 4.6, o peso w_{ij} determina a amplitude da resposta e corresponde a eficiência da sinapse que conecta o neurônio j ao neurônio i ; e F_i representa o conjunto de todos os instantes de disparo do neurônio.

A equação (4.6) também pode ser formulada em termos de um limiar dinâmico $v_i(t)$, para isso considera-se uma condição limiar inicial $u_i(t) = v_i$, obtendo-se a equação (4.7).

$$\sum_{j \in \Gamma_i} \sum_{t_j \in F_j} w_{ij} \varepsilon_{ij}(t - t_j) = v_i - \sum_{t_i \in F_i} \eta_i(t - t_i) \quad (4.7)$$

O lado direito da equação (4.7), pode ser considerado como um limiar dinâmico $v_i(t)$ que aumenta após cada disparo do neurônio, e que decresce lentamente até o valor assintótico v_i caso não ocorra mais nenhum disparo do neurônio i .

Adotando que a resposta é única para qualquer par de neurônios, exceto pelo fator w_{ij} , implica que os índices ij podem ser omitidos escrevendo simplesmente $\varepsilon(t - t_j)$.

Existe ainda uma variação do modelo de resposta impulsiva, que é útil para simplificar sua análise. Esta simplificação consiste em considerar que apenas o último disparo do neurônio contribui para o comportamento durante o período refratário. Desta forma a equação (4.6) pode ser simplificada, para refletir apenas a influência do disparo mais recente na soma das η_i contribuições, através da substituição:

$$\sum_{t_i \in F_i} \eta_i(t - t_i) \rightarrow \eta_i(t - \hat{t}_i)$$

onde $\hat{t}_i$ reflete o último disparo do neurônio i . A esta simplificação é dado o nome de memória de curto prazo (*short term memory*).

Para permitir uma visualização do modelo de resposta impulsiva em um diagrama em blocos, as funções de base de η_i e ε_{ij} , serão consideradas filtros simples. Com isso assume-se que o comportamento refratário do neurônio é modelado conforme a equação (4.8).

$$\eta_i(s) = -\eta_0 \exp\left(-\frac{s}{\tau_i}\right) H(s) \quad (4.8)$$

Nesta equação, $s = (t - t_i)$ e τ é a constante de tempo de decaimento. H representa a função degrau de Heavyside e a constante η_0 equivale a amplitude da refratariedade relativa. Da mesma forma, o comportamento da resposta aos pulsos pré-sinápticos é modelado conforme a equação (4.9).

$$\varepsilon_{ij}(s) = w_{ij} \exp\left(-\frac{s}{\tau_{ij}}\right) H(s) \quad (4.9)$$

Onde τ_{ij} é outra constante de decaimento e w_{ij} são as intensidades das conexões pré-sinápticas. Substituindo as equações (4.8) e (4.9) na equação (4.7), e considerando o efeito de todos os instantes de disparo do neurônio, as parcelas descritas pelas equações (4.10) e (4.11) podem ser interpretadas como integradores com perdas (*leaky integrator*).

$$\sum_{t_j \in F_i} \exp\left(-\frac{s}{\tau_{ij}}\right) H(s) \quad (4.10)$$

$$-\eta_0 \sum_{t_i \in F_i} \exp\left(-\frac{s}{\tau_i}\right) H(s) \quad (4.11)$$

Na equação (4.10), os pulsos provenientes do neurônio j são acumulados, sendo decaídos de acordo com o tempo de chegada e com a constante de decaimento τ_{ij} . Da mesma forma, a equação (4.11) acumula os pulsos de saída e decai os mesmos no tempo de acordo com a constante τ_i , provendo um controle de limiar de disparo dinâmico. A Fig. 4.7 ilustra o modelo de tempo contínuo obtido.

Fig. 4.7 – Modelo de integração e disparo representado pela equação (4.7).

Apesar dos modelos de integração e disparo e o modelo de resposta impulsiva parecerem semelhantes, apresentam abordagens ligeiramente diferentes quanto a generalização. Estas diferenças foram consideradas, para a adoção do modelo utilizado nesta tese. No modelo de integração e disparo, os parâmetros são dependentes de tensão e no modelo de resposta impulsiva são dependentes do tempo desde o último disparo. Outra diferença entre estes modelos refere-se a formulação das equações. Enquanto que no modelo de integração e disparo a descrição do modelo é feita por equações diferenciais, o modelo de resposta impulsiva é descrito pelo potencial de membrana no ponto t como uma integração dos valores deste potencial no passado.

4.3.3 Modelos de taxa de disparo (rate models)

Este modelo é mencionado apenas para manter a análise dos modelos sem omissões significativas. Corresponde ao modelo formal utilizado nas RNAs denominadas de clássicas e o fator temporal está presente de forma implícita, no cálculo das taxas de disparo dos neurônios. A atividade neural é descrita em termos de valores médios da taxa de disparo dos neurônios, acumulados ao longo de um intervalo de tempo. A taxa de disparo x_i de um neurônio i corresponde a uma variável analógica que apresenta uma dependência não linear da excitação u_i do neurônio. A excitação u_i é dada pela soma linear de todas as conexões de entrada, de neurônios que enviam pulsos para o neurônio i , conforme mostrado na equação (4.12).

$$x_i = \sigma \left(\sum_j w_{ij} x_j \right) \quad (4.12)$$

O parâmetro w_{ij} corresponde ao peso atribuído para a conexão do neurônio j para o neurônio i . A função $\sigma(\cdot)$ modela a não linearidade da unidade, sendo usualmente a função sigmoideal ou tangente hiperbólica, conforme apresentado no capítulo 3. A equação (4.12) é o ponto de partida para a teoria padrão de RNAs [102] [92].

4.4 O modelo de resposta impulsiva discreto

Uma vez que para a análise do sinal de voz, é importante fazer um processamento temporal do sinal, o modelo de resposta impulsiva foi escolhido por apresentar uma implementação direta através de procedimentos computacionais [103]. Este modelo de neurônio será usado como base para a implementação das diversas redes utilizadas nesta tese, visando compor um sistema de reconhecimento automático do locutor.

Para sinais digitais, tanto o tempo como a amplitude são discretos e portanto um modelo discreto precisa ser construído. O ponto de partida para o modelo discreto é a

equação (4.6). Os integradores presentes no modelo contínuo são substituídos por filtros digitais recursivos de primeira ordem. O tempo contínuo é dividido em intervalos de duração constante T , e a cada intervalo um novo estado do neurônio é calculado, baseado no estado interno e das entradas, no intervalo anterior.

O resultado destas considerações é apresentado na equação (4.13). Esta é uma equação recursiva e a cada passo é efetuado o cálculo do novo potencial de membrana do neurônio u_i .

$$u_i(n) = -\{\eta_0 y(n-1) + r_i u_{ti}(n-1) + v\} + \sum_{j \in \Gamma_i} \{w_{ij} x_{ij}(n) + r_{ij} u_{fij}(n-1)\} \quad (4.13)$$

onde:

u_i é o potencial de membrana atual do neurônio i

x_{ij} é a excitação de entrada do neurônio i proveniente do neurônio j

w_{ij} corresponde a eficiência da conexão do neurônio j para o neurônio i

y é o potencial de ação na saída do neurônio

r_i e r_{ij} são dados pela equação (4.14), com fator de relaxação constante τ_i e τ_{ij} respectivamente;

u_{fij} corresponde a saída de cada filtro de entrada do neurônio (conexões provenientes de outros neurônios) e u_{ti} representa a saída do filtro no ramo de controle do limiar de disparo.

$$r = \exp\left(-\frac{T}{\tau}\right) \quad (4.14)$$

O período T deve ser escolhido de tal maneira que o comportamento do neurônio tenha tempo suficiente para ser estabilizado, sem gastar recursos computacionais com passos de cálculo desnecessários. Em simulações buscando representações em tempo real, um valor normalmente usado para o intervalo de tempo é de 1ms. Este intervalo representa aproximadamente a duração do potencial de ação, portanto a

simulação pode ser considerada em tempo real se o intervalo de tempo usado for de 1ms ou menos.

De forma a minimizar os requisitos computacionais, o modelo pode ser simplificado, assumindo que os diversos ramos de entrada x_{ij} (conexões provenientes dos neurônios pré-sinápticos) decaiam com a mesma constante τ_{ij} . Neste caso a somatória de todas as entradas pode ser feita antes da etapa de integração e todas as entradas passam a usar um único filtro.

A seguir é apresentado um algoritmo para o cálculo do estado do neurônio, ao longo dos intervalos de tempo.

- a) fase de entrada: executar a função de entrada para todos os neurônios

$$pi_{ij}(n) = w_{ij} \cdot x_{ij}(n), \quad po_i(n) = \eta_0 \cdot y(n)$$

- b) fase de filtragem: executar a função de filtro para todos os neurônios

$$u_{fij}(n) = pi_{ij}(n) + r_{ij} \cdot u_{fij}(n-1)$$

$$u_{ii}(n) = po_i(n) + r_i \cdot u_{ii}(n-1)$$

- c) fase de saída: executar a função de saída para todos os neurônios

- calcular o potencial de membrana $u_i(n)$:

$$u_i(n) = -\{u_{ii}(n) + v\} + \sum_{j \in \Gamma_i} u_{fij}(n)$$

- emitir um pulso, se $u_i(n)$ for positivo

- d) fase de aprendizado: ajustar os pesos de acordo com a regra de aprendizado adotada.

5. EXPERIMENTOS PRELIMINARES

5.1 Metodologia

A metodologia utilizada para o desenvolvimento desta pesquisa consistiu em dividir o estudo e os testes de forma incremental partindo de um neurônio individual, para posteriormente propor os diversos arranjos de neurônios que foram testados como estágios do sistema de reconhecimento independente do texto.

Inicialmente foi efetuada a preparação da base de dados de locutores a ser utilizada. Esta preparação consistiu na escolha do conjunto de locutores, geração das características do sinal de voz (*features*) a serem utilizados no reconhecimento e validação dos conjuntos de treinamento e teste escolhidos.

Em seguida foi procedida a implementação do modelo escolhido de neurônio pulsado, com a realização de testes de resposta a seqüências de estímulo pré-determinadas, para a avaliação dos diversos parâmetros que fazem parte do modelo de neurônio. Após os testes com um único neurônio, foram realizados testes de avaliação do comportamento de diversos neurônios interligados.

Passou-se então para a etapa de avaliação do funcionamento dos arranjos de neurônios, para compor os estágios do sistema reconhecimento do locutor. Inicialmente foram realizados testes dependentes do texto, como forma de avaliar a representação obtida pela rede pulsada, na tarefa de conversão das características do sinal de voz em uma seqüência de pulsos.

Após estes testes, passou-se para os experimentos na forma independente do texto. Nesta etapa foi desenvolvida uma segunda camada de neurônios pulsados auto-organizável, com base nos resultados obtidos no reconhecimento dependente do texto. Foram utilizadas, como características do sinal de voz, a energia de um banco de filtros na escala Mel e os coeficientes *Mel-Cepstrais*, que são representações de conhecida eficiência nos sistemas de reconhecimento de voz. É importante observar que nesta tese não houve a preocupação em avaliar a extração de características do sinal de voz, e sim a avaliação da capacidade do reconhecedor em manter as informações contidas no mesmo e permitir o reconhecimento do locutor. Como

parâmetro de comparação do percentual de reconhecimento obtido com a utilização dos estágios de neurônios pulsados, foi utilizado o resultado de reconhecimento obtido a partir de um classificador neural clássico o MLP.

5.2 Base de dados utilizada

A base de dados utilizada neste trabalho é denominada *Speaker Recognition v1.0*, de propriedade do *Center for Spoken Language Understanding – CSLU* – do *Oregon Graduate Institute* nos E.U.A.. Esta base de dados foi escolhida, pois a mesma é amplamente utilizada pela área acadêmica nos trabalhos relacionados com a área de voz, e para que se tenha um parâmetro internacional permitindo a comparação com trabalhos correlatos.

Esta base de dados é composta por locutores de diversas faixas etárias e ambos os sexos. As amostras de voz foram coletadas a partir de gravações sobre linha telefônica digital, em um único canal (mono), com 16 bits por amostra e frequência de amostragem de 8KHz. Os arquivos desta base, foram fornecidos no formato RIFF e, para utilização nesta tese, foram convertidos para o formato texto.

Cada locutor, participou de 12 sessões de gravação, distribuídas ao longo de 2 anos, com intervalos variados entre as sessões. As locuções proferidas em cada sessão incluíram: respostas à perguntas previamente preparadas, seqüências numéricas cuja repetição foi solicitada, frases pré-selecionadas e frases sugerindo fala espontânea de forma a resultar em locuções com conteúdo livre. Nos experimentos, foram utilizadas as locuções com frases pré-selecionadas, as quais são apresentadas a seguir:

frase 01 – *stop each car if it's little*

frase 02 – *play in the street up ahead*

frase 03 – *a fifth wheel caught speeding*

frase 04 – *it's been about two years since davey kept shotguns*

frase 05 – *charlie did you think to measure the tree*

frase 06 – *tina got cued to make a quicker escape*

frase 07 – *joe books very few judges*

frase 08 – *here I was in miami and illinois*

Foram escolhidos aleatoriamente 10 locutores com idade entre 21-30 anos, Foram selecionadas 5 sessões do conjunto de frases por locutor, resultando em 40 locuções por indivíduo. Adicionalmente foram selecionadas mais 5 sessões para a frase 08, totalizando 100 locuções para esta frase, que foram utilizadas nos testes dependentes do texto. Foram escolhidas as sessões de gravação que se apresentavam completas, com todas as frases do conjunto, pois foi constatado que muitas sessões de gravação apresentavam problemas como falta de todo o conjunto de locuções, truncamento da locução ou sessões com dois locutores misturados (por exemplo, voz masculina para uma frase e feminina para outras). Com relação a frase 08, esta frase foi escolhida em função do seu conteúdo nasal, que tem sido comprovado de grande utilidade para o reconhecimento automático do locutor.

Para os testes de reconhecimento na forma independente do texto, as locuções foram agrupadas em dois conjuntos, denominados de conjunto 1 e conjunto 2. O conjunto1, compreende as frases 01-02-03-04 e o conjunto2 as frases 05-06-07-08.

pré-processamento

A etapa de pré-processamento, foi baseada nos resultados obtidos em Timoszczuk [48], tendo sido executados os seguintes passos:

- a) edição manual dos pontos terminais da locução (*endpoints*), de forma a eliminar os trechos de silêncio no início e no final de cada locução;
- b) divisão das locuções em segmentos com superposição entre segmentos consecutivos. A Tab.5.1 apresenta as combinações de número de amostras por segmento e deslocamento entre segmentos consecutivos utilizadas.
- c) pré-ênfase do sinal de voz, de forma a melhorar a relação sinal-ruído, através de um filtro de primeira ordem (vide equação (5.1), com $\alpha = 0,94$);
- d) submissão de cada um dos segmentos a uma janela de *Hamming*, definida pela equação (5.2), de forma a minimizar a introdução de componentes

frequenciais indesejadas geradas pelas transições abruptas nos extremos de cada segmento;

segmento		deslocamento	
amostras	tempo(ms)	amostras	tempo(ms)
256	32	128	16
512	64	32	4
128	16	32	4

Tab. 5.1 – Combinações de comprimento de segmento e deslocamento entre segmentos consecutivos que foram utilizados.

$$H(z) = 1 - \alpha z^{-1} \quad (5.1)$$

$$W_H(n) = 0.54 - 0.46 \cos\left(\frac{2n\pi}{N-1}\right) \quad (5.2)$$

onde: n é o número do segmento atual, e
 N é o número total de segmentos.

extração dos features

Os *features* escolhidos foram os coeficientes cepstrais (cepstrum real) e a energia na saída de um banco de filtros segundo a escala Mel. Para os coeficientes cepstrais, foram calculados os 16 primeiros coeficientes Mel-cepstrais. A escolha destes coeficientes é justificada em função de sua robustez, conforme exposto no capítulo 2. Para o cálculo da energia em cada segmento, foi utilizado um banco de 20 filtros distribuído segundo a escala perceptual Mel, sendo o mesmo conjunto de filtros utilizado para o cálculo dos coeficientes Mel-cepstrais (MFCCs), o que resultou em 20 coeficientes de energia.

A sequência adotada para o cálculo dos MFCCs é aquela descrita em Reynolds [104], utilizando a DCT (*Discrete Cosine Transform*), e apresentada na Fig. 5.1. A

filtragem foi efetuada por um banco de 20 filtros, igualmente espaçados, segundo a escala *Mel*.

Fig. 5.1 - Seqüência de geração dos MFCCs.

Uma nova representação derivada dos coeficientes MFCC foi desenvolvida, tendo em vista que os coeficientes MFCCs apresentam variações pequenas, especialmente nos coeficientes de ordem mais alta obtidos pelo método da Fig. 5.1. Esta variação pode passar sem ser percebida pela rede pulsada, e para isso foi montada uma representação codificada por meio de pulsos, onde cada pulso representa um ponto de máximo (ponto de inflexão) do coeficiente MelCepstral correspondente. De forma análoga, a mesma representação foi implementada para a energia dos filtros Mel.

5.3 Testes com um neurônio

Nestes testes é verificado o comportamento do modelo de neurônio discreto, com a variação dos seus parâmetros internos e do sinal de estímulo apresentado em sua entrada. O modelo de neurônio discreto implementado, corresponde ao modelo de resposta impulsiva discreto apresentado no capítulo 4, possuindo uma única entrada.

Por questões de ordem prática e visando facilitar a explicação dos testes, foi adotada uma nova nomenclatura para os parâmetros do neurônio conforme segue:

- N_0 – constante que determina a amplitude do período refratário;
- V_t – constante de limiar de disparo estático – *threshold* estático;
- W_i – peso da conexão de entrada;

- T_i – fator de decaimento do filtro de entrada, correspondendo a $\frac{1}{\tau_{ij}}$ no modelo;
- T_o – fator de decaimento do filtro de saída, correspondendo a $\frac{1}{\tau_i}$ no modelo;
- V_{abs} – indica o número de iterações em que um novo disparo não é permitido.

As variáveis de observação do funcionamento do neurônio, seguem a seguinte nomenclatura:

- V_{entr} – estímulo na entrada do neurônio - potencial pré-sináptico;
- V_{sinap} – estímulo após o processamento da sinapse – potencial pós-sináptico;
- V_{refr} – limiar de disparo – somatória do limiar estático com o filtro de saída;
- V_{memb} – estado interno do neurônio – potencial de membrana;
- V_{spike} – saída do neurônio – potencial de ação gerado (disparo).

A implementação do modelo de neurônio discreto foi feita em MatLab e o código utilizado nos testes é apresentado no Apêndice I.

5.3.1 Resposta a um pulso único

Neste teste é avaliada a resposta do neurônio a um único impulso, sendo estudada a influência dos parâmetros no seu funcionamento. Os parâmetros utilizados neste teste foram: $N_o=1$, $V_t=0.1$, $W_i=1$, $T_i=1$, $T_o=1$. No instante de tempo 10, é apresentado um pulso na entrada do neurônio. A resposta do neurônio é apresentada na Fig. 5.2, onde pode ser observado que pelo fato do neurônio partir do repouso, o potencial de membrana acompanha o potencial pós-sináptico durante a geração do potencial de ação. Após o potencial de ação ser gerado na saída no neurônio, o potencial de membrana acompanha o potencial dinâmico do limiar de disparo.

Fig. 5.2 – Resposta do neurônio a um pulso único na entrada ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 1$, $T_o = 1$).

parâmetros N_o e V_t

Os parâmetros N_o e V_t influenciam a amplitude do potencial de limiar de disparo dinâmico do neurônio. O parâmetro N_o regula a intensidade do valor inicial do limiar dinâmico e com isso determina a sensibilidade do neurônio a novos estímulos na entrada, após um disparo. V_t controla o nível de repouso do potencial de membrana e com isso determina a sensibilidade de disparo do neurônio, a partir do estado de repouso. No teste de resposta a um pulso único, estes parâmetros serão mantidos constantes.

parâmetro W_i

Este parâmetro está relacionado com a eficácia da conexão sináptica, sendo chamado usualmente de peso da conexão e determina a contribuição do pulso de entrada para a geração do potencial de membrana. Uma outra maneira de interpretar o parâmetro W_i é como um controle de sensibilidade do neurônio com relação a entrada. Quanto menor este parâmetro, menos sensível o neurônio será com relação ao estímulo presente na entrada correspondente. Na Fig. 5.2, o parâmetro W_i é positivo e com isso propaga o estímulo para o interior do neurônio. Na Fig. 5.3, é apresentada a

resposta do neurônio para a mesma excitação da Fig. 5.2 com o parâmetro W_i negativo, onde pode ser observado que o potencial pós-sináptico apresenta-se negativo. Isto faz com que o potencial de membrana varie de forma negativa e não atinja o limiar de disparo. Como consequência o neurônio não apresenta potencial de ação na saída.

O fato do parâmetro W_i de dificultar ou facilitar o disparo do neurônio permite que as conexões possam ser classificadas como inibitórias ou excitatórias. Esta característica pode ser explorada em um neurônio com múltiplas entradas, permitindo o controle do disparo através da combinação de entradas.

Fig. 5.3 – Resposta do neurônio a um pulso de entrada, com conexão de caráter inibitório ($N_o = 1$, $V_t = 0.1$, $W_i = -1$, $T_i = 1$, $T_o = 1$).

parâmetro T_i

O parâmetro T_i regula a duração do potencial pós-sináptico. Neste teste, T_i foi variado e o parâmetro T_o foi mantido constante. Na Fig. 5.2, foi apresentada a resposta do neurônio quando $T_i = T_o$, situação na qual o potencial pós-sináptico e o período refratário apresentaram a mesma duração. A seguir são apresentadas as respostas para três casos: $T_i < T_o$, $T_i > T_o$ e a condição limite de quase segundo disparo $T_i = 0.5 T_o$.

caso 1: $T_i < T_o$

Nesta situação, ilustrada pela Fig. 5.4, o potencial pós-sináptico decai mais lentamente do que a recuperação do potencial de membrana. Assim, quando o potencial de membrana retorna para o nível de repouso, ainda existe um “resíduo” do potencial pós-sináptico que possibilita novos disparos do neurônio. Esta situação se repete, até que o nível “residual” do potencial pós-sináptico não seja mais suficiente para ocasionar novos disparos. Um detalhe que deve ser observado, é o incremento negativo do potencial de membrana a cada disparo sucessivo, fazendo com que o disparo subsequente ocorra após intervalos de tempo cada vez mais longos. Este comportamento faz com que ocorra uma modulação na frequência dos pulsos de saída. Esta modulação, permite que os pulsos de saída de um neurônio possam ser modulados por um sinal analógico, gerando uma codificação em frequência do sinal de entrada.

Fig. 5.4 – Resposta do neurônio para um pulso com $T_i < T_o$ ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 0.1$, $T_o = 1$).

caso 2: $T_i > T_o$

Quando esta relação entre os parâmetros T_i e T_o é observada, o potencial pós-sináptico decai rapidamente, antes que o potencial de membrana retorne para o nível de repouso.

Fig. 5.5 - Resposta do neurônio para um pulso com $T_i > T_o$ ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 10$, $T_o = 1$).

Desta forma, ocorre um único disparo, conforme é apresentado na Fig. 5.5. Em neurônios com múltiplas entradas, esta condição em conjunto com a utilização adequada do parâmetro W_i , permite configurar um detector de coincidência.

caso 3: $T_i = 0.5 T_o$

Esta é uma condição denominada de limite, em que quase ocorre um segundo disparo, sendo útil apenas para ilustrar o comportamento do potencial de membrana. Na Fig. 5.6, pode ser observado como a remanência do potencial pós-sináptico afeta o potencial de membrana, mas não consegue fazer com que este último ultrapasse o limiar de disparo.

