SISTEMA DE DETECÇÃO DE FALHAS DE MANOBRAS EM SECCIONADORES DE ALTA TENSÃO BASEADO EM PROCESSAMENTO DIGITAL DE SINAIS E RNA

André Colen Carrasco

Orientador: Prof. Germano Lambert Torres

Co-Orientador: Prof. Luiz Eduardo Borges da Silva

Dissertação apresentada à Universidade Federal de Itajubá, para obtenção do título de Mestre em Engenharia Elétrica.

Março / 2005

“Dedico este trabalho a meus grandes ídolos, meus pais Álvaro e Edith.”

AGRADECIMENTOS

Ao orientador Professor Germano Lambert Torres pelo apoio, confiança e pela sua dedicada orientação.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo apoio financeiro.

À Universidade Federal de Itajubá – UNIFEI.

À Cynthia, pelos inúmeros momentos de apoio, compreensão, pelas alegrias e companheirismo durante todo o tempo.

À meus grandes irmãos e amigos Rodrigo, Thiago, Divino, Rogério, Ronaldo, Ricardo e Paulinho pela ajuda e amizade nos momentos difíceis.

À tia Elvira e prima Kátia pela revisão ortográfica.

A todos aqui não mencionados que tiveram participação direta ou indireta na confecção desta dissertação.

Muito obrigado! O carinho que sinto por todos vocês é imensurável!

SUMÁRIO

Lista de Tabelas

Lista de Figuras

Lista de Símbolos

Capítulo 2

$\underline{x_i}$ – *j*-ésimo vetor de padrão.

$\underline{X}$ – conjunto dos *m* vetores de padrão.

$\underline{y_j}$ – *j*-ésimo vetor de característica.

$\underline{Y}$ – conjunto dos *n* vetores de característica.

M – quantidade de classes distintas.

^{T} – transposta.

z – protótipo (saída desejada do sistema classificador).

z_j^k – *k*-ésimo protótipo da classe j.

w_k – *k*-ésima classe do sistema classificador.

$d_i(x)$ – *i*-ésima função discriminante.

Capítulo 3

w – peso sináptico.

xi – *i*-ésima entrada da rede.

yi(n) – valor de saída do *i*-ésimo neurônio para o *n*-ésimo padrão.

di(n) – valor de saída desejado no *i*-ésimo neurônio para o *n*-ésimo padrão.

ei(n) – valor de erro no *i*-ésimo neurônio para o *n*-ésimo padrão.

?(n) – energia instantânea do erro.

?med – energia média do erro, ou função de custo.

N – número de total de padrões no conjunto de treinamento.

?j(n) – valor de propagação do *j*-ésimo neurônio para o *n*-ésimo padrão.

f j – função de ativação do *j*-ésimo neurônio.

RESUMO

Esta dissertação tem por objetivo desenvolver um sistema capaz de detectar falhas incipientes e verificar a correta operação de abertura e fechamento de seccionadores de alta tensão pela análise digital da curva do torque fornecida pelo motor de acionamento durante as manobras do seccionador, superando assim as dificuldades envolvidas com o monitoramento direto dos contatos da parte ativa. Este sistema será baseado na utilização de dispositivos dedicados ao Processamento Digital de Sinais (DSP's) associado a algoritmos de Redes Neurais, para identificação de padrões de falhas recorrentes durante as manobras dos seccionadores. Os sinais a serem processados pelo DSP serão os valores do torque do motor de acionamento, calculado pelos valores instantâneos da potência elétrica fornecida. A análise, em tempo real, da curva do torque fornecida pelo motor durante as operações de abertura e fechamento possibilitará um monitoramento indireto da posição dos contatos principais do seccionador, aumentando a confiabilidade de operação do sistema elétrico. As informações fornecidas pelo sistema de detecção de falhas permitirão uma manutenção preditiva mais eficiente dos seccionadores, baseada nas indicações de desajustes ou quebras de componentes no sistema mecânico de acionamento e parte ativa da chave.

ABSTRACT

The purpose of this dissertation is to present the development of a system capable of detecting incipient malfunction and also verify the correct operation of opening and closing of high voltage disconnectors thru digital analysis of the torque curve supplied by the drive engine during the maneuvers of the disconnector, surpassing the difficulties involved with the direct monitoring of the contacts. This system will be based on the use of dedicated devices to the Digital Signals Processing (DSP's) associated to the algorithms of Artificial Neural Networks, for pattern recognitions of recurrent malfunction during the maneuvers of the high voltage disconnectors. The signals to be processed by the DSP will be the values of the torque of the drive engine, calculated by the immediate values of the electric power supplied. The real-time analysis of the curve of the torque supplied by the engine during the operations of opening and closing will make possible for an indirect monitoring of the position of the main contacts of the high power switch, increasing the trustworthiness of the electrical system’s operation. The information supplied by the detection system of malfunction will allow a more efficient preventive maintenance of the disconnectors, based on the indication of misaligned or broken components in the mechanical system of power switch.