Fig. 5.6 – Resposta do neurônio para um pulso de entrada, na condição limiar $T_i = 0.5 T_o$ ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 0.5$, $T_o = 1$).

parâmetro T_o

De forma semelhante ao parâmetro T_i , T_o controla a duração do período refratário e provoca respostas do neurônio equivalentes às apresentadas na análise do parâmetro T_i . O aspecto a ser destacado com relação ao parâmetro T_o pode ser observado na Fig. 5.7, onde é apresentada a resposta do neurônio para a condição em que o valor de T_o é reduzido. Apesar da resposta do neurônio ser a mesma que aquela apresentada na Fig. 5.5 ($T_i > T_o$), observa-se o aumento na duração do período refratário, o que faz com que aumente a intensidade do estímulo necessário para provocar novos disparos.

Estas combinações entre os parâmetros T_i e T_o sugerem que os neurônios podem ser implementados com sensibilidades diferentes à estímulos e com resposta do tipo pulso único ou trem de pulsos.

Fig. 5.7 - Resposta do neurônio para um pulso com $T_i > T_o$, mostrando o aumento de duração do período refratário ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 1$, $T_o = 0.1$).

5.3.2 Resposta a uma seqüência de pulsos

Neste item são apresentados os resultados obtidos, quando o neurônio recebe uma seqüência de estímulos sob a forma de um trem de pulsos, com duração finita.

parâmetro Vabs e a resposta a uma seqüência de pulsos

O parâmetro Vabs não foi mencionado até o momento, pois os testes de resposta a um pulso único são influenciados apenas pelas temporizações dos filtros de entrada e saída. Para entender a necessidade deste parâmetro, a resposta do neurônio deve ser avaliada para um trem de pulsos. Na Fig. 5.8, é apresentada a resposta de um neurônio onde $V_{abs}=0$. Na situação apresentada, pode ser observado que o potencial de membrana aumentou seu valor, viabilizando o disparo permanente do neurônio e este apresenta sua saída permanentemente em nível ativo. Esta condição impede que seja gerada uma resposta ao estímulo sob a forma de um trem de pulsos com duração constante, passando a modular a largura do pulso de saída. É importante notar que nos neurônios biológicos, não ocorre a modulação da largura do potencial de ação e

sim a modulação da frequência dos pulsos. Assim o parâmetro V_{abs} , permite controlar o tipo de modulação dos pulsos de saída desejado.

Fig. 5.8 - Resposta do neurônio para uma sequência de pulsos com $V_{abs}=0$, mostrando a modulação da largura dos pulsos de saída ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 0.1$, $T_o = 1$).

A resposta do neurônio, resultante da alteração do parâmetro $V_{abs}=1$, é apresentada na Fig. 5.9. Pode ser observado que os pulsos gerados pelo neurônio têm a mesma duração, apresentando modulação em frequência na medida em que o potencial pós-sináptico diminui. O período refratário absoluto (V_{abs}), permite obter do modelo de resposta ao impulso, um comportamento semelhante ao observado nos neurônios biológicos.

O efeito da modulação em frequência dos pulsos de saída pode ser controlado por meio da variação dos parâmetros T_i e T_o , proporcionando diferentes respostas para uma mesma sequência de estímulos. A Fig. 5.10 apresenta esta variação na resposta, para $T_i > T_o$.

Fig. 5.9 - Resposta do neurônio para uma sequência de pulsos com $V_{abs}=1$, quando ocorre a modulação da frequência dos pulsos de saída ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 0.1$, $T_o = 1$).

Fig. 5.10 - Resposta do neurônio para uma sequência de pulsos para $V_{abs}=1$ e com $T_i > T_o$ ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 1$, $T_o = 0.1$).

Caso a sequência de estímulos, seja trocada por uma sequência de pulsos aleatória, a característica de resposta do neurônio é mantida e a resposta do neurônio será prolongada de forma proporcional a duração da sequência de estímulos.

Do ponto de vista das frequências apresentadas pelos pulsos de saída, a Fig. 5.9 e Fig.5.10 sugerem uma forma intuitiva de classificar os neurônios:

- neurônios de resposta rápida, caracterizados por um período refratário curto e uma resposta com o formato de um trem de pulsos de frequência máxima (limitados apenas pelo período refratário absoluto);
- neurônios de resposta lenta, caracterizados por um período refratário longo e uma resposta de frequência (pulsos mais espaçados).

resposta a uma sequência de amplitude variável

Os estímulos do neurônio testado até aqui foram sempre considerados pulsos de amplitude unitária. Como o objetivo é processar características do sinal de voz, e estas características tem a forma de uma sequência discreta com amplitude variável, é conveniente verificar o comportamento da resposta do neurônio para este tipo de sequência. A Fig. 5.11, apresenta a resposta do neurônio para uma sequência de amplitude variável. Neste caso o neurônio tem período refratário longo, e funciona como um codificador onde a frequência dos pulsos na saída, é proporcional a amplitude do sinal de entrada. A situação de período refratário rápido produz uma resposta equivalente a apresentada na Fig.5.9 e por isso não será repetida aqui.

Através destes testes, pôde ser observada a influência individual de cada um dos parâmetros que controlam a resposta do modelo implementado.

Fig. 5.11 – Resposta do neurônio com período refratário longo, para uma sequência de entrada de amplitude variável, resultando na modulação da frequência dos pulsos de saída.

5.4 Testes de comportamento coletivo

A partir deste ponto, passa a ser analisado o comportamento coletivo dos neurônios. Como estímulo dos neurônios, foram usados os coeficientes *Mel-Cepstrais* e a representação sob a forma de pulsos derivada dos coeficientes *Mel-Cepstrais*. O objetivo é visualizar que tipo de padrão de pulsos é apresentado na saída da rede e a influência dos parâmetros sobre a saída.

5.4.1 Neurônios sem realimentação

O arranjo de neurônios utilizado nestes testes é formado por 16 neurônios dispostos em uma única camada. O objetivo desta camada é atuar como codificador das sequências de entrada, transformando-as em uma sequência de pulsos, para ser processada pelas camadas seguintes do reconhecedor.

A Fig. 5.12 apresenta a resposta para a sequência de pulsos gerada a partir dos coeficientes MFCC. Cada asterisco no gráfico, corresponde ao instante em que um

neurônio disparou, permitindo observar comportamento de cada neurônio ao longo da apresentação do estímulo de entrada.

Fig. 5.12 – Comportamento do arranjo de neurônios, quando estimulado pela seqüência de pulsos obtidos a partir dos MFCCs ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 1$, $T_o = 0.1$, $V_{abs}=1$).

Observando a Fig. 5.12, é verificado que cada neurônio dispara segundo o estímulo recebido, sem qualquer sincronização com os demais neurônios. Certamente características dos estímulos estão presentes, porém o objetivo é utilizar a seqüência de saída desta rede em estágios posteriores do reconhecedor.

Na Fig. 5.13 onde pode ser observada a resposta da rede para a sequencia de MFCCs (valores do coeficientes). Neste caso ocorre uma redução nos disparos, e cabe observar que apenas os neurônios que dispararam são representados na figura, por isso o eixo dos neurônios apresenta valores de 1 a 11. A variação nas freqüências de disparo e o não disparo de alguns neurônios deve-se a diferença de amplitude entre os diversos coeficientes.

Fig. 5.13 - Comportamento do arranjo de neurônios, para a mesma seqüência de MFCCs como estímulo e mesmos parâmetros da Fig. 5.12.

5.4.2 Neurônios com realimentação

Neste teste, foi utilizada a mesma rede do experimento sem realimentação, sendo que cada um dos neurônios foi dotado de múltiplas entradas. Estas entradas adicionais, recebem os sinais de saída dos demais neurônios que compõem a rede. A auto-realimentação do neurônio não foi implementada, pois a mesma já ocorre através do controle do período refratário.

A conexão entre dois neurônios é única, e dotada de um peso (W_i) próprio, que foi inicializado de forma pseudo-aleatória. Foi permitido que os pesos assumissem valores positivos e negativos, resultando em conexões do tipo excitatório e inibitório. O esperado com estas conexões é que a rede manifeste um padrão próprio de sincronismo para cada tipo seqüência de estímulo. Os testes correspondentes a condição sem conexões de realimentação foram repetidos, e os resultados são apresentados nas figuras Fig. 5.14 e Fig. 5.15.

Fig. 5.14 – Comportamento do arranjo de neurônios com realimentação, para a mesma seqüência de estímulo da Fig. 5.12 ($N_o = 1$, $V_t = 0.1$, $W_i = 1$, $T_i = 1$, $T_o = 0.1$, $V_{abs} = 1$).

Fig. 5.15 – Comportamento do arranjo de neurônios com realimentação, para a mesma seqüência de estímulo da Fig. 5.13.

Foram realizados testes adicionais, onde em lugar de uma conexão única de realimentação entre um neurônio e outro, foram incluídas duas conexões. Uma destas conexões necessariamente possui peso positivo, sendo de caráter excitatório, e a outra conexão apresenta peso negativo, sendo portanto inibitória. Os testes realizados com esta configuração produziram resultados equivalentes aos apresentados pelas Fig. 5.14 e Fig. 5.15. A razão para não ter ocorrido diferença na resposta deve-se ao fato de que o efeito destas conexões, ao serem somadas na entrada do neurônio,

equivale a situação de conexões pesos pseudo-aleatórios (soma de valores pseudo-aleatórios).

5.4.3 Variação dos parâmetros de tempo

Outro ponto que deve ser considerado é que os parâmetros de tempo da rede foram mantidos fixos até o momento, o que pode ter prejudicado a manifestação de padrões de disparo. Por este motivo foi feito um teste extenso de variação dos parâmetros de tempo, para verificar o efeito sobre a resposta da rede. Este teste foi feito apenas com a seqüências de pulsos obtidas a partir dos MFCCs, uma vez que os coeficientes não apresentaram bons resultados até aqui.

Foi feita a diminuição de T_i , fazendo com que a rede apresentasse resposta refratária rápida. O resultado apresentou a ocorrência de disparo sucessivo de todos os neurônios com diferenças no instante em que os pulsos cessam, após o término da seqüência de entrada e pode ser observado na Fig. 5.16. Esta diferença no tempo com que os pulsos cessam, deve-se ao valor máximo assumido pelo potencial pós-sináptico. Quanto maior o valor deste potencial, maior é a duração do trem de pulsos de saída.

Em função do disparo contínuo de todos os neurônios, e baseando-se no comportamento observado no neurônio simples decidiu-se por alterar o limiar de disparo estático de forma a diminuir a sensibilidade de disparo e com isso tentar obter um padrão de sincronismo. Dentre vários resultados obtidos, é apresentado na Fig. 5.17 o resultado da redução do parâmetro de limiar V_t , que foi alterado para 1.5. Esta alteração fez com que surgissem respostas dos neurônios moduladas em freqüência, sem nenhum padrão de sincronismo.

Outra variação experimentada consistiu em alterar a relação entre T_i e T_o , de forma que T_i seja muito maior do que T_o . Neste caso em lugar de efetuar a integração da seqüência de estímulo de entrada somente no estágio de entrada, o potencial de membrana também passa a atuar como fator de integração temporal. O resultado deste teste é apresentado na Fig. 5.18, onde pode ser observado o surgimento de disparos conjuntos, tendendo a apresentar sincronismo. Esta última representação será explorada nos testes de reconhecimento na forma dependente do texto.

Fig. 5.16 – Comportamento do arranjo de neurônios com resposta refratária rápida.

Fig. 5.17 – Comportamento do arranjo de neurônios com resposta refratária rápida, tendo o limiar de disparo aumentado.

neurônios

Fig. 5.18 – Comportamento do arranjo de neurônios com resposta refratária lenta, combinado com integração da entrada ($T_i=1$, $T_o=0.03$).

6. EXPERIMENTOS DE RECONHECIMENTO DO LOCUTOR

Neste capítulo são apresentados os testes visando o reconhecimento automático do locutor. O primeiro teste é feito no modo dependente do texto e complementa os testes de comportamento coletivo da rede de neurônios, fornecendo subsídios para a implementação do reconhecedor no modo independente do texto.

6.1 Reconhecimento no modo dependente do texto

Para os testes no modo dependente do texto, foi utilizada uma base de dados composta por 10 locutores, que proferiram a frase 08 dez vezes em diferentes sessões de gravação. A partir destas locuções foram extraídos as características do sinal de voz utilizadas nos testes, a saber:

- 16 coeficientes *Mel-Cepstrais*;
- energia de um banco de filtros, com 20 filtros segundo a escala *Mel*;
- sequência de pulsos gerada a partir dos coeficientes MFCC;
- sequência de pulsos gerada a partir da energia do banco de filtros.

6.1.1 Comportamento inter e intra locutor

Com a rede configurada conforme a parametrização da Fig. 5.18 no teste de comportamento coletivo apresentado no capítulo 5, foi realizado um teste para a análise da saída da rede pulsada para um mesmo locutor e para os diversos locutores.

Inicialmente foi tomada uma repetição de cada locutor e geradas as seqüências de saída da rede para cada caso, teste inter locutor. As seqüências de saída obtidas têm o aspecto semelhante ao da Fig 5.18 e o detalhamento destas seqüência para cada um dos locutores, é apresentado no Apêndice II. A Fig. A2.1. apresenta as seqüências de saída para cada locutor, considerando como entrada a seqüência de pulsos geradas a partir dos MFCCs. Em seguida foi realizada esta mesma comparação tendo como seqüência de entrada os coeficientes MFCC, e o resultado é apresentado na Fig. A2.2 do Apêndice II. Foi observado que apesar do aumento do número de pulsos no

padrão de disparo, a sequência de pulsos derivada dos MFCCs, produz um padrão mais regular na saída da rede.

Em seguida o teste foi repetido considerando todas as repetições de um mesmo locutor, teste intra-locutor. Os resultados são apresentados na Fig. A2.3 do Apêndice II, para a sequência de entrada na forma de pulsos derivados dos MFCCs, e na Fig. A2.4 do Apêndice II, para a sequência de entrada na forma dos coeficientes MFCC.

Observando os resultados obtidos, não é possível concluir se irá ser conseguida a separação dos locutores. Então foi utilizada uma ferramenta, para avaliar a estatística das sequências geradas na saída, para que as mesmas ou algum vetor derivado delas possa ser utilizado como sequência de entrada em um classificador. Pela sua plausibilidade biológica e pela suas propriedades de preservação da topologia do espaço de entrada, optou-se por uma rede auto organizável do tipo SOM – *Self Organizing Map* – descrita no capítulo 3.

6.1.2 Análise com mapa auto-organizável

A rede auto-organizável SOM foi aplicada pela propriedade que tem de aprender a estatística das sequências de entrada que à ela são apresentadas. O objetivo é verificar se cada um dos locutores apresenta uma estatística própria, e com isso produzir padrões representativos de cada classe, para o treinamento de um classificador.

Para os testes iniciais, foi configurada uma rede SOM composta por um mapa de 16x16 neurônios, sendo que cada neurônio recebe todas as 16 saídas da rede pulsada. Assim, a rede aprende a distribuição estatística das combinações de disparo dos neurônios apresentadas na saída da rede pulsada.

Para verificar o funcionamento da rede, foi efetuado um teste utilizando uma sequência gerada a partir de ruído branco, a qual deve fazer com que a rede SOM apresente uma distribuição uniforme no seu mapa final de pesos. Observando a amplitude da distribuição final dos pesos após o treinamento, foi verificado que a mesma apresenta-se com uma distribuição uniforme conforme mostrado na Fig. 6.1, o que valida o aprendizado da rede.

Fig. 6.1 – Distribuição da amplitude dos pesos, após o treinamento da rede SOM com seqüências de ruído branco.

Após a validação da rede SOM, foi realizado o treinamento de uma rede para cada locutor, sendo obtidos portanto 10 mapas estatísticos ao final do treinamento. Durante o treinamento, apenas as seqüências pertencentes a um dado locutor foram apresentadas durante o treinamento, de forma a gerar um mapa individualizado.

A Fig. 6.2 apresenta dois exemplos dos mapas obtidos, que apresentam a distribuição do valor final dos pesos ao longo de cada um dos 256 neurônios do mapa, sendo 16 pesos para cada neurônio. O resultado com os mapas de todos os locutores é apresentado no Apêndice III, para evitar o carregamento desnecessário do texto.

Fig. 6.2 – Mapa de distribuição dos pesos obtido após o treinamento; mapa (A) corresponde ao locutor 01 e o mapa (B) corresponde ao locutor 02.

Analisando os mapas de todos os locutores, pode ser notado que a distribuição final das amplitudes dos pesos no mapa de cada locutor foi diferente, o que permite pressupor a existência de uma distribuição estatística diferente para cada locutor.

A análise dos mapas estatísticos obtidos, pode ser feita através do traçado de trajetórias dos locutores na rede SOM. A análise de trajetórias, consiste em apresentar as locuções de cada locutor para a sua respectiva rede, e observar qual é o neurônio que apresenta a maior amplitude saída. A partir desta observação feita para cada segmento da locução, tem-se as coordenadas de localização dos neurônios vencedores no mapa SOM e a respectiva amplitude de saída. Com estes dados pode ser traçado um gráfico tridimensional, representando a evolução do neurônio vencedor no mapa SOM, que foi denominado de trajetória.

Para realizar o traçado das trajetórias, foi aumentada a dimensão da rede SOM, para melhorar a resolução da estatística aprendida pela rede. O mapa foi alterado para um arranjo de 100x100 neurônios, compreendendo um total de 10.000 neurônios. Estes mapas foram treinados, respeitando-se a divisão feita na base de locuções entre os conjuntos de treino e teste.

Após o treinamento dos mapas, as seqüências de saída da rede pulsada pelas locuções foram apresentadas e as coordenadas do neurônio vencedor de cada mapa foi gravada a cada segmento apresentado, gerando as trajetórias no espaço tridimensional. A Fig 6.3 apresenta um exemplo de dois locutores que apresentaram trajetórias completamente diferentes.

As trajetórias para cada locutor foram calculadas para cada tipo de vetor de entrada. Examinando as trajetórias obtidas, foi observado que aquelas trajetórias geradas a partir dos coeficientes de energia dos filtros são visualmente melhores candidatas para aplicação ao classificador e serão usadas como entradas do classificador. As trajetórias obtidas para todos os locutores são apresentadas no Apêndice IV.

Fig. 6.3 – Exemplos de trajetórias tridimensionais de dois locutores na rede SOM; (A) corresponde ao locutor 01 e (B) corresponde ao locutor 05.

6.1.3 Classificador

Conforme mencionado na análise dos gráficos de trajetórias, os vetores de entrada a serem usados no classificador serão as trajetórias geradas a partir das seqüências de energia dos filtros Mel. Neste teste, foi escolhido como classificador clássico a rede do tipo perceptron de multiplas camadas ou MLP – *Multi Layer Perceptron* – cujo funcionamento foi exposto no capítulo 3.

Para que este classificador possa ser utilizado é necessário que todos os vetores de entrada do MLP tenham o mesmo tamanho, o que não ocorre uma vez que as locuções não foram normalizadas em comprimento, desde o início do processo. Optou-se então por utilizar o algoritmo LTW – *Linear Time Warping* – que efetua a normalização pela média do comprimento dos vetores e apresenta um bom compromisso entre robustez e eficiência na implementação. O comprimento final dos vetores de *features* após a normalização foi fixado em 100 segmentos. Assim os vetores de entrada do classificador MLP são as trajetórias normalizadas com comprimento de 100 pontos com 3 elementos por ponto (3 coordenadas espaciais).

Foi feita uma verificação visual nos mapas das trajetórias após a normalização, para verificar se foram introduzidas distorções significativas. Não foi constatada nenhuma distorção grave e as trajetórias tiveram suas características preservadas.

A configuração do classificador MLP utilizado nos testes é a seguinte:

- 300 neurônios na camada de entrada (100×3);
- 100 neurônios na camada oculta; e
- 10 neurônios na camada de saída, correspondendo a um neurônio para cada classe de locutor.

A classificação é feita tomando-se como parâmetro o valor de saída apresentado pelos neurônios da camada de saída, sendo declarado vencedor aquele que apresentar o maior valor de amplitude em sua saída.

6.1.4 Diagrama em blocos do reconhecedor e resultados

Uma vez determinados todos os blocos que compõem o reconhecedor, foram feitos os testes finais para verificar o seu funcionamento. A Fig. 6.4 apresenta o diagrama em blocos, do reconhecedor utilizado neste experimento.

O treinamento do classificador de saída, foi feito apresentando os conjuntos de trajetórias de treinamento (70 trajetórias) até que o erro do MLP convergisse para 0.01 ou então ocorressem 100 épocas de treinamento. Cada época de treinamento corresponde a 1000 apresentações dos vetores de entrada de forma aleatória. Após o treinamento foi realizado o teste, com um sub-conjunto do próprio conjunto de treino, consistindo de uma locução de cada locutor (10 locuções no total), e os resultados são apresentados na Tab. 6.1.

Fig. 6.4 – Diagrama em blocos do reconhecedor dependente do texto.

Saídas correspondentes a cada locutor no reconhecedor

Locutor	1	2	3	4	5	6	7	8	9	10
1	0,7620	0,0248	-0,0148	-0,0390	0,0287	-0,0378	0,0606	0,1269	0,0415	0,0062
2	0,0697	0,7554	0,0157	-0,0284	0,0247	-0,0115	0,0670	-0,0507	-0,0451	0,0462
3	-0,0379	0,0462	0,7318	0,0341	-0,0395	0,0979	-0,0448	0,0449	0,1071	-0,0385
4	-0,0322	0,0004	0,0126	0,8276	-0,0203	0,0008	-0,0367	-0,0196	0,0964	-0,0450
5	0,0441	0,0422	-0,0447	-0,0235	0,7839	0,0287	0,0256	-0,0716	-0,0568	0,0766
6	-0,0368	0,0420	0,1234	0,0144	0,0470	0,7519	0,0904	0,0757	-0,0359	-0,0686
7	0,0911	0,0617	-0,0291	-0,0301	0,0066	0,0699	0,6859	0,0752	-0,0030	-0,0113
8	0,0536	-0,0027	0,0306	-0,0292	-0,0724	0,0230	0,1193	0,6959	0,0430	0,0622
9	-0,0046	-0,0525	0,0923	0,0994	-0,0619	-0,0477	-0,0191	0,0107	0,7322	0,0155
10	0,0227	0,0474	-0,0433	-0,0476	0,0595	-0,0776	0,0185	0,0126	0,0156	0,8132

teste com conjunto de treino original

testes 10
 acertos 10
 erros 0
 taxa 100,00%

Tab. 6.1 – Resultados do classificador para testes com o próprio conjunto de treino.

A Tab. 6.1 apresenta na coluna identificada como “Locutor” o número do locutor sob teste, apresentado para a identificação. As colunas numeradas de 1 até 10, correspondem cada uma a um locutor e indicam o valor da saída do classificador, para cada locutor de teste apresentado. O número do locutor identificado corresponde a saída que apresentar o maior valor (identificada em negrito). Conforme pode ser observado, a identificação foi de 100%.

Em seguida, foi apresentado o conjunto de teste onde se verifica uma taxa de acerto de 90% (total de 30 vetores – 3 para cada locutor), e o resultado é apresentado na Tab 6.2. A representação é a mesma utilizada na Tab 6.1, e pode ser observado que ocorreram 3 identificações erradas

Saídas correspondentes a cada locutor no reconhecedor

Locutor	1	2	3	4	5	6	7	8	9	10
1	0,4909	0,0439	-0,0448	-0,0397	0,0559	0,0248	0,2634	0,0395	0,0195	0,0034
1	0,4227	0,0395	-0,0562	-0,0422	0,0832	0,0156	0,4506	0,0015	0,0190	0,0085
1	0,7095	0,0495	-0,0102	-0,0390	0,0042	-0,0444	0,0505	0,1614	0,0311	0,0240
2	-0,0152	0,6414	0,0030	-0,0653	0,0252	-0,0317	0,1192	0,2314	-0,0020	0,0015
2	0,0139	0,6842	0,0197	-0,0539	0,0645	-0,0338	0,0737	0,1694	-0,0219	-0,0246
2	-0,0021	0,6529	0,0209	-0,0385	0,0797	-0,0295	0,0359	0,1876	-0,0355	-0,0144
3	-0,0034	0,0130	0,8058	-0,0276	-0,0070	-0,0370	-0,0170	0,1186	-0,0390	-0,0333
3	-0,0017	0,0144	0,8010	-0,0237	-0,0022	-0,0363	-0,0192	0,1234	-0,0385	-0,0355
3	-0,0231	-0,0043	0,7659	-0,0434	-0,0280	-0,0215	-0,0069	0,1998	-0,0224	-0,0277
4	0,0028	-0,0535	-0,0381	0,2697	-0,0371	0,1807	-0,0426	0,0002	0,5574	0,0228
4	0,1098	-0,0283	-0,0103	0,6630	0,0354	-0,0017	-0,0535	-0,0614	0,1394	0,0331
4	-0,0031	-0,0702	-0,0440	0,5236	-0,0388	0,2052	-0,0523	0,0076	0,2530	0,0458
5	0,0061	0,1137	-0,0052	0,0494	0,5672	0,0384	0,0861	-0,0030	-0,0693	0,0406
5	0,0702	0,1064	0,0085	0,0299	0,6726	0,1210	0,0887	-0,0336	-0,0632	-0,0423
5	0,0332	0,0743	-0,0007	-0,0114	0,5856	0,1332	0,1700	-0,0154	-0,0454	-0,0379
6	0,0370	-0,0303	-0,0412	-0,0334	0,0019	0,5489	0,1580	0,0230	0,0237	0,1430
6	0,1312	-0,0142	-0,0318	-0,0548	0,0009	0,4890	0,2313	-0,0099	0,0714	0,0639
6	-0,0232	-0,0174	-0,0329	0,0069	0,0290	0,3313	0,2331	0,0236	-0,0182	0,3681
7	0,0137	0,0495	-0,0128	-0,0563	0,1098	0,1144	0,6240	-0,0058	0,0131	0,0604
7	-0,0057	0,0340	-0,0287	-0,0562	0,1489	0,1515	0,4111	0,0453	0,0023	0,0048
7	-0,0310	0,0432	-0,0258	-0,0502	0,0698	0,0458	0,6168	0,0313	-0,0009	0,1711
8	-0,0219	0,3638	0,0904	-0,0412	0,0096	-0,0468	0,0190	0,4093	-0,0258	-0,0059
8	-0,0027	0,2463	0,2508	-0,0383	0,0149	-0,0489	0,0139	0,3648	-0,0272	-0,0311
8	-0,0368	0,1887	0,0979	-0,0573	-0,0298	-0,0205	0,0162	0,6035	0,0007	-0,0157
9	0,0523	-0,0598	-0,0517	0,1616	-0,0751	0,0166	-0,0222	0,0277	0,6225	0,0711
9	0,0086	-0,0541	-0,0502	0,1123	-0,0711	0,0469	-0,0047	0,0187	0,7677	0,0104
9	0,0700	-0,0475	-0,0391	0,1909	-0,0647	0,0642	-0,0153	0,0045	0,6261	0,0311
10	0,0255	-0,0110	-0,0278	-0,0488	0,2869	0,0820	0,2565	-0,0370	0,0145	0,3323
10	0,0255	-0,0110	-0,0278	-0,0488	0,2869	0,0820	0,2565	-0,0370	0,0145	0,3323
10	0,0255	-0,0110	-0,0278	-0,0488	0,2869	0,0820	0,2565	-0,0370	0,0145	0,3323

teste com conjunto de teste original

testes 30

acertos 27

erros 3

taxa 90,00%

Tab. 6.2 – Resultados do classificador para o conjunto de testes.

6.2 Reconhecimento no modo independente do texto

A seguir são descritas as redes implementadas e os experimentos realizados, para a implementação do reconhecedor. Em função dos resultados obtidos no teste de reconhecimento dependente do texto, foi decidido adotar uma arquitetura em camadas para o reconhecedor, cada qual com uma função específica.

A Fig. 6.5, apresenta um diagrama em blocos do reconhecedor, utilizado nos experimentos que são descritos a seguir.

Fig. 6.5 – Diagrama em blocos do reconhecedor com redes pulsadas.

6.2.1 Base de dados

A base de dados para os testes de reconhecimento de forma independente do texto foi montada com todas as locuções disponíveis dos 10 locutores. Desta forma a base compreendeu 8 frases por locutor, com 5 repetições para cada uma das frases, totalizando 40 locuções por locutor.

Os parâmetros representativos das características do sinal voz, escolhidos para este teste, foram:

- 16 coeficientes *Mel-Cepstrais*, escolhidos por serem parâmetros de reconhecida robustez e estabilidade, para serem usados como *benchmark*;
- energia de um banco de 20 filtros distribuídos segundo a escala Mel, pois este processo de extração de parâmetros aproxima a forma com que o ouvido humano extrai as características do sinal de voz.

Inicialmente, estes parâmetros foram extraídos das locuções por meio de segmentos de 256 pontos (32mseg), com deslocamento 128 (16mseg) pontos entre segmentos consecutivos.

Uma vez composto o banco de dados e escolhidos os parâmetros de representação, foi feita a divisão aleatória do conjunto de locuções, em dois agrupamentos correspondendo aos conjuntos de treino e de teste. O conjunto de treinamento, foi denominado de conjunto 1, sendo composto pelas frases 01-02-03-04. O conjunto de teste, foi denominado de conjunto 2, sendo composto pelas frases 05-06-07-08.

De forma a evitar o desbalanceamento entre os conjuntos de treinamento e de teste, foram feitos testes preliminares para verificar o equilíbrio dos conjuntos. Caso exista um desbalanceamento muito grande entre os conjuntos, as taxas de reconhecimento na inversão dos conjuntos treino/teste podem ser muito diferentes, como a obtida no experimento de reconhecimento dependente do texto ao ser feita a inversão dos conjuntos.

O método utilizado para fazer esta verificação foi utilizar um classificador neural clássico – no caso o *Multilayer Perceptron* – MLP, em conjunto com os coeficientes MFCC e verificar qual a taxa de acerto no reconhecimento. A utilização deste classificador seguiu o método das MTIs que é descrito em Timoszczuk [48]. O vetor de entrada do MLP foi composto por um único segmento a cada iteração (MTI10), de forma a não influenciar a taxa de reconhecimento com o uso dos padrões temporais usados pela MTI (pior caso). O MLP foi treinado por um total de 100 épocas, sendo cada época composta por 24.000 iterações em um teste e 26.000 em outro. O número de iterações corresponde ao número de vetores de treinamento utilizados. A rede utilizada possui 16 neurônios na camada de entrada, 100 neurônios na camada oculta

e 10 neurônios na camada de saída, correspondentes a cada um dos locutores a ser identificado. Os resultados obtidos neste teste são apresentados na Tab.6.3.

treino	teste	número de acertos	número de erros	taxa de reconhecimento
conjunto1	conjunto2	186	14	93%
conjunto2	conjunto1	192	8	96%

Tab. 6.3 – Resultado de verificação do balanceamento da base de dados.

Foi feito o mesmo teste utilizando a energia dos filtros na escala MEL, e os resultados obtidos são apresentados na Tab. 6.4.

treino	teste	número de acertos	número de erros	taxa de reconhecimento
conjunto1	conjunto2	158	42	79%
conjunto2	conjunto1	163	37	81.5%

Tab. 6.4 – Resultado de verificação dos conjuntos com classificador MLP para energia dos filtros.

Como os resultados com a inversão dos conjuntos foram próximos, concluiu-se que os conjuntos estão balanceados e são adequados para o desenvolvimento dos trabalhos.

6.2.2 Camada1 do reconhecedor

A camada 1 é a responsável por converter as seqüências de entrada em uma seqüência de pulsos, que serão processadas pelas camadas subseqüentes do reconhecedor.

Para a implementação desta camada, foi utilizada a estratégia de um conjunto de neurônios pulsados, conforme descrito no estudo de comportamento coletivo. Estes neurônios não fazem uso de realimentações entre os mesmos, de forma que apenas capturem a sequência de entrada e a transformem em um trem de pulsos. Esta forma de operação se assemelha bastante a conversão que é feita pela cóclea.

Esta camada foi implementada por meio de um programa de computador, em linguagem C, cujo código é apresentado no Apêndice VI. Este programa permite controlar os parâmetros de funcionamento dos neurônios, através de um arquivo de configuração, cujo formato é apresentado no trecho de código a seguir.

Arquivo ConfigC1

[DIM] - dimensão do vetor de entrada - por conseguinte da rede

[INTEGRACAO] - número de entradas integradas por neurônio

[N_NEURON_POR_ENTRADA] - número de neurônios da rede

[OFFSET_DE_ENTRADA] - offset aplicado a entrada para garantir valores positivos

[MAX_POTENCIAL_MEMB] - parâmetro que regula o valor máximo que V_p pode assumir (V_p)

[CONSTANTE_TEMPO_Ti] - constante de relaxação do filtro de entrada (para todas conexões)

[CONSTANTE_TEMPO_To] - constante de relaxação do filtro de saída (para todos os neurônios)

[LIMIAR_DE_DISPARO] - constante que indica o valor do threshold estático, deve ser menor que V_p

[PESO_REFRATARIO] - constante/peso da realimentação da saída para o filtro de limiar dinâmico

[PERIODO_REFRAT_ABS] - número de ciclos do período refratário absoluto

Nesta implementação, foi assumido que os parâmetros de todos os neurônios são idênticos (população homogênea de neurônios). Esta rede permite ainda que um neurônio tenha uma única ou mais de uma entrada, ou que uma entrada seja avaliada por neurônios com pesos diferentes.

Esta camada não passa por nenhum processo de aprendizado, e, uma vez parametrizada, gera seqüências de pulsos em suas saídas proporcionais a seqüência de entrada correspondente.

6.2.3 Camada2 do reconhecedor

Esta camada foi montada com o objetivo de conseguir aprender as características relevantes das seqüências de entrada. Sua arquitetura foi derivada da rede auto-organizável SOM, apresentada neste trabalho, tendo em vista que a mesma conseguiu distinguir a estatística entre os locutores.

A configuração desta rede é um arranjo linear de neurônios, com os extremos conectados, formando um anel. Cada um dos neurônios recebe todas as saídas da camada1, e os neurônios competem entre si e interagem através da função de vizinhança variável durante a fase de treinamento.

A parametrização desta rede é semelhante a camada1, tendo sido adicionados os parâmetros que controlam a fase de aprendizado. A listagem do programa em linguagem C, que implementa esta camada, é apresentada no Apêndice VI.

A seguir é apresentada uma descrição da fase de aprendizado desta rede. Durante a fase de treinamento, dado um segmento da seqüência de entrada, é calculada a saída para todos os neurônios. Caso um ou mais neurônios apresentem disparo, é declarado vencedor aquele que apresentar maior potencial de membrana. O neurônio vencedor tem os seus pesos corrigidos e os neurônios da vizinhança também são atualizados, porém a intensidade da atualização é inversamente proporcional a distância do neurônio vencedor. A vizinhança é variável, diminuindo ao longo do treinamento, o que faz com que surjam regiões de maior atividade ao redor dos neurônios vencedores. A topologia circular evita o problema de distorção de borda, comum em redes auto-organizáveis. A correção dos pesos, é dada pela equação (6.1), sendo feita pela diferença entre a entrada e o peso atual da conexão.

$$w(n+1) = w(n) + \alpha(x(n) - w(n)) \quad (6.1)$$

onde: α = coeficiente de aprendizado

x = potencial pré-sináptico recebido, com valores discretos 0 ou 1

w = peso da conexão a ser atualizado

Os critérios de parada de treinamento, podem ser escolhidos entre o número de iterações da rede, ou um limiar de energia da rede. A energia da rede é definida como a somatória do quadrado de todos os pesos. A diferença entre a energia de uma iteração e da iteração seguinte permite avaliar se a rede está convergindo. Caso ocorra diminuição sucessiva no nível de energia, indica que a rede está convergindo.

6.2.4 Classificador

A etapa de classificação final do reconhecedor não foi foco deste trabalho, e portanto não foi implementada sobre redes pulsadas. O objetivo aqui é demonstrar que as redes pulsadas tem capacidade de manter as características relevantes do sinal de voz e destacá-las, permitindo utilizar estas redes para o reconhecimento automático do locutor.

O classificador escolhido para os testes é o MLP, utilizado segundo a mesma estratégia definida na etapa de validação dos conjuntos de treino e teste. A configuração de neurônios do classificador, varia de acordo com a dimensão do vetor de características utilizado.

6.2.5 Reconhecimento utilizando a energia dos filtros

O primeiro teste de reconhecimento, modo independente do texto, foi feito utilizando a energia dos filtros segundo a escala MEL. Desta forma foram utilizadas seqüências, correspondentes as locuções, com dimensão de 20 coeficientes. As locuções não foram normalizadas em comprimento.

camada1

Inicialmente procedeu-se a escolha dos parâmetros para a camada1. Os parâmetros críticos de ajuste são: T_i e T_o , que correspondem às constantes de tempo dos filtros de entrada e saída, e o limiar de disparo. Como o objetivo da primeira camada é o de codificar a variação da seqüência de entrada em uma seqüência de pulsos. Por isso,

foi fixado $T_i=10.0$, que faz com que o decaimento da exponencial do filtro de entrada seja completo antes da apresentação do segundo segmento. Para fixar o valor de T_o , tomou-se como base os experimentos preliminares realizados com o neurônio pulsado, onde foi usada uma relação de 10 para T_i/T_o . Esta relação permite que seja obtida a modulação em frequência dos pulsos. A variação do limiar de disparo é que determinará a sensibilidade a entrada e a duração das seqüências de pulsos. A Fig. 6.6, ilustra a seqüência de pulsos obtida para um dos locutores, após o ajuste dos parâmetros.

Fig. 6.6 – Seqüência resultante da conversão feita pela camada1 ($T_i=10$, $T_o=1$, $V_t=2.6$).

Foram experimentados outros valores para o limiar de disparo menores, porém a quantidade de pulsos produzida na saída da camada1 gerou muitas redundâncias de informação para a camada2. Este ajuste foi um bom compromisso conseguido entre modulação e quantidade de pulsos.

camada2

As seqüências obtidas da camada1 foram então submetidas para a camada2, composta por 100 neurônios organizados conforme descrito no item 6.2.3. Esta camada teve os parâmetros configurados experimentalmente, observando o padrão de

pulsos gerados por esta camada. A final de vários testes foram adotados $T_i=10$ e $T_o=1$, como primeira aproximação destes parâmetros. O limiar de disparo foi ajustado de forma que ocorressem disparos em número suficiente para que o treinamento fosse processado, e foi fixado em $V_t=0.2$.

O parâmetro de número de épocas para treinamento foi fixado em 500 épocas. O coeficiente de aprendizado foi fixado em 0.01, que é um valor prático de forma que a atualização dos pesos se processe de forma suave. Caso o valor do coeficiente de aprendizado seja muito grande a rede oscila e não apresenta convergência do mapa de pesos. A convergência do mapa de pesos é ilustrada pela Fig. 6.6.

Fig. 6.7 – Evolução dos pesos durante o treinamento da camada2 – 500 iterações.

Após o treinamento, as seqüências de pulsos da camada1 foram apresentadas para a rede treinada, sendo então obtidas as seqüências para a classificação. A camada2,

aprende as combinações de entrada e faz um mapeamento das seqüências segundo a estatística aprendida. Neste experimento, foi utilizado um único mapa para todos os locutores. A Fig. 6.7 ilustra uma seqüência obtida, após o mapeamento pela camada2.

Fig. 6.8 – Seqüência de pulsos, após mapeamento pela camada 2.

classificador

O classificador MLP foi configurado com 100 neurônios na camada de entrada, 300 neurônios na camada oculta e 10 neurônios na camada de saída. Para a utilização é necessário definir qual será o vetor de entrada para o mesmo. Foram estudadas as alternativas de codificação resultantes da camada 2, a saber:

- a) codificação de número de pulsos por neurônio ao longo da locução; nesta representação o número total de vezes que cada neurônio disparou, ao longo da locução é contado;
- b) codificação de número de pulsos por tempo; indica a quantidade de disparos que ocorreram em cada segmento da seqüência;
- c) utilização de toda a seqüência obtida com vetor de entrada para o classificador.

A representação c) apresenta-se como a mais completa, pois mantém todas as características temporais e espaciais da seqüência original. Contudo, esta representação é a mais onerosa do ponto de vista computacional, uma vez que obriga a uma dimensão de entrada grande no classificador. As representações a) e b), são mais práticas do ponto de vista de entrada no classificador, porém deixam de lado informações temporais e espaciais, respectivamente.

Foram realizados testes inicialmente, com as representações a) e b). Para o treinamento (camada1 e classificador) foi utilizado o conjunto1 e para os testes foi utilizado o conjunto 2. Os melhores resultados obtidos, são apresentados na Tab. 6.5.

Teste1

camada 1 com $T_i=10$, $T_o=1$, $V_t=2.6$
camada 2 com $T_i=10$, $T_o=1$, $V_t=0.2$

no.épocas	representação (a)	
	conjunto1	conjunto2
50	96,50%	59%
110	92%	63,50%
500	92,50%	68%

Teste2

camada 1 com $T_i=10$, $T_o=0.1$, $V_t=2.0$
camada 2 com $T_i=10$, $T_o=1$, $V_t=0.4$, épocas=500

representação (a)		representação (b)	
conjunto1	conjunto2	conjunto1	conjunto2
89%	63,50%	100%	35%

Tab. 6.5 – Melhores percentuais de reconhecimento com energia dos filtros MEL.

O baixo índice apresentado para a representação b) indica que os padrões gerados pelo conjunto2 são totalmente distintos daqueles gerados pelo conjunto1 e utilizados para o treinamento do classificador. Para o melhor resultado obtido (Teste1 com 500 épocas) é detalhado o número de acertos por locutor na Tab. 6.6.

Locutor	Reconhecido									
Testado	1	2	3	4	5	6	7	8	9	10
1	9	0	2	1	2	2	0	1	3	0
2	0	14	2	1	1	1	0	0	0	1
3	2	1	13	2	1	1	0	0	0	0
4	1	0	0	19	0	0	0	0	0	0
5	0	1	1	0	12	0	0	0	4	2
6	0	0	0	4	2	10	0	0	4	0
7	0	0	0	0	0	0	20	0	0	0
8	0	2	0	5	4	1	0	8	0	0
9	0	0	0	1	3	0	2	0	14	0
10	0	2	0	1	0	1	0	0	0	16

Tab. 6.6 – Melhor resultado detalhado por locutor.

6.2.6 Reconhecimento utilizando os coeficientes MFCC

A partir da configuração da camada1 e camada2, que apresentou o melhor resultado, foram executados testes utilizando os coeficientes MFCC, buscando verificar se as redes pulsadas estavam preservando a informação contida nas seqüências de coeficientes.

Uma vez que a seqüência de processamento é a mesma utilizada para o teste com a energia dos filtros, não será explicado novamente o funcionamento de cada camada que compõe o reconhecedor. Os melhores resultados serão apresentados, com os comentários relativos as alterações de configuração e distribuição da base de dados.

Os testes com os coeficientes MFCC foram realizados de forma paralela aos testes com a energia dos filtros. Nos testes iniciais apenas o conjunto de treino foi avaliado, e os resultados iniciais são apresentados na Tab. 6.7, onde foi feita a variação da constante de tempo do filtro de saída da camada2.

Teste 1

camada1 $T_i=10$, $T_o=1$, $V_t=0.52$

camada2 $T_i=10$, $T_o=1$, $V_t=0.2$, épocas=500

Ti camada2	To camada2	conjunto1	conjunto2
10	1	99,50%	82%
0,5	0,1	92%	71,50%
0,5	2	95%	76,50%
0,5	0,5	95%	78,50%
0,5	10	92%	77,50%
1	1	96%	79,50%

Tab. 6.7 – Resultados iniciais de reconhecimento com coeficientes MFCC.

Foi observado então, que o percentual de reconhecimento para o conjunto de treino (conjunto1), ficava sempre entre 99,5% e 92%. Isto é um forte indicador de que o classificador estava assimilando os valores do conjunto de treino. Foram feitas então variações na estratégia de treinamento, alterando o período de treinamento e até a dimensão da camada oculta do MLP. Os resultados não apresentaram alteração significativa, mantendo-se estáveis.

Foi feito então, o treinamento do classificador utilizando os dois conjuntos, para verificar se o comportamento não era devido a uma diferença grande na quantidade de informações do locutor presente em cada um. Foram obtidos percentuais de 98,5% para o conjunto1 e de 99% para o conjunto2.

Decidiu-se então, variar o tamanho do segmento de amostras analisado, sendo testadas duas alternativas:

- segmentos de 512 pontos (64mseg) com deslocamento de 32 pontos (4mseg);
- segmentos de 128 pontos (16mseg) com deslocamento de 32 pontos (4mseg).

No primeiro caso, a utilização da nova segmentação resultou na manutenção do percentual do conjunto1 – 96% - e em melhora no percentual de reconhecimento do conjunto2 – 84%. Comprovando que esta segmentação acrescentou informação às seqüências de representação das locuções. O segundo caso apresentou piora considerável para os dois conjuntos, com o percentual de reconhecimento ficando abaixo dos 50%.

Então foi usada uma abordagem de redistribuição dos conjuntos de treino e teste, visando aumentar a quantidade de informação no treinamento, para verificar se ocorre melhora no percentual de reconhecimento.

Primeiramente foram consideradas apenas as frases 04 e 08, como conjunto de teste, e o treinamento foi efetuado com o restante das frases. Em seguida as frases 04 e 08 foram substituídas pelas frases 01 e 05, que contém conteúdo fonético semelhante. Por último, apenas a frase 05 foi utilizada para o teste. O uso de frases alternadas foi feito para que os dois conjuntos iniciais fossem afetados.

Nas duas primeiras situações, houve alteração significativa na taxa de reconhecimento para as frases 04 e 01 do conjunto de teste, indicando que com mais informação o classificador fica apto para reconhecer o locutor com estas frases. Esta é uma indicação forte de que as redes pulsadas estão preservando a informação. No último caso, nos dois primeiros testes o reconhecimento com as frases 08 e 05 manteve-se dentro dos níveis anteriores. Porém ao retornar a frase 01 para o conjunto de treino, a frase 05 apresentou 10% de melhora, indicando mais uma vez que foi acrescentada informação ao classificador e que a rede pulsada foi capaz de extraí-la. A Tab. 6.8, apresenta os resultados destes testes.

Teste 2

camada1 $T_i=10$, $T_o=1$, $V_t=0.52$

camada2 $T_i=10$, $T_o=1$, $V_t=0.2$, épocas=500

conjunto de treino A: frases 01-02-03-05-06-07

conjunto de treino B: frases 02-03-04-06-07-08

conjunto de treino C: frases 01-02-03-04-06-07-08

conjunto de treino	frase de teste	reconhecimento
A	4	98%
A	8	76%
B	1	98%
B	5	76%
C	5	86%

Tab. 6.8 – Reconhecimento com ampliação do conjunto de treino.

7. DISCUSSÃO DOS RESULTADOS E CONCLUSÃO

7.1 Discussão dos resultados

Nesta tese foram estudados os fundamentos sobre o funcionamento das redes neurais pulsadas, e especificamente o modelo de resposta impulsiva, o qual pode ser facilmente implementado computacionalmente.

Os testes preliminares realizados foram fundamentais para o entendimento do princípio de funcionamento dos neurônios pulsados. Através destes experimentos pôde ser verificado o efeito de integração temporal que é executado pelo neurônio, e entendido o mecanismo de funcionamento do limiar de disparo dinâmico. Este dois mecanismos permitem ao neurônio pulsado, gerar padrões de pulso em sua saída altamente complexos, e com isso espera-se diminuir sensivelmente o tamanho de uma rede neural para implementar uma função complexa.