Capítulo 1

INTRODUÇÃO E OBJETIVOS

1.1 O PROBLEMA

Chaves seccionadoras são extremamente importantes e largamente utilizadas nas redes de distribuição de energia. Elas têm o papel de energizar ou não barramentos, ramos, linhas de transmissão, ou subestações. Para a manutenção deste equipamento, muitas vezes se faz necessário o desligamento do ramo, interrompendo assim o fornecimento de energia.

A correta manutenção das chaves seccionadoras e o supervisionamento técnico constante se fazem necessário para evitar tal transtorno e possíveis prejuízos.

São conhecidos diversos defeitos em seccionadores de alta tensão, desde falta de alimentação no cubículo de acionamento, falta de lubrificação nas junções, até fechamento incorreto das fases.

Este trabalho visa elaborar um instrumento computacional capaz de determinar a distância o comportamento de uma chave seccionadora, assim como supervisionar o seu funcionamento e detectar possíveis defeitos. Este instrumento deverá fazer uso de técnicas de processamento digital de sinais e de reconhecimento de padrões para classificar o funcionamento da chave seccionadora.

1.2 A CHAVE SECCIONADORA

A chave seccionadora em estudo tem as seguintes características:

- Fabricante: Bowthorpe Power Equipment – Banbury – England
- Modelo: R500
- Tensão Nominal: 345 kV / 60 Hz
- Corrente Nominal: 1250 A

Figura 1.1 – A chave seccionadora.

Figura 1.2 – Destaque das barras paralelas responsáveis pelo fechamento/abertura.

Figura 1.3 – Destaque dos contatos.

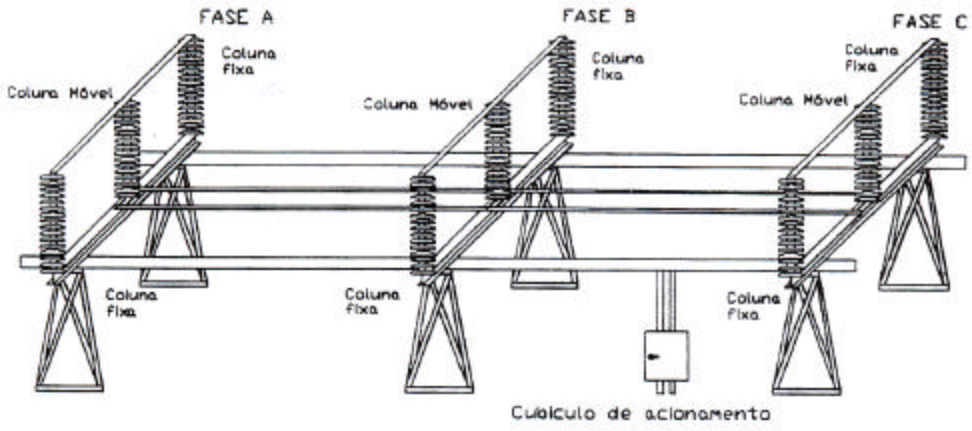


Figura 1.4 – Visão geral da chave seccionadora.

1.3 IDÉIA DE SOLUÇÃO

Para resolver a proposta deste trabalho, utiliza-se, como referência do funcionamento da chave seccionadora, a curva de torque obtido na abertura e no fechamento do equipamento, que fora calculado através dos valores instantâneos da potência elétrica fornecida ao motor de acionamento do seccionador.

Adquirindo diversas curvas para diferentes defeitos e separando-as por tipo, utilizam-se técnicas de reconhecimento de padrões para classificá-las. Técnicas de processamento digital de sinais também serão empregadas para eliminar ruídos e fornecer bancos de dados confiáveis.

Após o treinamento e testes de uma rede neural artificial que satisfaça o objetivo deste trabalho, será desenvolvido um programa computacional capaz de realizar o pré-processamento dos sinais e simular a saída da rede a partir dos pesos obtidos no treinamento.