Não existe ainda uma regra geral, que permita ao neurônio buscar o melhor ajuste dos seus parâmetros, de forma a independe por completo de qualquer ajuste manual em algum de seus parâmetros. Este ajuste é totalmente empírico e mesmo nos casos onde é reportado o treinamento automático na literatura. Ao examinar o algoritmo é comum deparar-se com constantes empíricas, via de regra justificadas através do paralelo com a biologia. Esta foi uma dificuldade sentida neste trabalho, e certamente devem ser colocados esforços no sentido de estabelecer métodos práticos e automáticos para o estabelecimento dos parâmetros do neurônio pulsado.

Na análise de comportamento coletivo, pôde ser observado como um padrão de resposta pode ser gerado, e a partir daí passar a propor os diversos estágios do reconhecedor desenvolvido. Um ponto de dificuldade foi conseguir ferramentas que propiciassem visualizar de forma adequada, as seqüências geradas pelas redes pulsadas. Ferramentas usuais como cálculo de matriz de distâncias ou *learning vector quantization* – *LVQ* não foram capazes de proporcionar uma visão do tipo de representação com a qual se estava trabalhando. Assim, foi usado um método das próprias redes neurais para permitir a análise das seqüências geradas, o *Self Organizing Map* – *SOM*. Esta rede proporcionou visualizar o comportamento das

seqüências de saída e inspirou a implementação de uma camada auto-organizável, com arranjo e algoritmo de treino da qual não se teve notícia na literatura.

Com relação a arquitetura implementada, a utilização de um primeiro estágio de neurônios pulsados foi inspirada no mecanismo de funcionamento da cóclea. Tendo sido também motivador para a utilização de características do sinal de voz extraídas a partir de um banco de filtros segundo a escala perceptual.

A segunda camada do reconhecedor foi implementada inspirada no mecanismo de auto-organização das redes de *Kohonen*, a qual tem plausibilidade biológica e teve por objetivo buscar padrões nas seqüências de disparo que pudessem permitir a identificação do locutor. Pelo fato de ter sido implementada em uma única dimensão, esta rede funciona como um quantizador vetorial, aproximando os padrões de entrada que são apresentados. A evolução do mapa de pesos, para um formato menos suave, deve-se ao fato das entradas serem discretizadas em dois níveis (zero e um). O que pôde ser observado no mapa é que o mesmo aproxima bem as seqüências de entrada e, ao final do treinamento os pesos passam a apresentar uma distribuição que corresponde as seqüências que ocorrem com maior freqüência. Entretanto, a implementação de uma versão bidimensional deve proporcionar resultados mais precisos, uma vez que a representação será feita em uma dimensão maior.

Outro ponto que merece destaque é a pequena dimensão das redes, com excessão do classificador que será objeto de comentários mais adiante. Na primeira camada a rede foi construída com 16 ou 20 neurônios, de acordo com o tipo de representação utilizado. Na camada 2, por sua vez foi utilizada uma rede de 100 neurônios. Assim em comparação com outros classificadores neurais como o MLP ou a RBF, a quantidade de neurônios é menor.

Os resultados obtidos na forma dependente do texto, apesar de razoáveis, apontaram para uma certa fragilidade do conjunto de locuções utilizado, porém serviu para constatar que os padrões gerados pela camada 1 carregam informações do locutor.

Na forma independente do texto, a partir do momento em que os resultados obtidos ficaram bem abaixo do resultado obtido com o classificador clássico, surgiu o dilema sobre se o classificador estava sendo viciado pelas amostras de treinamento, ou se realmente as informações do locutor não estavam sendo fornecidas pelas redes

pulsadas. Foram realizados vários testes, com o objetivo de determinar porque a taxa de reconhecimento obtida era baixa. Foi questionada a representação final utilizada para o treinamento do classificador, indagando se a mesma era a melhor alternativa. O problema verificado com o classificador clássico é que o mesmo depende de padrões estáticos para ser treinado, o que contraria a forma dinâmica com que a informação é processada pelas redes pulsadas. Conforme comentado na parte experimental, certamente tem-se perda de informação no instante em que é criada uma representação que condense o aspecto temporal ou espacial da informação apresentada pela saída da camada2. Apesar desta condensação de informação o resultado obtido foi 27% melhor do que os resultados obtidos no trabalho de Ho[105], uma das poucas referências que trata do reconhecimento do locutor e que também utiliza um classificador neural na saída do reconhecedor.

A seguir é apresentado um resumo comparativo com outros trabalhos que utilizam as redes pulsadas para o reconhecimento automático do locutor.

RAL dependente do texto

- Berger et al – Dynamic synapses [11]
 - banco de filtros e wavelets no pré-processamento;
 - base de dados com 6 locutores;
 - 240 parâmetros de ajuste;
 - reconhecimento de 100% no conjunto de treino e de 87% no conjunto de teste.
- Malkani – Pulsed coupled neural networks [14]
 - espectrograma no pré-processamento;
 - base de dados com 10 locutores;
 - contagem do número de neurônios que dispara, gerando ícone;
 - não apresenta resultados quantitativos, apenas mostra que os ícones diferem entre os locutores.
- Sistema desenvolvido nesta teste
 - pré-processamento através de banco de filtros na escala Mel;
 - base de dados de 10 locutores;

- reconhecimento de 100% no conjunto de treino e de 90% no conjunto de teste.

RAL independente do texto

- Tuong Vinh Ho [105]
 - pré-processamento através de banco de filtros;
 - base de dados com 10 locutores;
 - uma única camada de neurônios pulsados atuando como codificador da seqüência de entrada;
 - classificador implementado por meio de uma rede neural clássica;
 - taxa de reconhecimento de 57%.
- Sistema desenvolvido nesta tese
 - pré-processamento utilizando banco de filtros na escala Mel e coeficientes MFCC;
 - base de dados com 10 locutores;
 - duas camadas de neurônios pulsados atuando como codificador e mapeador da seqüência de entrada;
 - classificador implementado por meio de uma rede neural clássica;
 - taxa de reconhecimento utilizando coeficientes de energia do banco de filtros, 68%;
 - taxa de reconhecimento utilizando os coeficientes MFCC, 84%.

Para que a dinâmica das seqüências de saída possa ser totalmente aproveitada, é necessário concentrar esforços no sentido do desenvolvimento de um classificador com redes pulsadas.

Apesar de não configurar o reconhecedor de forma completa, com todas as suas camadas formadas por neurônios pulsados, está claro que as redes pulsadas apresentam uma capacidade de representação sem igual em outros paradigmas neurais. Por este motivo presta-se ao processamento de sinais temporais, conforme foi mostrado nesta tese.

7.2 Conclusão

Esta tese apresentou um estudo sobre a possibilidade de se utilizar as redes neurais pulsadas, para a tarefa do reconhecimento automático do locutor. Conforme pôde ser observado na revisão dos trabalhos correlatos, quase nada foi publicado sobre esta aplicação das redes neurais pulsadas.

As principais contribuições originais desta tese foram:

- abordar o estudo das redes neurais pulsadas sob a ótica do reconhecimento automático do locutor;
- apresentar uma rede auto-organizável, composta por neurônios pulsados, que possibilita a memorização dos padrões característicos das seqüências de entrada;
- apresentar uma arquitetura nova, composta por redes pulsadas interligadas, para o reconhecimento automático do locutor.

De uma forma geral o objetivo de estudar a aplicação das redes neurais pulsadas, para o reconhecimento automático do locutor foi atingido.

Certamente os resultados obtidos nesta tese darão origem a diversos outros trabalhos na área de voz, uma vez que foi comprovado que o paradigma das redes pulsadas é adequado para o reconhecimento automático do locutor.

Como sugestão para trabalhos futuros apontamos dois pontos de extremo interesse:

- ampliação da rede auto-organizável para uma segunda dimensão e avaliação desta nova rede integrada ao reconhecedor;
- desenvolvimento de uma camada classificadora, com neurônios pulsados, que permita aproveitar toda a dinâmica das seqüências geradas pelas camadas anteriores;
- estudo de métodos de extração de características do sinal de voz, que sejam biologicamente plausíveis e possam ser integrados com as redes pulsadas;
- estudos para a utilização de algoritmos genéticos para o desenvolvimento da rede;
- estudos para verificar se as trajetórias geradas pela rede auto-organizável formam caos.

8. REFERÊNCIAS

- [1] MAASS, W. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, v.10, n.9, p.1659-71, Dec. 1997.
- [2] KRÖTZSCH, M. Networks of Spiking Neurons – An alternative approach to connectionist systems. Disponível em:
<http://www.wv.inf.tudresden.de/~pascal/teaching/ss02/manuscripts>.
 Data de acesso: 22 de maio de 2003.
- [3] KURONAYAGI, S. IWATA, A. Auditory pulse neural network model to extract the inter-aural time and level difference for sound localization. *IEICE Transactions*. v.E77-D, n.4, p.466-74, Apr. 1994.
- [4] KURONAYAGI, S. IWATA, A. The supervised learning rules of the pulse neuron model – learning of the connection weights and the delay times. In: *PROCEEDINGS OF THE ICONIP'99*, 1999, p.7-12.
- [5] KURONAYAGI, S. IWATA, A. An unsupervised learning rule for the pulsed neuron model - the vector quantization of the auditory temporal signals. p.285-290, 2000.
- [6] KURONAYAGI, S. IWATA, A. An competitive learning pulsed neural network for signals temporal. In: *PROCEEDINGS OF THE ICONIP'02*, 2002, v.1, p.348-52.
- [7] SCHAUER, C. et al Binaural sound localization in an artificial neural network. In: *IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING*, s.l., 2000. Proceedings. Piscataway, IEEE, 2000, v.2, p.865-8.
- [8] SMITH L.S. Onset-based sound segmentation. In: *ADVANCES IN NEURAL INFORMATION AND PROCESSING*, s.l., 1995. Proceedings. Cambridge, MIT Press, 1996, p.729-35.
- [9] GLOVER, M.A. HAMILTON, A. SMITH, L.S. An analog VLSI integrante-and-fire neural network for sound segmentation. In: *Proceedings of NC 1998, INTERNATIONAL ICSC/IFAC SYMPOSIUM ON NEURAL COMPUTATION*, s.l., 1998. Proceedings. Zurich, ICSC Academic Press, 1998, p.86-92.
- [10] LIAW, J.S. BERGER, T.W. Robust speech recognition with dynamic synapses. In: *IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS*, s.l., 1998. Proceedings. New York, IEEE, 1998, v.3, p.2175-9.
- [11] GEORGE, S. DIBAZAR, A. BERGER, T.W. Speaker recognition using dynamic synapse neural networks. In: *PROCEEDINGS OF THE SECOND JOINT EMBS/BMES CONFERENCE*, 2002. Proceedings. Houston, IEEE, 2002, p.151-2.

- [12] STORCK, J. JÄKEL, F. DECO, G. Temporal clustering with spiking neurons and dynamic synapses: towards technological applications. *Neural Networks*, v.14, n.3, p.275-85, Apr. 2001.
- [13] MERCIER, D. SÉGUIER, R. Spiking neurons (STANNs) in speech recognition. In: 3rd WSES INTERNATIONAL CONFERENCE ON NEURAL NETWORKS AND APPLICATIONS, Interlaken, Feb. 2002. Proceedings.
- [14] CHANDRASEKARAN P. et al Speech recognition using pulse coupled neural network. In: PROCEEDINGS OF THIRTIETH SOUTHEASTERN SYMPOSIUM ON SYSTEMS THEORY, s.l., 1998. New York, IEEE, 1988, p.515-19.
- [15] HOPFIELD, J.J. BRODY, C.D. What is a moment? “Cortical” sensory integration over a brief interval (I). In: PROCEEDINGS OF THE NATIONAL ACADEMY OF SCIENCES OF THE USA, v.97, n.25, p.13919-13924, 2000.
- [16] WILLS, S.A. Computing with spiking neurons. Phd Report of Inference Group Cavendish Laboratory, Jun. 2001.
- [17] HOPFIELD, J.J. BRODY, C.D. What is a moment? Transient synchrony as a collective mechanism for spatiotemporal integration. In: PROCEEDINGS OF THE NATIONAL ACADEMY OF SCIENCES OF THE USA, v.99, n.3, p.1282-1287, 2001.
- [18] FURUI, S. Digital speech processing, synthesis, and recognition. New York, Marcel Dekker, 1989.
- [19] RABINER, L.R. Applications of voice processing to telecommunications. *Proceedings of the IEEE*, v.82, n.2, p.197-228, Feb. 1994.
- [20] ROSENBERG, A.E. Automatic speaker verification: a review. *Proceedings of the IEEE*, v.64, n.4, p.475-86, Apr. 1976.
- [21] ATAL, B.S. Automatic recognition of speakers from their voices. *Proceedings of the IEEE*, v.64, n.4, p.460-75, Apr. 1976.
- [22] CORSI, P. Speaker recognition: a survey. In: PROCEEDINGS OF THE NATO ADVANCED STUDY INSTITUTE, Bonas, 1981. *Automatic Speech Analysis and Recognition*, Dordrecht, D.Reidel Publishing company, 1982. p.277-308
- [23] GISH, H.; SCHMIDT, M. Text-independent speaker identification. *IEEE Signal Processing Magazine*, v.11, n.4, p.18-32, Oct. 1994.
- [24] CAMPBELL JR., J.P. Speaker recognition: a tutorial. *Proceedings of the IEEE*, v.85, n.9, Sep. 1997.

- [25] SCHAFER, R.W.; RABINER, L.R. Digital representations of speech signals. *Proceedings of the IEEE*, v.63, n.4, p.662-77, Apr. 1975.
- [26] SCHAFER, J.J.; RABINER, L.R. System for automatic formant analysis of voiced speech. *The Journal of the Acoustical Society of America*, v.47, n.2, p.634-48, 1970.
- [27] RABINER, L.R.; SCHAFER, R.W. *Digital processing of speech signals*. Englewood Cliffs, Prentice Hall, 1978.
- [28] DELLER JR., J.R. et al. *Discrete-time processing of speech signals*. New York, MacMillan, 1993.
- [29] BEZERRA, M.R. Reconhecimento automático de locutor para fins forenses, utilizando técnicas de redes neurais. Rio de Janeiro, 1994. 202p. Dissertação (Mestrado) - Instituto Militar de Engenharia.
- [30] CABRAL JR, E.F. Reconhecimento de locuções discretas: estudo e implementação. Rio de Janeiro, 1988. 132p. Dissertação (Mestrado) - Instituto Militar de Engenharia.
- [31] FURUI, S. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech and Signal Processing*. v.29, n.2, p.254-72, Apr. 1981.
- [32] O'SHAUGHNESSY, D. Speaker recognition. *IEEE ASSP Magazine*, v.3, p.4-17, Oct.1986.
- [33] WOLF, J.J. Efficient acoustic parameters for speaker recognition. *The Journal of the Acoustical Society of America*, v.51, n.6, p.2044-56, 1972.
- [34] GOLDSTEIN, U.G. Speaker-identifying features based on formant tracks. *The Journal of the Acoustical Society of America*, v.59, n.1, p.176-81, Jan. 1976.
- [35] SAMBUR, M.R. Selection of acoustic features for speaker identification. *IEEE Transactions on Acoustics, Speech and Signal Processing*, v.23, n.2, p.176-82, Apr. 1975.
- [36] PICONE, J.W. Signal modeling techniques in speech recognition. *Proceedings of the IEEE*, v.81, n.9, p.1215-47, Sept. 1993.
- [37] TOHKURA, Y. A weighted cepstral distance measure for speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*. v.35, n.10, p.1414-22, Oct. 1987.
- [38] RUNSTEIN, F.; VIOLARO, F.; NUNES, H.F. Uso de diferentes parâmetros de entrada em u sistema de reconhecimento de fala baseado em redes neurais. In:

SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES, 13º, Aguas de Lindóia, 1995. Anais. Campinas, Unicamp, 1995, p.155-60

[39] JANKOWSKI JR., C.R.; VO, H.H.; LIPPMANN, R.P. A comparison of signal processing front ends for automatic word recognition. IEEE Transactions on Speech and Audio Processing, v.3, n.4, p.286-93, July 1995.

[40] JANKOWSKI JR., C.R. Noise Masking in the MFCC domain. Proceedings of the Institute of Acoustics, v.14, part 6, p.363-367, 1992.

[41] REYNOLDS, D.A.; ROSE, R.A. Robust text-independent speaker identification using gaussian mixture speaker models. IEEE Transactions on Speech and Audio Processing, v.3, n.1, p.72-83, Jan.1995.

[42] GLENN, J.W.; KLEINER N. Speaker identification based on nasal phonation. The Journal of the Acoustical Society of America, v.43, n.4, p.368-72, 1968.

[43] BRICKER, P.D. et al. Statistical techniques for talker identification. The Bell System Technical Journal. v.50, n.4, p.1427-54, Apr. 1971.

[44] GARCIA, J.A. et al. A dynamic approach for clustering data. Signal Processing, v.44, p.181-96, 1995.

[45] BASSEVILLE, M. Distance measures for signal processing and pattern recognition. Signal Processing, v.18, p.349-69, 1989.

[46] CABRAL JR., E.F.; TATTERSALL, G.D. Trace-segmentation of isolated utterances for speech recognition. In: INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, Detroit, 1995. Proceedings. Piscataway, IEEE, 1995. v.1, p.365-8.

[47] LINFORD, P.W.; TATTERSALL, G.D. Non-linear time normalization of utterances for speech recognition using MLP's. Proceedings of the Institute of Acoustics, v.12, part 10, p.291-297, 1990.

[48] TIMOSZCZUK, A.P. Reconhecimento automatico do locutor com redes neurais artificiais do tipo radial basis function (RBF) e minimal temporal information (MTI), Dissertação de Mestrado, USP, Sao Paulo, 1998.

[49] WILPON, J.G.; RABINER, L.R. A modified K-means clustering algorithm for use in isolated word recognition. IEEE Transactions on Acoustics, Speech and Signal Processing. v.33, n.3, p.587-94, June 1985.

[50] CHINRUNGRUENG, C.; SÉQUIN, C.H. Optimal adaptive K-means algorithm with dynamic adjustment of learning rate. In: INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, s.l., 1991. Proceedings. s.ed., s.l., 1991. v.1, p.855-62

- [51] LINDE, Y.; BUZO, A.; GRAY, R.M. An algorithm for vector quantizer design. IEEE Transactions on Communications. v.28, n.1, p.84-95, Jan. 1980
- [52] BURTON, D.K. Text-dependent speaker verification using vector quantization source coding. IEEE Transactions on Acoustics, Speech and Signal Processing. v.35, n.2, p.133-43, Feb. 1987.
- [53] KURASHIMA, C.S.; CABRAL JR., E.F. A survey on speech enhancement techniques of interest to speaker recognition. São Paulo, EPUSP, 1994. (Boletim Técnico da USP. Departamento de Engenharia Eletrônica, BT/PEE94-06).
- [54] WIDROW, B.; LEHR, M.A. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. Proceedings of the IEEE, v.78, n.9, p.1415-41, Sept. 1990.
- [55] HEBB, D.O. The organization of behavior: a neuropsychological theory. New York, John Wiley, 1949.
- [56] ASHBY, W.R. Design for a brain. New York, John Wiley, 1952.
- [57] MINSKY, M.L. Theory of neuro-analog reinforcement systems and its application to the brain-model problem. Princeton, Thesis (PhD) - Princeton University.
- [58] UTLEY, A.M. A theory of the mechanism of learning based on the computation of conditional probabilities. In: INTERNATIONAL CONFERENCE ON CYBERNETICS, 1º, Namur, 1956. Proceedings. Paris, Gauthier-Vellair, 1956
- [59] ROSENBLATT, F. The Perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review. n.65, p.386-408, 1958.
- [60] WIDROW, B.; HOFF JR, M.E. Adaptive switching circuits. IRE WESCON. Convention Record, v.4, p.96-104, 1960.
- [61] MINSKY, M.L.; PAPERT, S.A. Perceptrons. Cambridge, MIT Press, 1969.
- [62] VON DER MARLSBURG, C. Self-organization of orientation sensitive cells in the striate cortex. Kybernetik, n.14, p.85-100, 1973.
- [63] WILLSHAW, D.J.; VON DER MARLSBURG, C. How patterned neural connections can be set up by self-organization. Proceedings of the Royal Society of London. Series B, 1976.
- [64] KOHONEN, T. Associative memory: a system-theoretical approach. New York, Springer, 1977.

- [65] GROSSBERG, S. How does a brain build a cognitive code? Psychological Review. n.87, p.1-51, 1980.
- [66] HOPFIELD, J.J. Neural networks and physical systems with emergent collective computational habilities. Proceedings of the national Academy of Sciences of the U.S.A., n.79, p.2554-8, 1982.
- [67] KOHONEN, T. Self-organized formation of topologically correct feature maps. Biological Cybernetics. n.43, p.59-69, 1982.
- [68] KIRKPATRICK, S.; GELATT, C.D.; VECCHI, M.P. Optimization by simulated annealing. Science, n.220, p.671-680, 1983.
- [69] COHEN, M.A.; GROSSBERG, S. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. IEEE Transactions on Systems, Man, and Cybernetics, v.13, p.815-26, 1983.
- [70] BARTO, A.G.; SUTTON, R.S.; ANDERSON, C.W. Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics. v.13, p.34-846, 1983.
- [71] WERBOS, P.J. Beyond regression: new tools for prediction and analysis in the behavioral sciences. Cambridge, 1974, Thesis(PhD) - Harvard University.
- [72] RUMELHART, D.E.; MCCLELLAND, J.L. Parallel distributed processing. San Diego, MIT Press, 1986.
- [73] BROOMHEAD, D.S.; LOWE, D. Multi-variable function interpolation and adaptive networks. Complex Systems, v.2, p.269-303, 1988.
- [74] STUART, G.J.; SAKMAN, B. Active propagation of somatic action potentials into neocortical pyramidal cell dendrites. Nature, v.367, 6 Jan., 1994.
- [75] HOPFIELD, J.J. Neurons, dynamics and computation. Physics Today, Feb., 1994.
- [76] BARINAGA, M. Dendrites shed their dull image. Science, v.268, 14 Apr. 1995.
- [77] MAINEN, Z.F.; SEJNOWSKI, T.J. Reliability of spike timing in neocortical neurons. Science, v.268, 9 Jun. 1995.
- [78] NATSCHLÄNGER, T. Networks of spiking neurons: a new generation of neural network models. Disponível em:
http://www.cis.tugraz.at/igi/tnatschl/online/3rd_gen_eng/3rd_gen_eng.html
 Data de acesso: 22 de novembro de 2003.

- [79] FU, L. Neural networks in computer intelligence. New York, MacGraw Hill, 1994.
- [80] HAYKIN, S. Neural networks: a comprehensive foundation. New York, MacMillan, 1994.
- [81] McCULLOCH, W.S.; PITTS, W. A logical calculus of ideas immanent in nervous activity. Bulletin of Mathematical Biophysics. n.5, p.115-133, 1943.
- [82] HUSH, D.R.; HORNE, B.G. Progress in supervised neural networks: what's new since Lippmann? IEEE Signal Processing Magazine, v.10, n.1, p.8-39, Jan. 1993.
- [83] LIPPMANN, R.P. An introduction to computing with neural nets. IEEE ASSP Magazine, v.4, n.2, p.4-22, Apr. 1987.
- [84] COVER, T.M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE Transactions on Electronic Computers, v.14, n.3, p.326-334, June 1965.
- [85] CABRAL JR., E.F Reconhecimento automático do locutor com redes neurais artificiais: um curso teórico/prático para engenheiros e cientistas. Edição dos autores, Ed.Suprema, 1998.
- [86] YU, X.; CHEN, G.; CHENG, S. Dynamic learning rate optimization of the backpropagation algorithm. IEEE Transactions on Neural networks, v.6, n.3, p.669-77, May 1995.
- [87] HERTZ, J.; et.al. Introduction to the theory of neural computation. Redwood City, CA, Addison-Wesley, 1991.
- [88] KOHONEN, T. *The Self-organizing Map* Proceedings of the IEEE vol.78, n.9, pp. 1464-1480, 1990.
- [89] KOHONEN, T. *Clustering, Taxonomy, and Topological Maps of Patterns* Proceedings of the 6th International Conference on Pattern Recognition, pp. 114-128, Munich, Germany, 1982b.
- [90] CHEN, S. K.; MANGIAMELI, P.; WEST, D. *The Comparative Ability of Self-organizing Neural Networks to Define Cluster Structure* Omega Int. J. Mgmt Sci., vol. 23, n.3, pp. 271-279, 1995.
- [91] TATTERSALL, G. D.; LINFORD, P. W. *Self-Organizing Kohonen Networks* University of Anglia, U. K., 1990.
- [92]

- [93] HODGKIN, A.L. HUXLEY, A.F. A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes. *J. Physiol.* , 117, p.500-544, London, 1952.
- [94] GERSTNER, W.; KISTLER, W. Spiking neuron models: single neurons, populations, plasticity. Cambridge University Press, Cambridge, 2002.
- [95] RIEKE, F.; et.al. Spikes: exploring the neural code. MIT Press, Cambridge, 1996.
- [96] MAINEN, Z.F; SEJNOWSKI, T.J. Reliability of spike timing in neocortical neurons. *Science*, v.268, p.1503-6, 9 Jun. 1995.
- [97] GERSTNER, W. Spiking Neurons. In MAASS, W. and BISHOP, C., editors , *Pulsed Neural Networks*, pp.3-53, MIT-Press, Cambridge, 1999.
- [98] HOPFIELD – 1997 – “Neural” Computing with Action Potentials. In: *Electrical Engineering Colloquium Series – Spring 1997*. Moore School of Electrical Engineering, Philadelphia, 1997.
- [99] SINGER – 1993 – Neuronal Representations and Temporal Codes. In: T.A. Poggio and D.A. Glaser (eds): *Exploring Brain Functions: Models in Neuroscience*. Dahlem Conferences, 1991. Chichester, John Wiley & Sons Ltd, pp.179-194.
- [100] STEIN, R.B. The information capacity of nerve cells using a frequency code. *Biophysics, J.* n.7, 1967, p.797-826.
- [101] GERSTNER, W. Associative memory in a network of “biological” neurons. In *Advances in Neural Information Processing Systems*, 1991, pp.84-90, San Mateo, Morgan Kaufmann Publishers.
- [102] HERTZ, J. KROGH, A. and PALMER, R.G. Introduction to the theory of neural computation. Addison-Wesley, 1991.
- [103] JAHNKE, A. ROTH, U. SCHÖNAUER, T, Digital simulation of spiking neural networks. In Mass, W. and Bishop, C., editors , *Pulsed Neural Networks*, pp.237-257, MIT-Press, Cambridge, 1999.
- [104] REYNOLDS, D.A.; ROSE, R.A. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, v.3, n.1, p.72-83, Jan.1995.
- [105] HO, T.V. Un réseau de neurones à décharges pour la reconnaissance de processus spatio-temporels. Thèse, Université de Montréal, jan. 1999.

APÊNDICE I

Código do modelo de neurônio em MatLab.

```
% Spike Response Model
% Data: 20/03/2003
% Autor: Antonio Pedro Timoszczuk
% Descricao: Teste de modelo implementado de neuron implementado a
% partir da sequencia descrita no capítulo 4. Para uso criar uma %
%%% sequência de entrada xi no Matlab para submeter ao neurônio.
% Revisões:
% 18/04/2003 - Neste modelo foi introduzido o periodo refratario
% absoluto
%

% inicializacao de parametros

No=1; % Constante de realimentação
Vt=0.1; % Threshold estático
Vabs=0; % Constante de período refratário absoluto (uma iteracao
apos
    emissao de pulso
wi=1; % Peso de entrada
T=1; % Base de tempo, nao utilizada
Ti=1; % Constante de tempo do filtro de entrada, persistencia da
    entrada
To=0.1; % Constante de tempo do periodo refratario

% inicializacao dos vetores de teste

for n=1:1:139
    pi(n)=0;
    po(n)=0;
    %xi(n)=0; nao esquecer de montar o vetor xi no ambiente
    fi(n)=0;
    fo(n)=0;
    fot(n)=0;
    ui(n)=0;
    y(n)=0;
end

% Estagio do potencial pre-sinaptico, acoplamento com peso unitario

for n=2:1:138

% fase de entrada
    pi(n)=wi*xi(n);
    po(n)=No*y(n-1);

% fase de filtragem
    fi(n)=pi(n) + (exp(-Ti*T))*fi(n-1);
    fo(n)=po(n) + (exp(-To*T))*fo(n-1);

% fase de saída
    fot(n)= -(fo(n)+Vt);
    ui(n)= fot(n)+fi(n);
```



```

    if Vabs==0;
        if ui(n)>=0;
            y(n)=1;
            Vabs=4; % Aqui seta a condicao refrataria absoluta, pois
ocorreu
                        disparo neste caso o período é de 4 unidades de
tempo
            else
                Vabs=0; % Se nao ha o disparo seta a condicao refrataria
absoluta
                        zero
            end
        else
            Vabs=Vabs-1; % Reseta a condicao refrataria absoluta, pois veio
de
                        um disparo
        end
    end

% fase de apredizado entra aqui

end

% montagem do grafico de apresentacao

subplot(5,1,1)
plot(xi)
text(-20,1,'Ventr')
ylabel('Ventr')
subplot(5,1,2)
plot(fi)
text(-20,0,'Vsinap')
ylabel('Vsinap')
subplot(5,1,3)
plot(fot)
text(-20,0,'Vrefr')
ylabel('Vrefr')
subplot(5,1,4)
plot(ui)
text(-20,0,'Vmemb')
ylabel('Vmemb')
subplot(5,1,5)
plot(y)
text(-20,1,'Vspike')
ylabel('Vspike')
xlabel('tempo')

```

APÊNDICE II

Resultados do teste de verificação do comportamento inter-locutor e intra-locutor.

Fig. A2.1 – teste inter-locutor utilizando a seqüência de pulsos derivada dos MFCCs.

Fig. A2.2 – teste inter-locutor utilizando os coeficientes MFCC como entrada.

Fig. A2.3 – teste intra-locutor utilizando a seqüência de pulsos derivada dos MFCCs.

Fig. A2.4 – teste intra-locutor utilizando os coeficientes MFCC como entrada.

Fig. A2.1 - Resultado teste inter-locutor / locutores 1 –10 na ordem da esquerda para direita e de cima para baixo/ pulsos pico positivo / repetição 1 .

Fig. A2.2 - Resultado do teste inter-locutor / locutores 1 a 10 na ordem da esquerda para direita e de cima para baixo / coeficientes mfccs / repetição 1 .

Fig. A2.3 - Resultado teste intra-locutor / locutor 1/ repetições de 1-10 na ordem da esquerda para direita e de cima para baixo / pulsos derivados dos coeficientes mfcc / repeticao 1 –10.

Fig. A2.4 - Resultado do teste intra-locutor / locutor 1/ repetições de 1-10 na ordem da esquerda para direita e de cima para baixo / coeficientes mfcc / repetições de 1 a 10.

APÊNDICE III

Nas páginas seguintes, são apresentados os mapas de pesos obtidos após o treinamento das redes SOM, para cada um dos locutores.

TESTE03 – run01 – MAPA DE KOHONEN – pesos finais por locutor

LOCUTOR 01

LOCUTOR 02

LOCUTOR 03

LOCUTOR 04

LOCUTOR 05

LOCUTOR 06

LOCUTOR 07

LOCUTOR 08

LOCUTOR 09

LOCUTOR 10

APÊNDICE IV

Trajetórias tridimensionais dos locutores na rede SOM:

Fig A4.1a e A4.1b – Trajetórias obtidas a partir da energia dos filtros na escala MEL.

Fig A4.2a e A4.2b – Trajetórias obtidas a partir dos coeficientes MFCC.

Fig A4.3a e A4.3b – Trajetórias obtidas a partir dos pulsos gerados a partir dos coeficientes MFCC.

Fig A4.4a e A4.4b – Trajetórias obtidas a partir dos pulsos gerados a partir dos coeficientes de energia dos filtros.

TRAJETORIAS DOS LOCUTORES NA REDE SOM
COEFICIENTES DE ENERGIA DOS FILTROS NA ESCALA MEL

LOCUTOR 01

LOCUTOR 02

LOCUTOR 03

LOCUTOR 04

LOCUTOR 05

LOCUTOR 06

Fig. A4.1a

LOCUTOR 07

LOCUTOR 08

LOCUTOR 09

LOCUTOR 10

Fig. A4.1b

TRAJETÓRIAS DOS LOCUTORES NA REDE SOM – COEFICIENTES MFCC

LOCUTOR 01

LOCUTOR 02

LOCUTOR 03

LOCUTOR 04

LOCUTOR 05

LOCUTOR 06

Fig. A4.2a

LOCUTOR 07

LOCUTOR 08

LOCUTOR 09

LOCUTOR 10

Fig. A4.2b

TRAJETORIAS DOS LOCUTORES NA REDE SOM
PULSOS GERADOS A PARTIR DOS COEFICIENTES MFCC

LOCUTOR 01

LOCUTOR 02

LOCUTOR 03

LOCUTOR 04

LOCUTOR 05

LOCUTOR 06

Figura A4.3a

LOCUTOR 07

LOCUTOR 08

LOCUTOR 09

LOCUTOR 10

Fig. A4.3b

TRAJETORIAS DOS LOCUTORES NA REDE SOM
PULSOS GERADOS A PARTIR DOS COEF. ENERGIA FILTROS ESCALA MEL

LOCUTOR 01

LOCUTOR 02

LOCUTOR 03

LOCUTOR 04

LOCUTOR 05

LOCUTOR 06

Fig. A4.4a

LOCUTOR 07

LOCUTOR 08

LOCUTOR 09

LOCUTOR 10

Fig. A4.4b

APÊNDICE V

Resultados do classificador, com os conjuntos de treino e teste invertidos, visando constatar a robustez do treinamento.

Tab A5.1 – teste do classificador com um subconjunto de 10 amostras do conjunto de treinamento;

Tab. A5.2a e A5.2b – teste do classificador, utilizando o conjunto de teste com 70 amostras (antigo conjunto de treino).

Saídas correspondentes a cada locutor no
reconhecedor

Locutor	1	2	3	4	5	6	7	8	9	10
1	0,7386	0,0465	-0,0166	-0,0343	0,0398	-0,0151	0,0622	0,1017	0,0224	0,0022
2	0,0336	0,7924	0,0348	-0,0117	0,0193	0,0222	0,0548	-0,0341	-0,0577	0,0425
3	-0,0247	0,0360	0,7334	0,0395	-0,0332	0,1021	-0,0516	0,0412	0,1055	-0,0433
4	-0,0318	0,0118	0,0234	0,8363	-0,0217	0,0048	-0,0427	-0,0149	0,0780	-0,0445
5	0,0417	0,0411	-0,0458	-0,0231	0,7881	0,0351	0,0245	-0,0716	-0,0569	0,0752
6	-0,0128	0,0048	0,0990	0,0094	0,0601	0,7468	0,0766	0,0736	-0,0201	-0,0696
7	0,0617	0,0338	-0,0375	-0,0248	0,0195	0,0999	0,6814	0,1029	-0,0135	-0,0069
8	0,0757	-0,0061	0,0374	-0,0255	-0,0705	0,0332	0,0874	0,7179	0,0295	0,0497
9	0,0056	-0,0543	0,1050	0,0799	-0,0633	-0,0496	-0,0258	0,0151	0,7472	0,0202
10	0,0160	0,0329	-0,0495	-0,0497	0,0631	-0,0738	0,0225	0,0159	0,0228	0,8145

teste com conjunto de treino -
invertido

total 10
acertos 10
erros 0
taxa 100,00%

Tab. A5.1 – Resultado do classificador para o conjunto de treinamento (antigo teste).

Saídas correspondentes a cada locutor no
reconhecedor

Locutor	1	2	3	4	5	6	7	8	9	10
1	0,4456	-0,0276	-0,0472	-0,0163	-0,0132	0,3006	0,0340	0,0767	0,0740	0,0429
1	0,3417	0,0254	-0,0311	0,0425	0,0659	0,3685	0,0064	0,0072	0,1464	-0,0319
1	-0,0165	0,5696	0,0450	-0,0085	0,0440	-0,0008	0,0410	0,1796	-0,0124	-0,0510
1	-0,0130	0,0537	0,7981	-0,0071	-0,0146	-0,0172	-0,0028	0,1175	-0,0221	0,0130
1	0,1291	-0,0511	-0,0280	0,3962	0,1382	-0,0573	-0,0150	-0,0145	0,2617	0,0651
1	-0,0061	0,1217	0,0062	0,0246	0,5770	0,1136	0,0978	0,0077	-0,0389	0,0132
1	0,1035	0,0093	-0,0268	0,0174	0,0605	0,7117	0,0195	0,0247	0,0258	0,0113
2	0,1286	0,0703	-0,0219	-0,0065	0,0757	0,1912	0,3945	0,0001	0,0064	-0,0263
2	-0,0065	0,3270	0,2012	-0,0077	0,0127	-0,0028	-0,0123	0,2623	-0,0221	-0,0338
2	0,0193	-0,0523	-0,0426	0,0586	-0,0603	0,0437	0,0064	0,0280	0,5266	0,1660
2	0,2862	-0,0045	-0,0449	0,1047	0,1214	0,0930	0,1824	0,0377	-0,0371	0,0188
2	0,3533	0,0021	-0,0335	0,1519	0,1025	0,1491	0,0086	0,0054	0,1286	-0,0503
2	-0,0092	0,5782	0,0722	0,0019	0,0697	-0,0239	0,0630	0,1137	-0,0333	-0,0465
2	-0,0150	0,0670	0,7940	-0,0054	-0,0193	-0,0109	-0,0030	0,1130	-0,0264	0,0103
3	0,1952	-0,0218	-0,0247	0,1129	-0,0613	-0,0470	0,1753	-0,0477	0,3520	0,0345
3	0,0190	0,1596	0,0189	-0,0072	0,2921	0,0899	0,3732	-0,0322	-0,0432	0,0194
3	0,1985	0,0255	-0,0477	0,0838	0,1290	0,0530	0,2953	0,0415	-0,0261	-0,0188
3	0,0762	0,1073	-0,0042	0,0142	0,1198	0,1516	0,5155	-0,0362	-0,0250	-0,0138
3	-0,0195	0,3247	0,2173	-0,0254	-0,0067	-0,0066	0,0455	0,1991	-0,0262	-0,0315
3	0,0724	-0,0379	-0,0225	0,0303	-0,0634	0,0557	-0,0051	-0,0278	0,7411	0,0765
3	0,4271	0,0164	-0,0354	0,0222	-0,0175	0,0964	0,2638	-0,0142	0,0157	-0,0207
4	0,3281	-0,0025	-0,0309	0,1165	0,0216	0,1192	0,1097	0,0064	0,1468	-0,0602
4	-0,0224	0,5965	0,0639	0,0000	0,0534	0,0029	0,0475	0,1272	-0,0140	-0,0533
4	-0,0085	0,0470	0,8051	-0,0019	-0,0033	-0,0240	-0,0139	0,1049	-0,0288	0,0224
4	0,1513	-0,0321	-0,0425	0,4900	0,0075	-0,0667	0,0026	0,0133	0,2629	0,0129
4	0,0389	0,0778	-0,0178	0,0122	0,6658	0,0814	0,0383	0,0415	-0,0336	0,0392
4	0,2199	0,0292	-0,0199	0,0345	0,0486	0,3379	0,2249	-0,0267	-0,0274	-0,0067
4	0,0606	0,0609	-0,0295	0,0497	0,2529	0,0667	0,3618	0,0297	-0,0236	-0,0087
5	-0,0131	0,2004	0,1865	-0,0235	-0,0006	-0,0181	-0,0078	0,4160	-0,0105	-0,0308
5	-0,0011	-0,0415	-0,0362	0,0727	-0,0778	0,0277	0,0649	-0,0190	0,6358	0,0809
5	0,4764	-0,0397	-0,0495	0,0159	0,1810	0,0143	0,0555	0,0340	0,0036	0,1126
5	0,0616	-0,0437	-0,0104	0,0757	0,0278	0,3580	0,1074	-0,0096	0,2677	-0,0513
5	0,0037	0,6582	0,0204	0,0060	0,0967	-0,0125	0,0442	0,1401	-0,0267	-0,0444
5	-0,0215	0,0422	0,7624	-0,0106	-0,0155	-0,0106	-0,0036	0,1703	-0,0246	0,0175
5	0,1068	-0,0409	-0,0371	0,5555	0,0680	-0,0683	0,0214	0,0096	0,1435	0,0381
6	-0,0071	0,1041	0,0080	0,0184	0,5387	0,1042	0,2076	-0,0198	-0,0159	-0,0025
6	0,0475	0,0102	-0,0170	0,0484	0,2661	0,0880	0,4843	-0,0237	-0,0205	-0,0063
6	0,0542	0,1106	-0,0030	0,0346	0,0202	0,1630	0,6311	-0,0376	-0,0261	-0,0297
6	-0,0068	0,4403	0,2210	-0,0218	-0,0001	-0,0153	0,0492	0,1156	-0,0354	-0,0314
6	0,0861	-0,0405	-0,0308	0,0256	-0,0075	-0,0176	0,0032	-0,0144	0,6355	0,1124
6	0,4991	-0,0142	-0,0436	0,0973	-0,0162	0,2881	0,0020	0,0586	0,0193	0,0207
6	0,2800	0,0070	-0,0367	0,2431	0,1570	0,0850	0,0504	0,0373	0,0868	-0,0419

teste com conjunto de teste original -
invertido

Tab. A5.2a – Resultados para testes com o conjunto de teste (antigo treinamento).

Saídas correspondentes a cada locutor no reconhecedor

Locutor	1	2	3	4	5	6	7	8	9	10
7	0,0001	0,6575	0,0360	-0,0148	0,0506	0,0286	0,0398	0,1324	-0,0327	-0,0419
7	-0,0260	0,0403	0,8072	-0,0093	-0,0060	-0,0164	-0,0040	0,1085	-0,0244	0,0178
7	0,0946	-0,0154	-0,0205	0,2947	0,0280	-0,0053	-0,0167	-0,0285	0,4114	-0,0086
7	0,1227	0,0621	-0,0211	0,0522	0,3953	0,0411	0,1965	0,0074	-0,0364	0,0026
7	0,1132	0,0210	-0,0193	0,0164	0,0624	0,4073	0,1780	-0,0270	0,0533	-0,0320
7	0,0991	0,0429	-0,0279	0,0254	0,1949	0,1271	0,3615	0,0178	-0,0232	0,0002
7	-0,0184	0,2613	0,1410	-0,0288	-0,0220	-0,0217	0,0631	0,3444	-0,0235	-0,0269
8	0,0316	-0,0290	-0,0251	0,0746	-0,0705	-0,0031	0,1202	-0,0182	0,5909	0,0698
8	0,0772	0,0079	-0,0350	0,1285	-0,0306	0,5686	0,1355	0,0304	0,0115	-0,0259
8	0,0119	-0,0047	-0,0134	0,1986	0,0151	0,4058	0,1099	-0,0183	0,1405	-0,0552
8	-0,0176	0,6490	0,0324	-0,0070	0,0831	-0,0136	0,0839	0,1028	-0,0177	-0,0498
8	-0,0233	0,0570	0,8067	-0,0073	-0,0144	-0,0082	-0,0007	0,1128	-0,0234	0,0081
8	0,4590	-0,0098	-0,0331	0,1899	0,0083	0,1487	-0,0480	0,0249	0,1553	0,0535
8	0,0950	0,0411	-0,0155	0,0426	0,5507	-0,0180	0,2500	0,0006	-0,0301	0,0111
9	0,0497	0,1960	0,0118	0,0556	0,0096	0,2184	0,4981	-0,0423	-0,0401	-0,0205
9	0,0813	0,1267	-0,0093	0,0041	0,0595	0,1430	0,5853	-0,0223	-0,0388	0,0104
9	-0,0121	0,2699	0,1677	0,0039	-0,0010	-0,0299	0,0150	0,3574	-0,0394	-0,0219
9	0,0179	-0,0003	-0,0268	0,4314	-0,0392	0,2764	-0,0296	0,0320	0,2473	0,0497
9	0,3262	-0,0216	-0,0353	0,0903	0,0298	0,0199	0,2352	-0,0027	0,0607	-0,0230
9	0,3784	-0,0319	-0,0429	0,2310	0,1101	0,0395	0,0153	0,0368	0,0925	-0,0488
9	-0,0071	0,5889	0,0227	-0,0023	0,0711	-0,0035	0,0368	0,1809	-0,0093	-0,0498
10	-0,0110	0,0580	0,7815	-0,0077	-0,0125	-0,0158	-0,0089	0,1332	-0,0216	0,0118
10	0,0376	-0,0268	-0,0304	0,7517	0,0320	0,0159	-0,0258	0,0154	0,1529	-0,0118
10	0,0698	0,0376	-0,0237	0,0681	0,5633	-0,0099	0,2167	0,0316	-0,0424	0,0114
10	0,1059	0,0196	-0,0125	0,0338	0,1331	0,4065	0,2068	-0,0274	0,0088	-0,0203
10	0,1470	0,0512	-0,0276	0,0261	0,0771	0,0392	0,5746	-0,0061	-0,0370	-0,0090
10	-0,0263	0,2830	0,1530	-0,0261	0,0029	0,0052	0,0077	0,3681	-0,0309	-0,0196
10	0,0760	-0,0215	-0,0484	0,2359	-0,0671	0,0398	-0,0212	0,0361	0,4136	0,0826

teste com conjunto de teste original - invertido

Tab. A5.2b – Resultados para testes com o conjunto de teste (antigo treinamento).

APÊNDICE VI

Códigos em linguagem C

1. Rede de neurônios pulsados da camada1.
2. Rede de neurônios pulsados auto-organizável da camada2.

/*****

Titulo : CAMADA1
 Autor : Antonio Pedro Timoszczuk
 Data : 09/08/2003
 Versao : 0.0 - alterado em 12/10/2003

Descricao : Programa que implementa uma rede de uma camada com neuronios pulsados, segundo o modelo "spike response model".

Esta rede converte um vetor de entrada de dimensao N, em sequencias de pulsos para uso pelo classificador pulsado. Cada dimensao do vetor de entrada pode ser tratada por um ou mais neuronios, configuravel no arquivo de parametros da rede.

Os neuronios nao apresentam conexoes de realimentacao para os demais neuronios.

Os pesos uma vez inicializados sao constantes e positivos. Caso seja usado mais de um neuronio por dimensao do vetor de entrada, o peso de cada conexao varia. O objetivo desta estrategia e' fazer o scalling da entrada, melhorando assim a resolucao da conversao analogico/pulso.

Na leitura do arquivo de entrada, por conveniência, e' feita a conversao para valor absoluto, impedindo a entrada de valor negativo.

Nesta versao foi implementada a leitura de um arquivo de configuracao, que contem alguns parametros da rede.

12/10/2003

Retirado o decaimento exponencial da entrada do neuronio, devido receber valor "analógico", conversão e amostragem de sensor. Só a saída passa a ter constante de tempo.

Foi incluída a integracao de entradas em um unico neuronio, a quantidade das entradas é definida pela variavel [INTEGRACAO] contida no arquivo de configuracao

O novo arquivo de configuracao fica sendo:

- [DIM] - dimensao do vetor de entrada - por conseguinte da rede
- [INTEGRACAO] - numero de entradas integrado por neuronio
- [N_NEURONIO_POR_ENTRADA] - numero de neuronios da rede
- [OFFSET_DE_ENTRADA] - offset aplicado a entrada para garantir valores positivos
- [MAX_POTENCIAL_MEMB] - parametro que regula o valor máximo que fi pode assumir
- [CONSTANTE_TEMPO_Ti] - constante de relaxacao do filtro de entrada (para todas conexoes)
- [CONSTANTE_TEMPO_To] - constante de relaxacao do filtro de saida (para todos os neuronios)
- [LIMIAR_DE_DISPARO] - constante que indica o valor do threshold estatico, deve ser menor que Vp
- [PESO_REFRATARIO] - constante/peso da realimentacao da saida para o filtro de saida
- [PERIODO_REFRAT_ABS] - numero de ciclos do periodo refratario absoluto

Chamada : CAMADA1 arqIN arqCONF

```

*****/

#include <conio.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <process.h>

///////////////////////////////// Configuracoes //////////////////////////////////

#define Nparam 4 // Numero de parametros recebidos por main
#define Wi 1 // Peso da conexao de entrada
#define T 1.0 // Constante de tempo de base dos filtros

///////////////////////////////// Variaveis globais //////////////////////////////////

```

```

char nomearqIN[20]; // Nome do arquivo de entrada
FILE *fileIN;
char nomearqCONF[20]; // Nome do arquivo de configuracao
FILE *fileCONF;
char nomearqIN2[20]; // Nome do arquivo de entrada modificado apenas para salvar
FILE *fileIN2;
char nomearqOUT[20]; // Nome do arquivo com o resultado / saida
FILE *fileOUT;
char nomearqMEM[20]; // Nome do arquivo com o potencial de membrana
FILE *fileMEM;
char nomearqWR[20]; // Nome do arquivo com os pesos da conexao de entrada
FILE *fileWR;

int npontos; // Número de pontos do vetor de entrada
int nelelem; // Dimensao do vetor de entrada
int integracao; // Quantidade de entradas que sao agrupadas em um neuronio
int N_iteracoes; // Numero de iteracoes que a rede ira passar
int N_neuronios_porentrada; // Numero de neuronios por entrada
int N_neuronios_total; // Numero de neuronios total da rede

float Tin; // Constante de relaxacao do filtro de entrada
float Tou; // Constante de relaxacao do filtro de saida
float No; // Constante de realimentacao do filtro de saida
float Vp; // Constante que regula o valor maximo de fi
float Vt; // Constante que indica threshold estatico
float Offset; // Constante aplicada a entrada para garantir valores positivos
float Refrat; // Constante que determina o numero de ciclos do periodo refratario

float **Entrada; // Matriz que armazena a sequencia de entrada
float **Saida; // Matriz que armazena a sequencia resultante de saida
float **Membrana; // Matriz que armazena a evolucao do potencial de membrana
float **Wr; // Matriz que armazena os pesos da conexao de entrada

float *Pesos; // Armazena temporariamente a definicao dos pesos para montagem de Wr
float *Yd; // Armazena a saida da rede delayed para a proxima iteracao
float *Fid; // Armazena a saida do filtro de entrada para a proxima iteracao

```

```

float *Fod;      // Armazena a saida do filtro de saida para a proxima iteracao
float *Vabs;     // Armazena a constante referente ao periodo refratario absoluto

//////////////////////// Prototipos das funcoes //////////////////////////

void Le_entrada(void);
void P_neuronio(void);
void Salva_resu(void);

//////////////////////// Programa Principal //////////////////////////

int main(int argc, char *argv[])
{
    if (argc!=Nparam)
    {
        printf("Uso Correto : CAMADA1 arqIN arqCONF arqWR\n");
        printf("      arqIN : arquivo de entrada da rede\n");
        printf("      arqCONF      : arquivo de configuracao\n");
        printf("      arqWR : arquivo de pesos\n");
        return (0);
    }
    strcpy (nomearqIN , argv[1]);
    strcpy (nomearqIN2 , argv[1]);
    strcpy (nomearqOUT , argv[1]);
    strcpy (nomearqMEM , argv[1]);

    strcat (nomearqIN2, "INP");
    strcat (nomearqOUT, "OUT");
    strcat (nomearqMEM, "MEM");

    strcpy (nomearqCONF, argv[2]);
    strcpy (nomearqWR, argv[3]);

    printf ("Processando %s\n", nomearqIN);
    printf ("      inicializando...\n");
    Le_entrada( );

```

```

        printf ("    iniciando iteracoes...\n");
        P_neuronio( );
        printf ("    salvando resultados...\n");
        Salva_resu( );

    return (1);
}

//////////////////////////////// Equacoes do neuronio //////////////////////////////////

void P_neuronio(void)
{
    int h, i, j, k;
    float Xi;
    float Pi, Po;
    float Fi, Fo;
    float Ui, Y;

    for ( h=0; h<N_iteracoes; h++)
    {
        for ( i=0; i<(nelem-integracao+1); i++)
        {
            for ( j=0; j<N_neuronios_porentrada; j++ )
            {
                // fase de entrada

                Pi=0;

                for ( k=0; k<integracao; k++) // Aqui efetua a integracao em frequencia
                {
                    Xi=Entrada[h][i+k];
                    Pi=Wr[i][j]*Xi+Pi;
                }

                Po=No*Yd[i*N_neuronios_porentrada+j];
            }
        }
    }
}

```

```

// fase de filtragem
Fi=Pi+(exp(-Tin*T))*Fid[i*N_neuronios_porentrada+j];
if (Fi>=Vp) Fi=Vp; // Limita o valor maximo de Fi (no.de canais na celula e' limitado)
Fo=Po+(exp(-Tou*T))*Fod[i*N_neuronios_porentrada+j];
if (Fo>=Vp) Fo=Vp; // Limita o valor maximo de Fi (no.de canais na celula e' limitado)

// fase de saida
Ui=Fi-Fo-Vt;
Membrana[h][i*N_neuronios_porentrada+j]=Ui;

if ( Vabs[i*N_neuronios_porentrada+j]==0 )
{
    if ( Ui>=Vt )
    {
        Y=1;          // Ocorreu o disparo
        Vabs[i*N_neuronios_porentrada+j]=Refrat; // Seta o periodo refratario
                                                    no.de ciclos
    }
    else
    {
        Y=0;
        Vabs[i*N_neuronios_porentrada+j]=0;
    }
}
else
{
    Y=0;
    Vabs[i*N_neuronios_porentrada+j]=Vabs[i*N_neuronios_porentrada+j]-1;
}

```

absoluto,

```

        }

        Fid[i*N_neuronios_porentrada+j]=Fi;
        Fod[i*N_neuronios_porentrada+j]=Fo;
        Yd[i*N_neuronios_porentrada+j]=Y;
        Saida[h][i*N_neuronios_porentrada+j]=Y;
    }
}
return;
}

////////// Leitura dos arquivos de entrada e alocações de memória //////////

void Le_entrada(void)
{
    int h, i;
    float aux;

    // Leitura do arquivo de configuração

    char    parametro[20];
    float   valor;
    FILE    *stream;

    if ((stream = fopen(nomearqCONF, "rt")) == NULL)
    {
        fprintf(stderr, "Erro no arquivo %s.\n\n", nomearqCONF);
        exit(0);
    }

    while (!feof(stream))
    {
        fscanf(stream, "%s %f\n", parametro, &valor);
        if (strcmp (parametro, "DIM_ENTRADA") == 0) nelem=valor;
        if (strcmp (parametro, "INTEGRACAO") == 0) integracao=valor;
    }
}

```

```

        if (strcmp (parametro, "N_NEURONIO_POR_ENTRADA") == 0) N_neuronios_porentrada=valor;
        if (strcmp (parametro, "OFFSET_DE_ENTRADA") == 0) Offset=valor;
        if (strcmp (parametro, "MAX_POTENCIAL_MEMB") == 0) Vp=valor;
        if (strcmp (parametro, "CONSTANTE_TEMPO_Ti") == 0) Tin=valor;
        if (strcmp (parametro, "CONSTANTE_TEMPO_To") == 0) Tou=valor;
        if (strcmp (parametro, "LIMIAR_DE_DISPARO") == 0) Vt=valor;
        if (strcmp (parametro, "PESO_REFRATARIO") == 0) No=valor;
        if (strcmp (parametro, "PERIODO_REFRAT_ABS") == 0) Refrat=valor;
    }

    fclose (stream);

    N_neuronios_total=(nelem-integracao+1)*N_neuronios_porentrada;

    // Inicio da leitura do arquivo de entrada

    if (( fileIN=fopen(nomearqIN, "rt"))==NULL)
    {
        printf("Erro no arquivo %s\n\n", nomearqIN);
        exit(0);
    }

    fscanf(fileIN, "%f", &aux);
    npontos=aux;
    N_iteracoes=npontos+0; // Numero de iteracoes, neste caso igual ao numero de pontos

    // Aloca memoria para o vetor de entrada da rede

    if (( Entrada=(float **) calloc (N_iteracoes, sizeof(float *)))==NULL)
    {
        printf("Erro alocando Entradala - memoria insuficiente!\n");
        exit(0);
    }

    for ( h=0; h<N_iteracoes; h++)

```

```

{
    if (( Entrada[h]=(float *) calloc((nelem), sizeof(float )))==NULL)
    {
        printf("Erro alocando Entradalb - memoria insuficiente!\n");
        exit(0);
    }
}

// Inicializacao da matriz entrada

printf(" lendo entrada...\n");

for ( h=0; h<N_iteracoes; h++ )
{
    for ( i=0; i<nelem; i++ )
    {
        Entrada[h][i]=0;
    }
}

// Leitura do arquivo de entrada

for ( h=0; h<npontos; h++)
{
    for ( i=0; i<nelem; i++)
    {
        fscanf (fileIN, "%f", &aux);
        Entrada[h][i]=aux+Offset; // Aplica offset na entrada para nao existir valor negativo
    }
}
fclose (fileIN);

// Aloca memoria para o vetor de saida da rede

if (( Saida=(float **) calloc (N_iteracoes, sizeof(float *)))==NULL)
{

```



```

        printf("Erro alocando Saidala - memoria insuficiente!\n");
        exit(0);
    }

    for ( h=0; h<N_iteracoes; h++)
    {
        if ((Saida[h]=(float *) calloc((N_neuronios_total), sizeof(float )))==NULL)
        {
            printf("Erro alocando Saidalb - memoria insuficiente!\n");
            exit(0);
        }
    }

    for ( h=0; h<N_iteracoes; h++) // Inicializa matriz de saida
    {
        for ( i=0; i<N_neuronios_total; i++)
        {
            Saida[h][i]=0;
        }
    }

    // Aloca memoria para o vetor de potencial de membrana

    if (( Membrana=(float **) calloc (N_iteracoes, sizeof(float *)))==NULL)
    {
        printf("Erro alocando Membranala - memoria insuficiente!\n");
        exit(0);
    }

    for ( h=0; h<N_iteracoes; h++)
    {
        if ((Membrana[h]=(float *) calloc((N_neuronios_total), sizeof(float )))==NULL)
        {
            printf("Erro alocando Membranalb - memoria insuficiente!\n");
            exit(0);
        }
    }

```

```

}

for ( h=0; h<N_iteracoes; h++) // Inicializa matriz de potencial de membrana
{
    for ( i=0; i<N_neuronios_total; i++)
    {
        Membrana[h][i]=0;
    }
}

// Aloca memoria para o vetor de pesos referencia para montagem de Wr
if (( Pesos=(float *) calloc((N_neuronios_porentrada), sizeof(float )))==NULL)
{
    printf("Erro alocando Pesos - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_neuronios_porentrada; h++) Pesos[h]=0;

// Leitura do arquivo de pesos

if (( fileWR=fopen(nomearqWR, "rt"))==NULL)
{
    printf("Erro no arquivo %s\n\n", nomearqWR);
    exit(0);
}

printf(" lendo arquivo de pesos...\n");

for ( i=0; i<N_neuronios_porentrada; i++)
{
    fscanff (fileWR, "%f", &aux);
    Pesos[i]=aux;
}

```

```

fclose (fileWR);

// Aloca memoria para o vetor de pesos

if (( Wr=(float **) calloc (nelem, sizeof(float *)))==NULL)
{
    printf("Erro alocando vetor pesos Wr_1a - memoria insuficiente!\n");
    exit(0);
}

for ( h=0; h<nelem; h++)
{
    if ((Wr[h]=(float *) calloc((N_neuronios_porentada), sizeof(float )))==NULL)
    {
        printf("Erro alocando vetor pesos Wr_1b - memoria insuficiente!\n");
        exit(0);
    }
}

// Monta a matriz de pesos, segundo os valores lidos
// A serie de pesos é igual para cada entrada

printf("    montando matriz de pesos...\n");

for ( h=0; h<nelem; h++)
{
    for ( i=0; i<N_neuronios_porentada; i++)
    {
        Wr[h][i]=Pesos[i];
    }
}

// Aloca memoria para o vetor de saida delayed

```

```

if (( Yd=(float *) calloc((N_neuronios_total), sizeof(float )))==NULL)
{
    printf("Erro alocando Yd - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_neuronios_total; h++)
Yd[h]=0;

// Aloca memoria para o vetor de filtro de entrada delayed

if (( Fid=(float *) calloc((N_neuronios_total), sizeof(float )))==NULL)
{
    printf("Erro alocando Fid - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_neuronios_total; h++)      Fid[h]=0;

// Aloca memoria para o vetor de filtro de saida delayed

if (( Fod=(float *) calloc((N_neuronios_total), sizeof(float )))==NULL)
{
    printf("Erro alocando Fod - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_neuronios_total; h++) Fod[h]=0;

// Aloca memoria para o vetor Vabs do periodo refratario absoluto

if (( Vabs=(float *) calloc((N_neuronios_total), sizeof(float )))==NULL)
{
    printf("Erro alocando Vabs - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_neuronios_total; h++) Vabs[h]=0;

return;

```

```

}

////////// Salva resultado da rede //////////

void Salva_resu(void)
{
    int h, i, auxi;
    float aux;

    // Abre os arquivos de saida

    if ((fileIN2=fopen (nomearqIN2, "wt"))==NULL)
    {
        printf ("Erro no arquivo %s\n\n", nomearqIN2);
        exit(0);
    }

    if ((fileOUT=fopen (nomearqOUT, "wt"))==NULL)
    {
        printf ("Erro no arquivo %s\n\n", nomearqOUT);
        exit(0);
    }

    if ((fileMEM=fopen (nomearqMEM, "wt"))==NULL)
    {
        printf ("Erro no arquivo %s\n\n", nomearqMEM);
        exit(0);
    }

    // Aqui inicia a escrita dos arquivos de saida

    for ( h=0; h<N_iteracoes; h++) // Salva o vetor de entrada modificado
    {
        for ( i=0; i<nelem; i++ )

```

```

        {
            aux = Entrada[h][i];
            fprintf (fileIN2, "%f ", aux);
        }
        fprintf (fileIN2, "\n");
    }

    auxi = npontos; // Salva na primeira linha o numero de pontos do vetor
    fprintf (fileOUT, "%i\n", auxi);

    for ( h=0; h<N_iteracoes; h++) // Salva o potencial de membrana e a saida da rede
    {
        for ( i=0; i<N_neuronios_total; i++)
        {
            auxi = Saida[h][i];
            fprintf (fileOUT, "%i ", auxi);
            aux = Membrana[h][i];
            fprintf (fileMEM, "%f ", aux);
        }
        fprintf (fileOUT, "\n");
        fprintf (fileMEM, "\n");
    }

    // Libera memoria alocada para encerrar o programa

    for ( h=0; h<nelem; h++)
        free (Entrada[h]);
    free (Entrada);
    for ( h=0; h<N_iteracoes; h++)
        free (Saida[h]);
    free (Saida);
    for ( h=0; h<N_iteracoes; h++)
        free (Membrana[h]);
    free (Membrana);
    free (Pesos);
    for ( h=0; h<N_neuronios_porentrada; h++)

```

```
        free (Wr[h]);  
free (Wr);  
free (Yd);  
free (Fid);  
free (Fod);  
free (Vabs);  
  
fclose (fileIN2);  
fclose (fileOUT);  
fclose (fileMEM);  
return;  
}
```

Código em linguagem C que implementa a rede de neuronios pulsados auto-organizável da camada2.


```

/*****

```

```

Titulo      : CAMADA2
Autor       : Antonio Pedro Timoszczuk
Data        : 06/08/2003
Versao      : 0.0

```

```

Revisoes    : 14/09/2003 - APT

```

Descricao : Programa que implementa uma rede de uma camada com neuronios pulsados, segundo o modelo "spike response model" auto-organizável.

Esta rede usa o principio da auto-organizacao de Kohonen para aprender a estatistica das sequencias de pulso apresentadas a ela.

Cada neuronio desta rede, recebe todas as dimensoes do vetor de entrada e os neuronios nao apresentam conexoes de realimentacao para os demais neuronios.

A inicializacao dos pesos, e' feita a partir de um arquivo, e os pesos sao valores pequenos positivos ou negativos. Os pesos sao atualizados durante o treinamento e o arquivo da matriz de pesos e' atualizado ao final do processo de treino.

A atualizacao dos pesos e' feita apenas para o neuronio vencedor, ou seja, para aquele neuronio que disparou na iteracao. Caso mais de um neuronio tenha disparado, e' avaliado potencial de disparo de cada neuronio e o maior e' declarado vencedor e tem os seus pesos atualizados. O neuronio vencedor e' impedido de disparar nas proximas N_supress iteracoes (N_supress > Vabs).

A parametrizacao da rede e' feita por meio de um arquivo de configuracao, com os seguintes parametros:

```

Revisao     : 06/09/2003
Foi acrescentada a facilidade de salvamento intermediario dos pesos durante o treinamento, para permitir o acompanhamento da evolucao dos pesos. A periodicidade com que os pesos sao salvos, e definida no

```

arquivo de configuracao.

Foi acrescentado mais um nivel de saida. O atual passa a ser um estado interno da rede e a saida real passa a ser apenas a do neurônio com maior potencial de membrana e que tenha disparado, ira ser apresentado na saida. Agora sao gerados dois arquivos OUT (estado interno) e OUT1 (saida considerando apenas o neurônio vencedor).

14/09/2003

Foi feito uma revisao nas rotinas de alocação e dealocação de memoria, pois ocorria vazamento de memoria. As alocações foram divididas em tres grupos: as que dependem do tamanho do vetor de entrada, as que dependem do numero de neuronios e as especificas para o vetor de pesos.

A variacao dos pesos foi limitada a valores entre zero e um, pois com a estratégia de manter a faixa de valores dos pesos entre -1 e +1, fazia com que a rede oscilasse e nao diminuísse a energia durante o treinamento. Foi introduzido o conceito de avaliacao da energia da rede ao final de cada epoca como forma de avaliar a convergencia da mesma. O delta de energia entre iteracoes e salvo apos cada epoca.

Foi introduzido o coeficiente de aprendizado adaptativo e o valor fornecido no arquivo de configuracao, passou a ser o valor inicial do coeficiente de aprendizado. A adaptacao e feita segundo uma curva exponencial.

Foi incluída a vizinhança variavel, a qual segue o conceito básico das redes auto-organizaveis. A vizinhança é máxima no inicio do treinamento, sendo reduzida gradativamente até restringir-se ao neurônio vencedor ao final do treinamento.

Foram alteradas as estrategias de atualização dos pesos dos neuronios vizinhos, passando a usar decaimento exponencial. Foi alterada tambem a estrategia de tamanho da vizinhança, que passa a ser inicializada de forma proporcional ao numero de neuronios da rede e tem sua redução feita de forma exponencial.

Introduzido o criterio de parada do treinamento, por valor minimo de delta energia. Assim a rede para de treinar ou por energia minima ou por numero maximo de iteracoes, o que ocorre primeiro.

O novo arquivo de configuracoes fica:

```
[N_NEURONIOS] - numero de neuronios da rede
[DIM_ENTRADA] - dimensao do vetor de entrada (vetor de pulsos da camadal)
[MAX_POTENCIAL_MEMB] - valor maximo que pode ser assumido por fi ou fo
```

[CONSTANTE_TEMPO_Ti] - constante de tempo do filtro de entrada
 [CONSTANTE_TEMPO_To] - constante de tempo do filtro de saida
 [LIMIAIR_DE_DISPARO] - valor de potencial de membrana para ocorre o disparo
 [PESO_REFRATARIO] - peso do elo de realimentacao do limar dinamico de disparo
 [PERIODO_REFRAT_ABS] - periodo apos um disparo que o neuronio nao pode disparar
 [N_EPOCAS] - numero de epocas do treinamento
 [CRITERIO_DE_PARADA] - valor minimo do delta energia para interromper o treino
 [COEF_APRENDIZADO] - coeficiente de aprendizado dos pesos da rede
 [N_INTERVALO_SALVAPESOS] - intervalo de epocas para salvar pesos intermediarios

Chamada : CAMADA2 arqIN arqCONF arqPESOS MODO

*****/

```

#include <conio.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <process.h>

```

////////// Configuracoes //////////

```

#define Nparam 5 // Numero de parametros recebidos por main
#define Wi 1      // Peso da conexao de entrada
#define T 1.0     // Constante de tempo de base dos filtros

```

////////// Variaveis globais //////////

```

char nomearqIN[20]; // Nome do arquivo de entrada
FILE *fileIN;
char nomearqCONF[20]; // Nome do arquivo de configuracao
FILE *fileCONF;
char nomearqATUAL[20]; // Nome do arquivo com vetor de entrada corrente

```

```

FILE *fileATUAL;
char nomearqIN2[20]; // Nome do arquivo de entrada modificado apenas para salvar
FILE *fileIN2;
char nomearqOUT[20]; // Nome do arquivo com o resultado / saida interno
FILE *fileOUT;
char nomearqOUT1[20]; // Nome do arquivo com o resultado / saida externo
FILE *fileOUT1;
char nomearqMEM[20]; // Nome do arquivo com o potencial de membrana
FILE *fileMEM;
char nomearqWR[20]; // Nome do arquivo com os pesos da conexao de entrada
FILE *fileWR;
char nomearqEN[10]; // Nome do arquivo com a energia da rede durante o treino
FILE *fileEN;

char ch;
char MODO[2]; // Modo de operacao da rede: treinamento (TR) ou teste (TE)
char nomearqPESOSFINAL[20]; // Nome do arquivo de pesos final
char nomearqPESOSINTER[20]; // Nome do arquivo de pesos intermediario
char numeroARQPESO[10]; // String com o numero do arquivo de pesos intermediario atual

int tamanho; // Número de pontos do vetor de entrada
int dimensao; // Dimensao do vetor de entrada
int N_epocas; // Numero de epocas que a rede ira passar durante o treinamento
int N_iteracoes; // Numero de iteracoes que a rede ira passar para tratar um vetor de entrada
int N_neuronios; // Numero de neuronios da rede
int modo; // Modo de operacao da rede: 0=treinamento (TR) ou 1=teste (TE)
int N_intervalo_salvapesos; // Intervalo de epocas para salvar pesos intermediarios (0=nao salva
// pesos intermediarios
int Vzi; // Taxa de vizinhança inicial
int Vz; // Define a vizinhanca de atualizacao dos neuronios ao redor do neuronio vencedor
int taxa_Vz; // Define a taxa de atualizacao da vizinhanca, sendo proporcional ao numero de epocas
// do treinamento

float Alfa_inicial; // Valor inicial do coeficiente de aprendizado dos pesos
float Alfa; // Coeficiente de aprendizado dos pesos adaptado no tempo
float Criterio_de_parada; // Valor de delta energia minimo para terminar o treinamento

```

```

float delta_wr_max; // Maior variacao de peso registrada em uma iteracao
float peso; // Armazena temporariamente o valor do peso lido do arquivo Wr
float Tin; // Constante de relaxacao do filtro de entrada
float Tou; // Constante de relaxacao do filtro de saida
float No; // Constante de realimentacao do filtro de saida
float Vp; // Constante que regula o valor maximo de fi e fo
float Vt; // Constante que indica threshold estatico
float Refrat; // Constante que determina o numero de ciclos do periodo refratario

float **Entrada; // Matriz que armazena a sequencia de entrada
float **Saida; // Matriz que armazena a sequencia resultante de saida interna
float **Saida_ext; // Matriz que armazena a sequencia resultante de saida da rede (unico disparo)
float **Membrana; // Matriz que armazena a evolucao do potencial de membrana
float **Wr; // Matriz que armazena os pesos da conexao de entrada

float *Yd; // Armazena a saida da rede delayed para a proxima iteracao
float *Fid; // Armazena a saida do filtro de entrada para a proxima iteracao
float *Fod; // Armazena a saida do filtro de saida para a proxima iteracao
float *Membrana_aux; // Armazena o potencial de membrana atual para uso no treinamento
float *Membrana_auxdisp; // Armazena o potencial de membrana atual apenas dos neuronios que dispararam
// usado para determinar a saida real
float *Vabs; // Armazena a constante referente ao periodo refratario absoluto
float *Energia; // Armazena a energia da rede a cada epoca de treinamento

////////// Prototipos das funcoes //////////////////////////////////////

void Aloca_matrizpesos(void);
void Aloca_memo(void);
void Atualiza_pesos(int iteracao);
void Inicializa_pesos(void);
void Le_entrada(void);
void Le_pesos(void);
void Libera_entrada(void);
void Libera_memo(void);
void P_neuronio(void);

```

```

void Salva_pesos(void);
void Salva_resu(void);
void Treina_rede(void);
void Testa_rede(void);

////////// Programa Principal //////////

int main(int argc, char *argv[])
{
    if (argc!=Nparam)
    {
        printf("Uso Correto : CAMADA2 arqIN arqCONF arqPESOS MODO\n");
        printf("    arqIN : arquivo de entrada da rede\n");
        printf("    arqCONF : arquivo de configuracao\n");
        printf("    arqWR : arquivo de pesos\n");
        printf("    MODO : TR=treinamento ou TE=teste\n");
        return (0);
    }
    strcpy (nomearqIN , argv[1]);

    strcpy (nomearqCONF, argv[2]);
    strcpy (nomearqPESOSFINAL, argv[3]);
    strcpy (MODO, argv[4]);

    if (strcmp (MODO, "TR") == 0) modo=0;
    if (strcmp (MODO, "TE") == 0) modo=1;

    // Leitura do arquivo de configuracao

    char parametro[25];
    float valor;
    FILE *stream;

    if ((stream = fopen(nomearqCONF, "rt")) == NULL)
    {
        fprintf(stderr, "Erro no arquivo %s.\n\n", nomearqCONF);
    }

```

```

        exit(0);
    }

    while (!feof(stream))
    {
        fscanf(stream, "%s %f\n", parametro, &valor);
        if (strcmp (parametro, "N_NEURONIOS") == 0) N_neuronios=valor;
        if (strcmp (parametro, "DIM_ENTRADA") == 0) dimensao=valor;
        if (strcmp (parametro, "MAX_POTENCIAL_MEMB") == 0) Vp=valor;
        if (strcmp (parametro, "CONSTANTE_TEMPO_Ti") == 0) Tin=valor;
        if (strcmp (parametro, "CONSTANTE_TEMPO_To") == 0) Tou=valor;
        if (strcmp (parametro, "LIMIAR_DE_DISPARO") == 0) Vt=valor;
        if (strcmp (parametro, "PESO_REFRATARIO") == 0) No=valor;
        if (strcmp (parametro, "PERIODO_REFRAT_ABS") == 0) Refrat=valor;
        if (strcmp (parametro, "N_EPOCAS") == 0) N_epocas=valor;
        if (strcmp (parametro, "CRITERIO_DE_PARADA") == 0) Criterio_de_parada=valor;
        if (strcmp (parametro, "COEF_APRENDIZADO") == 0) Alfa_inicial=valor;
        if (strcmp (parametro, "N_INTERVALO_SALVAPESOS") == 0) N_intervalo_salvapesos=valor;
    }

    fclose (stream);

    if (modo == 0) Treina_rede( );

    if (modo == 1) Testa_rede( );

    return (1);
}

////////// Treinamento da rede //////////////////////////////////////

void Treina_rede(void)
{
    int h, i, j, k, ctl_Vz;
    float aux, energia_anterior, delta_energia;

```

```

strcpy(nomearqEN, "energia");
energia_anterior=0;
delta_energia=0;
delta_wr_max=100000; // Valor grande para nao encerrar treinamento quando nao ocorrer disparo

Vzi=(N_neuronios/2)-1; // Define a vizinhanca inicial, proporcional ao numero de neuronios
Vz=Vzi;//1;//Vzi;

Aloca_matrizpesos( );
Inicializa_pesos( );
Aloca_memo( );

if ((fileIN = fopen(nomearqIN, "rt")) == NULL)
{
    fprintf(stderr, "Erro no arquivo %s.\n\n", nomearqIN);
    exit(0);
}

i=1;
for ( h=0; h<N_epocas; h++)
{
    Alfa=Alfa_inicial*exp(-0.007*h);
    i=i-1;
    if ( N_intervalo_salvapesos !=0 )
    {
        if ( i==0)
        {
            _itoa( h, numeroARQPESO, 10 );
            strcpy(nomearqWR, nomearqPESOSFINAL); // Aqui vai salvar os pesos intermediarios
            strcat(nomearqWR, numeroARQPESO);
            Salva_pesos( );
            i=N_intervalo_salvapesos;
        }
    }
}

```



```

printf ("Treinando epoca %i\n", h);

while ((fscanf(fileIN, "%s", nomearqATUAL)) == 1)
{
    Le_entrada( );
    printf ("Processando arquivo %s tamanho %i\r", nomearqATUAL, N_iteracoes);
    P_neuronio( );
    Libera_entrada( );
}
fseek( fileIN, 0L, SEEK_SET);

// Calculo da energia da rede, para avaliar a convergencia

Energia[h]=0;
for ( j=0; j<N_neuronios; j++)
{
    for ( k=0; k<dimensao; k++)
    {
        Energia[h]=Energia[h] +(Wr[j][k]*Wr[j][k]);
    }
}

if ( h == 0 ) energia_anterior=Energia[h];

delta_energia=fabs(Energia[h]-energia_anterior);

energia_anterior=Energia[h];

Energia[h]=delta_energia;

printf("\nDelta energia da rede= %f\n", Energia[h]);

Vz=Vzi*exp(-20*((float)h/(float)N_epocas));

if ((delta_energia < Criterio_de_parada) && h != 0) h=N_epocas; // Forca a para do treinamento se
delta_energia

```

```

minimo.
}
fclose(fileIN);
strcpy(nomearqWR, nomearqPESOSFINAL); // Aqui vai salvar os pesos finais apos treino
Salva_pesos( );

for ( h=0; h<N_neuronios; h++) // A matriz de pesos so e liberada ao final do teste
    free (Wr[h]);
free (Wr);

// Salva a variacao da energia da rede durante o treino

if ((fileEN=fopen (nomearqEN, "wt"))==NULL)
{
    printf ("Erro no arquivo %s\n\n", nomearqEN);
    exit(0);
}

for ( h=0; h<N_epocas; h++)
{
    aux = Energia[h];
    fprintf (fileEN, "%f\n", aux);
}
fclose (fileEN);
printf ("Final de treinamento\n");
Libera_memo( );
return;
}

////////// Teste da rede //////////

void Testa_rede(void)
{
    int h;

```

```

Aloca_matrizpesos( );
Aloca_memo( );
Le_pesos( );

if ((fileIN = fopen(nomearqIN, "rt")) == NULL)
{
    fprintf(stderr, "Erro no arquivo %s.\n\n", nomearqIN);
    exit(0);
}

while ((fscanf(fileIN, "%s", nomearqATUAL)) == 1)
{
    strcpy (nomearqIN2 , nomearqATUAL);
    strcpy (nomearqOUT , nomearqATUAL);
    strcpy (nomearqOUT1 , nomearqATUAL);
    strcpy (nomearqMEM , nomearqATUAL);

    strcat (nomearqIN2, "INP");
    strcat (nomearqOUT, "OUT");
    strcat (nomearqOUT1, "OUT1");
    strcat (nomearqMEM, "MEM");

    Le_entrada( );
    printf ("Processando arquivo %s tamanho %i\n", nomearqATUAL, N_iteracoes);
    //printf ("Iniciando iteracoes...\n");
    P_neuronio( );
    //printf ("Salvando resultados...\n");
    Salva_resu( );
    Libera_entrada( );
}

fclose(fileIN);

for ( h=0; h<N_neuronios; h++) // A matriz de pesos so e liberada ao final do teste
    free (Wr[h]);

```

```

    free (Wr);
    Libera_memo( );
    printf ("Final do teste\n");
    return;
}

///// Equacoes do neuronio no teste da rede //////////////////////////////////////

void P_neuronio(void)
{
    int h, i, j, k, vencedor;
    float membrana_max;
    float Xi;
    float Pi, Po;
    float Fi, Fo;
    float Ui, Y;

    for ( h=0; h<N_iteracoes; h++)
    {
        for ( i=0; i<N_neuronios; i++)
        {
            // fase de entrada

            Pi=0;

            for ( j=0; j<dimensao; j++ )
            {
                Xi=Entrada[h][j];

                Pi=Wr[i][j]*Xi+Pi;
            }

            Po=No*Yd[i];

            // fase de filtragem

```

```

Fi=Pi+(exp(-Tin*T))*Fid[i];

if (Fi>=Vp) Fi=Vp; // Limita o valor maximo de Fi (no.de canais na celula e' limitado)

Fo=Po+(exp(-Tou*T))*Fod[i];

if (Fo>=Vp) Fo=Vp; // Limita o valor maximo de Fi (no.de canais na celula e' limitado)

// fase de saida

Ui=Fi-Fo-Vt;

Membrana[h][i]= Ui;

if ( Vabs[i]==0 )
{
    if ( Ui>=Vt )
    {
        Y=1;          // Ocorreu o disparo
        Vabs[i]=Refrat; // Seta o periodo refratario absoluto, no.de ciclos
        Membrana_auxdisp[i]=Ui;
    }
    else
    {
        Y=0;
        Vabs[i]=0;
        Membrana_auxdisp[i]=0;
    }
}
else
{
    Y=0;
    Vabs[i]=Vabs[i]-1;
    Membrana_auxdisp[i]=0;
}

```

```

        Fid[i]=Fi;
        Fod[i]=Fo;
        Membrana_aux[i]=Ui;
        Yd[i]=Y;
        Saida[h][i]=Y;
    }
    //printf ("%d \r", h);

    if (modo == 0) Atualiza_pesos(h);

    if (modo == 1)
    {
        // Determina dos neuronios que dispararam qual apresentou o maior potencial de disparo e o
declara vencedor

        membrana_max=0;
        vencedor=0; // Vencedor igual a zero = nao houve vencedor

        for ( j=0; j<N_neuronios; j++)
        {
            if ( Yd[j]==1 && Membrana_auxdisp[j]>membrana_max)
            {
                membrana_max=Membrana_auxdisp[j];
                vencedor=j+1;
            }
        }

        for ( j=0; j<N_neuronios; j++) Saida_ext[h][j]=0;

        if ( vencedor != 0 ) Saida_ext[h][vencedor-1]=1;
    }
}
// printf ("\n");
return;
}

```

```

////////// Aloca memoria para matriz de pesos //////////

void Aloca_matrizpesos(void)
{
    int h;

    if (( Wr=(float **) calloc (N_neuronios, sizeof(float *)))==NULL)
    {
        printf("Erro alocando vetor pesos Wr_1a - memoria insuficiente!\n");
        exit(0);
    }

    for ( h=0; h<N_neuronios; h++)
    {
        if ((Wr[h]=(float *) calloc((dimensao), sizeof(float )))==NULL)
        {
            printf("Erro alocando vetor pesos Wr_1b - memoria insuficiente!\n");
            exit(0);
        }
    }
    return;
}

////////// Inicializa matriz de pesos para treinamento //////////

void Inicializa_pesos(void)
{
    int h, l;
    float aux;
    time_t t;

    srand((unsigned) time(&t));

```

```

for ( h=0; h<N_neuronios; h++ )
{
    for ( l=0; l<dimensao; l++ )
    {
        aux=(( (float)(rand() % 20001)) - 10000.0)/10000.0);

        aux=aux/10;

        Wr[h][l]=fabs(aux); // Armazena o peso na matriz, agora so valores positivos
    }
}
return;
}

```

////////// Leitura do arquivo de pesos //////////

```

void Le_pesos(void)
{
    int h, i;
    float aux;

    strcpy(nomearqWR, nomearqPESOSFINAL);

    // Monta a matriz de pesos, segundo os valores lidos do arquivo de pesos

    if (( fileWR=fopen(nomearqWR, "rt"))==NULL)
    {
        printf("Erro no arquivo %s\n\n", nomearqWR);
        exit(0);
    }

    printf("Montando matriz de pesos...\n");

    for ( h=0; h<N_neuronios; h++)
    {

```



```

        for ( i=0; i<dimensao; i++)
        {
            fscanf (fileWR, "%f", &aux);
            peso=aux;
            Wr[h][i]=peso;
        }
    }
    fclose (fileWR);
    return;
}

////////// Salva arquivo de pesos apos treinamento //////////

void Salva_pesos(void)
{
    int h, i;
    float aux;

    printf("\nSalvando pesos...\n");

    if ((fileWR=fopen (nomearqWR, "wt"))==NULL)
    {
        printf ("Erro no arquivo %s\n\n", nomearqWR);
        exit(0);
    }

    for ( h=0; h<N_neuronios; h++)
    {
        for ( i=0; i<dimensao; i++ )
        {
            aux = Wr[h][i];
            fprintf (fileWR, "%f ", aux);
        }
        fprintf (fileWR, "\n");
    }
}

```

```

        fclose (fileWR);
        return;
    }

    ///// Atualiza matriz de pesos durante o treinamento //////////////////////////////////

void Atualiza_pesos(int iteracao)
{
    int h, i, neuronio, vencedor;
    float delta_wr, membrana_max, normaliza, aux;

    membrana_max=Membrana_aux[0];
    vencedor=0; // Vencedor igual a zero = nao houve vencedor
    normaliza=0;

    // Determina o neuronio que apresentou o maior potencial de disparo e o declara vencedor

    // Primeiramente busca neuronios que dispararam

    for ( h=0; h<N_neuronios; h++)
    {
        if ( Yd[h]==1 && Membrana_aux[h]>=membrana_max)
        {
            membrana_max=Membrana_aux[h];
            vencedor=h+1;
        }
    }

    if ( vencedor!=0)
    {
        // Correcao dos pesos do neuronio vencedor

        for ( i=0; i<dimensao; i++)
        {

```

```

delta_wr= Alfa*(Entrada[iteracao][i]-Wr[vencedor-1][i]);

delta_wr_max=fabs(delta_wr); // Inicializa valor de delta pesos maximo para esta iteracao

    Wr[vencedor-1][i]=Wr[vencedor-1][i]+delta_wr;

    if ( fabs(Wr[vencedor-1][i]) >= 1) normaliza=1;
}

if ( Vz!=0 )
{
    // Correcao dos pesos dos neuronios a esquerda do neuronio vencedor

    neuronio=vencedor-2;

    if ( neuronio<0 ) neuronio=N_neuronios-1;

    for ( h=0; h<Vz; h++)
    {
        for ( i=0; i<dimensao; i++)
        {
            delta_wr= Alfa*(Entrada[iteracao][i]-Wr[neuronio][i])*Entrada[iteracao][i]*exp(-
5*(float)h/(float)Vz);

            Wr[neuronio][i]=Wr[neuronio][i]+delta_wr; // antes era subtracao

            if (fabs(delta_wr) > delta_wr_max) delta_wr_max= fabs(delta_wr); // Acumula delta wr
maximo

            if ( fabs(Wr[neuronio][i]) >= 1) normaliza=1;

            if ( Wr[neuronio][i] <= 0) Wr[neuronio][i]=0;
        }
    }

    if ( neuronio==0 ) neuronio=N_neuronios;
}

```

```

        neuronio=neuronio-1;
    }

    // Correcao dos pesos dos neuronios a direita do neuronio vencedor
    neuronio=vencedor;

    if ( neuronio>N_neuronios-1 ) neuronio=0;

    for ( h=0; h<Vz; h++)
    {
        for ( i=0; i<dimensao; i++)
        {
            delta_wr= Alfa*(Entrada[iteracao][i]-Wr[neuronio][i])*exp(-5*(float)h/(float)Vz);

            Wr[neuronio][i]=Wr[neuronio][i]+delta_wr; // antes era subtracao

            if (fabs(delta_wr) > delta_wr_max) delta_wr_max= fabs(delta_wr); // Acumula delta wr

            if ( fabs(Wr[neuronio][i]) >= 1) normaliza=1;

            if ( Wr[neuronio][i] <= 0) Wr[neuronio][i]=0;
        }

        if ( neuronio==N_neuronios-1 ) neuronio=-1;

        neuronio=neuronio+1;
    }
}

// Normalizacao dos pesos pelo maior valor do peso, caso ocorra variacao brusca nos pesos
if ( normaliza==1)
{

```

maximo

```

        aux=0;

        for ( h=0; h<N_neuronios; h++)
        {
            for ( i=0; i<dimensao; i++)
            {
                if ( fabs(Wr[h][i]) > aux ) aux=fabs(Wr[h][i]); // procura o maior valor em modulo
            }

            for ( h=0; h<N_neuronios; h++)
            {
                for ( i=0; i<dimensao; i++)
                {
                    Wr[h][i]=Wr[h][i]/(2*aux);
                }
            }

            printf ("\nnormalizacao efetuada\n");
        }

        return;
    }

}

///// Leitura dos arquivos de entrada e alocações de memoria ///////////

// Le o arquivo de entradas e aloca memoria para as matrizes que dependem do
// tamanho do vetor de entrada: entrada, saida e membrana.

void Le_entrada(void)
{
    int h, i;
    float aux;

```

```

// Inicio da leitura do arquivo de entrada

if (( fileATUAL=fopen(nomearqATUAL, "rt"))==NULL)
{
    printf("Erro no arquivo %s\n\n", nomearqATUAL);
    exit(0);
}

fscanf(fileATUAL, "%f", &aux);
tamanho=aux;
N_iteracoes=tamanho+0; // Numero de iteracoes, neste caso igual ao numero de pontos

// Aloca memoria para o vetor de entrada da rede

if (( Entrada=(float **) calloc (N_iteracoes, sizeof(float *)))==NULL)
{
    printf("Erro alocando Entradala - memoria insuficiente!\n");
    exit(0);
}

for ( h=0; h<N_iteracoes; h++)
{
    if (( Entrada[h]=(float *) calloc((dimensao), sizeof(float )))==NULL)
    {
        printf("Erro alocando Entradalb - memoria insuficiente!\n");
        exit(0);
    }
}

// Inicializacao da matriz entrada

//printf("Lendo entrada...\n");

for ( h=0; h<N_iteracoes; h++ )

```

```

{
    for ( i=0; i<dimensao; i++ )
    {
        Entrada[h][i]=0;
    }
}

// Leitura do arquivo de entrada

for ( h=0; h<tamanho; h++)
{
    for ( i=0; i<dimensao; i++)
    {
        fscanf (fileATUAL, "%f", &aux);
        Entrada[h][i]=aux;
    }
}
fclose (fileATUAL);

// Aloca memoria para o vetor de saida da rede interno

if (( Saida=(float **) calloc (N_iteracoes, sizeof(float *)))==NULL)
{
    printf("Erro alocando Saidala - memoria insuficiente!\n");
    exit(0);
}

for ( h=0; h<N_iteracoes; h++)
{
    if ((Saida[h]=(float *) calloc((N_neuronios), sizeof(float )))==NULL)
    {
        printf("Erro alocando Saidalb - memoria insuficiente!\n");
        exit(0);
    }
}

```

```

for ( h=0; h<N_iteracoes; h++)    // Inicializa matriz de saida interna
{
    for ( i=0; i<N_neuronios; i++)
    {
        Saida[h][i]=0;
    }
}

// Aloca memoria para o vetor de saida da rede externa

if (( Saida_ext=(float **) calloc (N_iteracoes, sizeof(float *)))==NULL)
{
    printf("Erro alocando Saida_extla - memoria insuficiente!\n");
    exit(0);
}

for ( h=0; h<N_iteracoes; h++)
{
    if ((Saida_ext[h]=(float *) calloc((N_neuronios), sizeof(float )))==NULL)
    {
        printf("Erro alocando Saida_extlb - memoria insuficiente!\n");
        exit(0);
    }
}

for ( h=0; h<N_iteracoes; h++)    // Inicializa matriz de saida externa
{
    for ( i=0; i<N_neuronios; i++)
    {
        Saida_ext[h][i]=0;
    }
}

```



```

    // Aloca memoria para o vetor de potencial de membrana

if (( Membrana=(float **) calloc (N_iteracoes, sizeof(float *)))==NULL)
{
    printf("Erro alocando Membranala - memoria insuficiente!\n");
    exit(0);
}

for ( h=0; h<N_iteracoes; h++)
{
    if ((Membrana[h]=(float *) calloc((N_neuronios), sizeof(float )))==NULL)
    {
        printf("Erro alocando Membranalb - memoria insuficiente!\n");
        exit(0);
    }
}

for ( h=0; h<N_iteracoes; h++)    // Inicializa matriz de potencial de membrana
{
    for ( i=0; i<N_neuronios; i++)
    {
        Membrana[h][i]=0;
    }
}

return;
}

//////// Libera a memoria alocada na leitura de entrada////////

// Libera memoria alocada para encerrar teste com o vetor de entrada
// Notar que a matriz de pesos tem tratamento a parte e so se liberada ao
// final do teste pois os pesos sao os mesmos para cada um dos vetores de
// treino ou de teste

```

```

void Libera_entrada(void)
{
    int h, i;

    for ( h=0; h<N_iteracoes; h++)
        free (Entrada[h]);
    free (Entrada);
    for ( h=0; h<N_iteracoes; h++)
        free (Saida[h]);
    free (Saida);
    for ( h=0; h<N_iteracoes; h++)
        free (Saida_ext[h]);
    free (Saida_ext);
    for ( h=0; h<N_iteracoes; h++)
        free (Membrana[h]);
    free (Membrana);

    return;
}

////////// Salva resultado da rede //////////

void Salva_resu(void)
{
    int h, i;
    float aux;

    printf("\nSalvando resultado...\n");

    // Abre os arquivos de saida

    if ((fileIN2=fopen (nomearqIN2, "wt"))==NULL)
    {
        printf ("Erro no arquivo %s\n\n", nomearqIN2);
    }
}

```

```

        exit(0);
    }

    if ((fileOUT=fopen (nomearqOUT, "wt"))==NULL)
    {
        printf ("Erro no arquivo %s\n\n", nomearqOUT);
        exit(0);
    }

    if ((fileOUT1=fopen (nomearqOUT1, "wt"))==NULL)
    {
        printf ("Erro no arquivo %s\n\n", nomearqOUT1);
        exit(0);
    }

    if ((fileMEM=fopen (nomearqMEM, "wt"))==NULL)
    {
        printf ("Erro no arquivo %s\n\n", nomearqMEM);
        exit(0);
    }

    // Aqui inicia a escrita dos arquivos de saida

    for ( h=0; h<N_iteracoes; h++) // Salva o vetor de entrada modificado
    {
        for ( i=0; i<dimensao; i++ )
        {
            aux = Entrada[h][i];
            fprintf (fileIN2, "%f ", aux);
        }
        fprintf (fileIN2, "\n");
    }

    //
    aux = tamanho;
    fprintf (fileOUT, "%i\n", tamanho);
    fprintf (fileOUT1, "%i\n", tamanho);

```

```

for ( h=0; h<N_iteracoes; h++) // Salva o potencial de membrana e a saida da rede
{
    for ( i=0; i<N_neuronios; i++)
    {
        aux = Saida[h][i];
        fprintf (fileOUT, "%f ", aux);
        aux = Saida_ext[h][i];
        fprintf (fileOUT1, "%f ", aux);
        aux = Membrana[h][i];
        fprintf (fileMEM, "%f ", aux);
    }
    fprintf (fileOUT, "\n");
    fprintf (fileOUT1, "\n");
    fprintf (fileMEM, "\n");
}

fclose (fileIN2);
fclose (fileOUT);
fclose (fileOUT1);
fclose (fileMEM);
return;
}

//////// Aloca memoria para vetores internos //////////

// Aloca memoria para os diversos vetores usados para os calculos
// do estado da rede e que nao dependem do tamanho do vetor de entrada.
// Estes vetores e matrizes dependem apenas da arquitetura da rede, ou seja,
// do numero de neuronios.

void Aloca_memo(void)
{
    int h, i;

```

```

// Aloca memoria para o vetor de potencial de membrana_aux

if (( Membrana_aux=(float *) calloc((N_neuronios), sizeof(float )))==NULL)
{
    printf("Erro alocando Membrana_aux - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_neuronios; h++)
    Membrana_aux[h]=0;

// Aloca memoria para o vetor de potencial de membrana_auxdisp

if (( Membrana_auxdisp=(float *) calloc((N_neuronios), sizeof(float )))==NULL)
{
    printf("Erro alocando Membrana_auxdisp - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_neuronios; h++)
    Membrana_auxdisp[h]=0;

// Aloca memoria para o vetor de saida delayed

if (( Yd=(float *) calloc((N_neuronios), sizeof(float )))==NULL)
{
    printf("Erro alocando Yd - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_neuronios; h++)
    Yd[h]=0;

// Aloca memoria para o vetor de filtro de entrada delayed

if (( Fid=(float *) calloc((N_neuronios), sizeof(float )))==NULL)

```

```

{
    printf("Erro alocando Fid - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_neuronios; h++)
    Fid[h]=0;

// Aloca memoria para o vetor de filtro de saida delayed

if (( Fod=(float *) calloc((N_neuronios), sizeof(float )))==NULL)
{
    printf("Erro alocando Fod - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_neuronios; h++)
    Fod[h]=0;

// Aloca memoria para o vetor Vabs do periodo refratario absoluto

if (( Vabs=(float *) calloc((N_neuronios), sizeof(float )))==NULL)
{
    printf("Erro alocando Vabs - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_neuronios; h++)
    Vabs[h]=0;

// Aloca memoria para o vetor Energia da rede durante o treinamento

if (( Energia=(float *) calloc((N_epocas), sizeof(float )))==NULL)
{
    printf("Erro alocando Energia - memoria insuficiente!\n");
    exit(0);
}
for ( h=0; h<N_epocas; h++)
    Energia[h]=0;

```

```
        return;
    }

    //////////// Libera a memoria alocada ////////////

    // Libera memoria alocada para encerrar teste com o vetor de entrada
    // Notar que a matriz de pesos so sera liberada ao final do teste
    // pois os pesos sao os mesmos para cada um dos vetores de teste

void Libera_memo(void)
{
    int h, i;

    free (Membrana_aux);
    free (Membrana_auxdisp);
    free (Yd);
    free (Fid);
    free (Fod);
    free (Vabs);
    free (Energia);

    return;
}
```

APÊNDICE VII

Resultados de testes de desenvolvimento da rede.

Teste 08

Neste teste é usada a estrutura de redes pulsadas, em conjunto com os coeficientes MFCC, para verificar se as redes pulsadas podem gerar parametros representativos do locutor.

A estrutura da rede pulsada, é em duas camadas, sendo a primeira uma camada codificadora da entrada para uma sequencia de pulsos. A segunda camada consiste de uma rede auto-organizada de uma dimensão que visa montar uma representação do locutor.

A decisão final sobre a identificação é feita por meio de um MLP.

A pasta ícones, é usada para gerar vetores que representam o numero de disparos de cada neuron, ao longo de uma locução.

Em MLP00 é feito um teste com uma única rede treinada para reconhecer os ícones. Os ícones correspondem ao numero de disparos de cada neuron da rede auto-organizável, ao longo da locução.

A pasta frames é usada para, a partir dos vetores gerados pela camada2, reorganiza-los segundo o conceito das mtis, e submete-los ao treinamento / identificação através do mlp.

Camada 1 – parâmetros iniciais ConfigC1

```

DIM_ENTRADA      16
N_NEURON_POR_ENTRADA 1
OFFSET_DE_ENTRADA 1
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 1.0
LIMIAR_DE_DISPARO 0.52
PESO_REFRATARIO   1
PERIODO_REFRAT_ABS 1
  
```

Camada 2 – parâmetros iniciais Config C2

```

N_NEURONS        100
DIM_ENTRADA      16
MAX_POTENCIAL_MEMB 40
CONSTANTE_TEMPO_Ti 1.0
  
```


CONSTANTE_TEMPO_To 0.1
 LIMIAR_DE_DISPARO 0.2
 PESO_REFRATARIO 1
 PERIODO_REFRAT_ABS 1
 N_EPOCAS 500
 COEF_APRENDIZADO 0.001
 N_INTERVALO_SALVAPESOS 0

Resultados alterando parâmetros da camada2:

Ti=10 e To=1 - resultado para conjunto 2 = 82%

Ti=0,5 e To=0,1 - resultado para conjunto 2 = 71,5%

Ti=0,5 e To=2,0 - resultado para conjunto 2 = 76,5%

Ti=0,5 e To=0,5 - resultado para conjunto 2 = 78,5%

Ti=0,5 e To=10 - resultado para conjunto 2 = 77,5%

Ti=1,0 e To=1,0 - resultado para conjunto 2 = 79,5%

Teste 09

Teste de resposta da camada1 com a variação de parâmetros.

Considerando L01F011 – energia dos filtros segundo a escala Mel

configuração inicial

```

DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 1
OFFSET_DE_ENTRADA  0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 1.0
LIMIAR_DE_DISPARO 2.0
PESO_REFRATARIO   1
PERIODO_REFRAT_ABS 1
  
```

aumentando T_o para 10.0

```

DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 1
OFFSET_DE_ENTRADA 0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 10.0
LIMIAR_DE_DISPARO 2.0
PESO_REFRATARIO 1
PERIODO_REFRAT_ABS 1

```

diminuindo T_o para 0.1

```

DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 1
OFFSET_DE_ENTRADA 0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 0.1
LIMIAR_DE_DISPARO 2.0
PESO_REFRATARIO 1
PERIODO_REFRAT_ABS 1

```

mantendo T_o com o valor inicial e diminuindo T_i para 0.1

```

DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 1
OFFSET_DE_ENTRADA 0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 0.1
CONSTANTE_TEMPO_To 1.0
LIMIAR_DE_DISPARO 2.0
PESO_REFRATARIO 1
PERIODO_REFRAT_ABS 1

```

como a representação aparentemente mais rica é a To 0.1, passa-se a explorar o limiar de disparo

diminuindo o limiar de disparo para 1.0

```

DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 1
OFFSET_DE_ENTRADA 0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 0.1
LIMAR_DE_DISPARO 1.0
PESO_REFRATARIO 1
PERIODO_REFRAT_ABS 1

```

aumentando o limiar de disparo para 2.6

```
DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 1
OFFSET_DE_ENTRADA  0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 0.1
LIMIAR_DE_DISPARO 2.6
PESO_REFRATARIO   1
PERIODO_REFRAT_ABS 1
```

alterada a estratégia de entrada, usando agora 10 neurons por entrada, ponderados pelo arquivo pesos

estratégia de pesos lineares

1.0 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1

```
DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 10
OFFSET_DE_ENTRADA  0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 0.1
LIMIAR_DE_DISPARO 2.0
PESO_REFRATARIO   1
PERIODO_REFRAT_ABS 1
```

com isso espera-se melhor definição das transições

ainda com a estratégia de entrada, usando agora 10 neurons por entrada, ponderados pelo arquivo pesos

estratégia de pesos lineares, alterados para

1.5 1.4 1.3 1.2 1.1 1.0 0.9 0.8 0.7 0.6

```
DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 10
OFFSET_DE_ENTRADA  0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 0.1
LIMIAR_DE_DISPARO 2.0
PESO_REFRATARIO    1
PERIODO_REFRAT_ABS 1
```

ainda com a estratégia de entrada, usando agora 10 neurons por entrada, ponderados pelo arquivo pesos

estratégia de pesos lineares, alterados para

1.0 0.9 0.85 0.8 0.75 0.7 0.65 0.6 0.55 0.5

```
DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 10
OFFSET_DE_ENTRADA  0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 0.1
LIMIAR_DE_DISPARO  2.0
PESO_REFRATARIO    1
PERIODO_REFRAT_ABS  1
```

Teste 10

Teste considerando:

camada1 com configuracao1

```

DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 1
OFFSET_DE_ENTRADA  0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 1.0
LIMIAR_DE_DISPARO 2.6
PESO_REFRATARIO   1
PERIODO_REFRAT_ABS 1

```

camada2 com configuracao1 - com limiar de disparo 0.2 na pasta MLP1

```

N_NEURONS      100
DIM_ENTRADA     20
MAX_POTENCIAL_MEMB 40
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 1.0
LIMIAR_DE_DISPARO 0.2
PESO_REFRATARIO   1
PERIODO_REFRAT_ABS 1
N_EPOCAS        500
COEF_APRENDIZADO 0.01
N_INTERVALO_SALVAPESOS 0

```

resultado conjunto treino 80,5%

resultado conjunto teste 65%

Teste 11

Resultado avaliando tipo de representação.

Configuração camada 1

```

DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 3
OFFSET_DE_ENTRADA  0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 0.1
LIMAR_DE_DISPARO  2.0
PESO_REFRATARIO    1
PERIODO_REFRAT_ABS 1
  
```

Configuração camada 2

```

N_NEURONS        100
DIM_ENTRADA       60
MAX_POTENCIAL_MEMB 40
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 1.0
LIMAR_DE_DISPARO  0.4
PESO_REFRATARIO    1
PERIODO_REFRAT_ABS 1
N_EPOCAS          500
COEF_APRENDIZADO   0.01
N_INTERVALO_SALVAPESOS 10
  
```

Resultado para contagem de numero de pulsos por neurônio- 62,5%

Resultado para contagem de número de pulsos a cada iteração- 45%

Teste 12

Em teste12 são refeitos os testes para o melhor resultado obtido até o momento, que corresponde ao teste10 com configuração1 para camada1 e configuração1 para camada2.

A alteração está na rede da camada2, que passa a ter critério de parada. O critério é o delta energia da rede, se ficar abaixo de mínimo o treinamento para. É mantido o critério de parada por iterações (para que funcione, basta setar o critério de energia em zero).

Configurações camada 1

```
DIM_ENTRADA      20
N_NEURON_POR_ENTRADA 1
OFFSET_DE_ENTRADA 0
MAX_POTENCIAL_MEMB 12
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 1.0
LIMiar_DE_DISPARO 2.6
PESO_REFRATARIO   1
PERIODO_REFRAT_ABS 1
```

Configurações camada 2

```
N_NEURONS        100
DIM_ENTRADA      20
MAX_POTENCIAL_MEMB 40
CONSTANTE_TEMPO_Ti 10.0
CONSTANTE_TEMPO_To 1.0
LIMiar_DE_DISPARO 0.2
PESO_REFRATARIO   1
PERIODO_REFRAT_ABS 1
N_EPOCAS          500
CRITERIO_DE_PARADA 0.0
COEF_APRENDIZADO 0.01
N_INTERVALO_SALVAPESOS 0
```

Resultado variando o número de iterações da camada 2:

50 iterações	conjunto treino – 92%	conjunto teste – 63,5%
110 iterações	conjunto treino – 96,5%	conjunto teste – 59%
500 iterações	conjunto treino – 92,5%	conjunto teste – 68%

APÊNDICE VIII

Publicações

- 1 – TIMOSZCZUK, A.P. ; CABRAL JR., E.F. Speaker Recognition Using Pulse Coupled Neural Networks. Submetido à Speech Communication, Elsevier Science Publishers.

Speaker Recognition Using Pulse Coupled Neural Networks

Antonio Pedro Timoszczuk¹ and Euvaldo F. Cabral Jr.²

¹University of Sao Paulo – DEE/EPUSP/PTC – Laboratory of Communication and Signals – LCS – P.O. Box 8174, Sao Paulo, SP, 01065-970, Brazil.^(*)

²Noetics Institute Inc., 112 South Center St., Plainfield, IN, USA.

Abstract

Pulse Coupled Neural Network (PCNN) is a paradigm that has not yet been explored enough in speaker recognition. This paper presents the results of experiments conducted to develop a new recognition architecture that applies PCNN to text independent speaker recognition. The proposed architecture comprises a two layer PCNN for feature extraction and a Multilayer Perceptron (MLP) for final classification. The first layer of PCNN performs a pulse coding task and its outputs are used as inputs to a second self-organizing layer that learns the pulse's statistics. Initially, the well known Mel-Cepstral Frequency Coefficients were used as a benchmark to verify the new architecture's capability of learning the speaker information. After these preliminary tests, the Mel scaled filter bank energy coefficients were used, resulting in a biologically plausible architecture. The first results demonstrated the PCNN's ability to deal with temporal information contained in speech signals. The proposed architecture is promising, presenting 82% recognition when compared with 96% of the classical MLP classifier.

Author Keywords: speaker recognition; artificial neural networks; pulse coupled neural networks; temporal information

1. Introduction

Speaker identification is a particular category of the more general task of speaker recognition and aims to reveal the identity of a particular speaker based on the comparison of the incoming utterance with a number of patterns previously stored. This identification is performed using features extracted from each utterance, which are used afterwards as inputs to a classifier.

Artificial neural networks (ANN) are widely used as pattern classifiers [1] and many researchers have applied ANN in speaker recognition tasks [2]. The actual wave of interest in Pulse Coupled Neural Networks (PCNN) comes from biology studies where a number of researchers are working to develop models that mimic the natural neurons. Most works are concentrated in theoretical aspects of the neuron model and a few researchers report practical applications of such a paradigm.

^(*) Correspondence address: Av. Prof. Jose Maria Alkmin, 99 – ap113 – São Paulo – SP – Brazil – 05366-000 – Tel: +55-11-5576-8519 Fax: +55-11-5574-6667 E-mail: timoszczuk@uol.com.br

This paper reports the use of PCNN in developing a new architecture for speaker recognition systems. First a presentation of a pulsed neuron model is done and a two-layer architecture of pulsed neurons is proposed. Afterwards, text independent speaker recognition is performed in a closed set of 10 speakers in order to verify if the proposed architecture is suitable for the task.

2. Pulsed neuron model

In pulsed neuron models, the inputs and outputs are given by short pulse sequences generated over time. The model used in this work is the *Spike Response Model* described in Maass [3] and presented in Figure 1 with some modifications. Considering that the neuron in the figure is the i th neuron, the internal state is described by a variable u_i that represents the membrane potential in biological systems. When this internal variable exceeds a threshold, the neuron fires and generates a pulse on its output y_i . This internal variable depends on external inputs given by connections from other neurons x_j and an internal feedback from the neuron output that adjusts the neuron sensitivity to inputs in order to fire the neuron again.

Figure 1 – Block diagram for *Spike Response Model*, as modified from Maass [3].

It should be noted that when a pulse arrives at the input x_j , it is multiplied by the corresponding weight which represents the synaptic efficiency. Afterwards, the pulse

passes through a leaky integrator with time constant τ_{ji} , that mimics the biological fading of post-synaptic potential. All incoming pulses are added prior to being compared at the soma and if the weighted sum exceeds a threshold v_i , a pulse is generated at the output. This analysis was conducted supposing that the neuron is at its rest state and no previous input pulse had occurred, so p_i is null and there is no effect due to dynamic threshold. Immediately after the output pulse is generated, the feedback link receives a pulse and generates a dynamic threshold that is added to the static threshold and makes the membrane potential more negative. The dynamic threshold is then implemented using a leaky integrator and corresponds to biological ionic channels returning to the rest state. This negative potential makes the next firing of the neuron more difficult, and several accumulated input pulses are necessary in order to make the neuron fire again. If the input does not receive more pulses, the post-synaptic and membrane potential fades with time constants τ_{ji} and τ_i respectively, conducting the neuron to the rest state.

In order to run this neuron model using digital computers, a discrete model must be used. This discrete model is obtained dividing the continuous time in intervals of constant duration T , which are used to compute each neuron state over the discrete time space. The leaky integrators were replaced by a first order digital filter with relaxation factor τ , corresponding to the fading factors.

The Equations 1 and 2 model the digital filter for post-synaptic fading:

$$p_{ji}(n) = w_{ji}x_j(n) + r_{ji}p_{ji}(n-1) \quad (1)$$

$$r_{ji} = \exp\left(-\frac{T}{\tau_{ji}}\right) \quad (2)$$

Similarly the Equations 3 and 4 model the digital filter for membrane potential fading:

$$p_i(n) = \eta_o y_i(n) + r_i p_i(n-1) \quad (3)$$

$$r_i = \exp\left(-\frac{T}{\tau_i}\right) \quad (4)$$

u_i corresponding to internal membrane potential for i th neuron

x_j Is input coming from neuron j to neuron i

w_{ji} represents the connection efficiency from neuron j to neuron i

y_i represents the pulse at neuron output

r_{ji} and r_i are given by Equation 2 and 4, with constant decay τ_{ji} and τ_i respectively

p_{ji} corresponds to digital filter output for post-synaptic potential fading

p_i corresponds to digital filter output for membrane potential fading

Finally, considering the digital filters presented above, Equation 5 describes the neuron model in discrete space that was used to implement the pulsed layers in this work. It models the membrane potential resulting from inputs and dynamic threshold, with r_{ji} and r_i given by Equation 2 and 4 respectively.

$$u_i(n) = -\{\eta_0 y_i(n) + r_i p_i(n-1) + v_i\} + \sum_{j \in \Gamma_i} \{w_{ji} x_j(n) + r_{ji} p_{ji}(n-1)\} \quad (5)$$

Where Γ_i is the space containing all pre-synaptic neurons connected to neuron i , and thus the Equation 6 represents the sum of all inputs from $j \in \Gamma_i$ neurons connected to neuron i :

$$\sum_{j \in \Gamma_i} \{w_{ji} x_j(n) + r_{ji} p_{ji}(n-1)\} \quad (6)$$

and the expression:

$$-\{\eta_0 y_i(n) + r_i p_i(n-1) + v_i\} \quad (7)$$

represents the contribution of dynamic threshold, that is related to recent fire states of neuron i and is indicated as ω_i in Figure 1.

3. Layered architecture proposed

The neural network architecture proposed for feature extraction consists of two layers of pulsed neurons as illustrated in Figure 2. The first layer converts the input sequence into a pulse-modulated sequence and the second one extracts and generates features to a MLP classifying layer.

Figure 2 – The block diagram of proposed architecture for speaker recognizer.

pulse coding layer

In this layer, the number of neurons is equal to the dimension of input vector and there are no feedback connections between the neurons.

The input sequence of Mel Cepstral Coefficients (MFCC) is presented to this layer, and each MFCC coefficient is fed to one of the input neurons. The neuron number 1 receives all first MFCC coefficient sequences, the second neuron the second coefficient and so on. As the sequence is presented to neuron inputs, the membrane potential varies proportionally with the amplitude of MFCC. The result is that a frequency modulated pulse sequence is obtained proportional to the amplitude of the MFCC coefficients.

feature extraction layer

This layer presents a novelty in this architecture. It is composed of a linear arrangement of neurons interconnected in such a way that forms a ring. This network follows the training principles of well-known self-organizing maps and is trained using competitive rules. The ring configuration works better than a simple linear arrangement and ensures that there is no border distortion introduced during the training.

This ring self-organized map has its training algorithm based on self-organization and makes use of neighbor updating rules. During the training, the winning neuron weights are updated strongly and the neighbors' weights are updated with a strength that is inversely proportional to the distance from the winner. The neighborhood is wide at the start of training and is decreased until only the winning weights are updated.

At the end of the training process, the weights assume the most common combination of values presented by the input sequence and are used to conduct a mapping of the input to the classifier layer.

classifying layer

The multilayer perceptron was used in this work as a reference classifier. It was implemented in a "classic configuration" [4] with three layers fully connected: the first layer, one hidden layer, and the last layer. The first layer always has the dimension of the output vector from feature extraction layer and the number of outputs (last layer) was made equal to the number of classes (speakers) to identify. In the hidden layer, the number of neurons was fixed at 300. In the first layer, the neuron function used was linear, and these units were used as coupling for the input vector. In the hidden and last layers, the neuron function was chosen to be an adjustable sigmoid as described in Equation 8 which made it possible adjust the neuron response to the dynamic range of input data.

$$y = \frac{a}{1 + e^{-c \cdot x}} - b \quad (8)$$

y = neuron output

x = neuron input

a, b, c = coefficients to adjust the neuron
function

The MLP training was done using a standard back propagation algorithm [4] with adaptive learning rate and a fixed momentum.

4. Speech corpus and feature extraction

speech corpus

The speech database was constructed using a subset of the Speaker Recognition v1.0 database from CSLU – Center for Spoken Language Understanding – of Oregon Graduate Institute - U.S.A. The subset selected was composed 10 of speakers (5 males and 5 females) and 8 phrases recorded over a digital phone line were used with 8KHz sampling rate and 16 bits. The phrases were uttered 5 times by the speakers in different recording sessions over time, resulting in 40 utterances for each speaker.

feature extraction

The utterances were segmented into 256-point frames resulting in 32ms frames of speech. Consecutive frames were overlapped 128 points (16ms) and 16 Mel Cepstral coefficients were calculated for each frame with 20 filters uniformly spaced in the mel scale, using a discrete cosine transform (DCT).

5. Experimental results

The phrases were split into two sets, the first one with phrases 1-2-3-4 for training and the second with phrases 5-6-7-8 for testing.

The first layer of the proposed architecture was configured with 16 pulsed neurons each one to process a corresponding MFCC coefficient. The second layer was configured with 100 pulsed neurons interconnected as described in session 3.

In order to train the classifier, it was necessary to define the feature vector to be used. In these tests, the number of output pulses for each neuron was used as a feature set, resulting in a one dimensional icon.

test 1

In test 1, the training set was presented to the recognizer and layer two was trained over 500 epochs. Afterwards, the training set was presented again and layer two generated the corresponding output pulse sequences. These pulse sequences were then transformed into icons as described previously and used to train the classifier. After the trainings were completed, the testing sequences were presented to the recognizer resulting in 96% recognizing for training set and 82% for testing.

test 2

Test 2 was conducted in order to verify the reason for such different recognition rates from test 1 and to verify if the pulsed layers were extracting the speaker information. Each phrase used in this experiment have a phonetic contents that characterizes the speaker. The best characterization is done when all phrases with complementary phonetic contents are used in a training set, providing a more complete model for each speaker. After these considerations, the phrases used in training and test sets were selected to provide variable information in the training phase and use only one phrase to test if the recognition rate changes. It is expected that as the train set

contains more information the recognition rate increases. The phrases used to train and test the recognizer as well as the results are presented in Table 1.

Training set A: phrases 01-02-03-05-06-07

Training set B: phrases 02-03-04-06-07-08

Training set C: phrases 01-02-03-04-06-07-08

Training set	test phrase	recognition rate
A	4	98%
A	8	76%
B	1	98%
B	5	76%
C	5	86%

Table 1 – Results for test 2, increasing information on training set.

6. Conclusion

This work aimed to use a pulse coupled neural network in a text independent speaker identification task. The approach using a new pulsed neuron layered architecture to extract speaker features presented good results and the use of PCNN for speaker recognition task is promising.

It was noted that the low recognition rate was due to a poor training set as demonstrated in test 2. The recognition rate is expected to increase with a large training set.

In test 2, when the phrase 1 was returned to the training set and the recognition rate increased, it was clear that the pulsed neural network was obtaining this additional information and giving it to the classifier demonstrating that the architecture proposed preserved the information.

Finally, the main conclusions here are:

- a) the PCNN are promising for speaker recognition tasks;
- b) the proposed architecture was able to generate features that conduct to speaker recognition.

Future work will explore other strategies to choose feature vectors from PCNN and implement a classifying layer using pulsed neurons.

7. References

- [1] Lippmann, R.P. Pattern classification using neural networks. IEEE Communications Magazine, v.27, n.11, p.47-64, Nov. 1989.
- [2] Farrel, K.R. , Mammone, R.J. Speaker recognition using neural networks and conventional classifiers. IEEE TSAP v.2, n.1, Part II, p.194-205, Jan. 1994.
- [3] Mass, W. and Bishop, C., editors, Pulsed Neural Networks, MIT-Press, Cambridge, 1999.
- [4] Haykin, S. Neural networks: a comprehensive foundation. New York, MacMillan, 1994.