A figura abaixo exemplifica o processo:

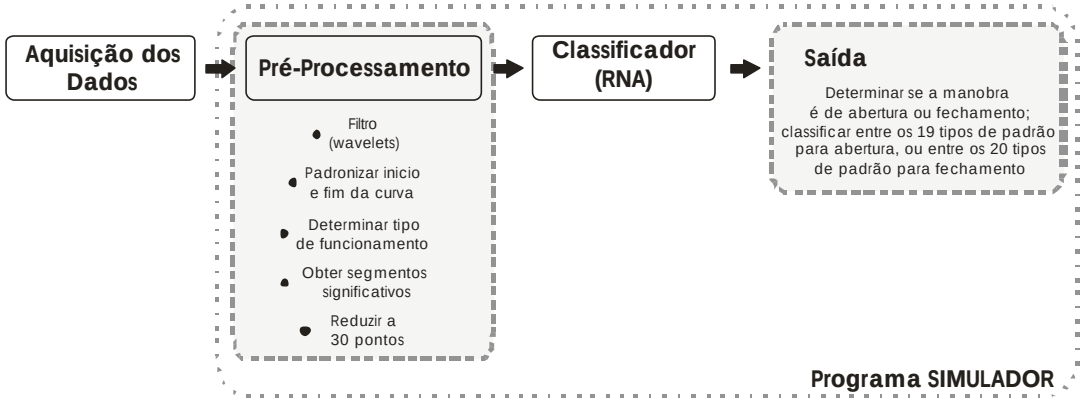


Figura 1.5 – Visão geral das etapas/fases a serem cumpridas.

1.4 ESTRUTURA DA DISSERTAÇÃO

O próximo capítulo desta dissertação contém uma abordagem teórica de reconhecimento dos padrões; serão discutidos temas como superfícies de decisão e redes neurais artificiais atuando como funções discriminantes. A teoria sobre as Redes Neurais Artificiais (RNA's) encontra-se no capítulo 3. No capítulo 4, um enfoque detalhado sobre o funcionamento da técnica de Wavelets é apresentado; comparações com algumas derivações da análise de Fourier são feitas, justificando a escolha pela análise de Wavelets. O capítulo 5 apresenta o desenvolvimento de todo o sistema aqui abordado: inicia-se o capítulo com um estudo preliminar do banco de dados utilizado, a seguir são apresentadas todas as etapas realizadas no pré-processamento, e o desenvolvimento do classificador adotado. Os testes e resultados das redes encontram-se no capítulo 6. No capítulo 7, é apresentado o desenvolvimento programa SIMULADOR, projetado para automatizar todos os cálculos contidos no pré-processamento e no classificador, facilitando a obtenção dos resultados pelo usuário. Conclusões e desenvolvimentos futuros encontram-se no capítulo 8 que precede os anexos deste trabalho e a lista das bibliografias e referências utilizadas.

Capítulo 2

EMBASAMENTO TEÓRICO DE RECONHECIMENTO DOS PADRÕES

2.1 PADRÕES E RECONHECIMENTO DOS PADRÕES

O conceito de *padrão* é muito conhecido e utilizado há milhares de anos. Desde a idade pré-histórica, o homem utiliza padrões para se comunicar através de figuras impressas em rochas e outros objetos. Com o passar dos anos, o conceito de padrão foi sendo desenvolvido e surgiram as línguas, as ferramentas e a tecnologia.

Em Inteligência Artificial, padrões de sinais podem ser utilizados para identificação pessoal por voz, caligrafia, impressões digitais, imagens faciais e também podem ser utilizados para reconhecer a fala, caracteres, objetos em imagens, entre outras. Incluem-se, ainda, identificação de alvos militares baseados em radares, infravermelho e/ou imagens de vídeo.

Há possibilidade de uso de padrões em diversas áreas do conhecimento humano, como por exemplo, em geologia, meteorologia, personalidade, cultura, história, e outros dados desde imagens microscópicas de células até imagens macroscópicas de regiões, como as imagens obtidas por satélite da terra e de outros planetas e galáxias.

Humanos são capazes de reconhecer, facilmente, rostos de amigos quando crianças, caracteres e palavras em uma página impressa, vozes, músicas, sons, distinguir frutas, objetos, texturas, entre outros.

Quando se consegue distinguir um objeto de uma população **P** e agrupá-la numa subpopulação **S**, pode-se dizer que o reconhecimento do padrão foi feito. O reconhecimento de um objeto individual como uma única classe é chamado de identificação.

Classificação é o processo de agrupar objetos em classes (subpopulações) de acordo com suas características ou similaridades.

Reconhecimento dos padrões envolve tanto identificação quanto classificação para a construção de uma máquina inteligente – esse é o processo de fazer máquinas aprenderem e tomarem decisões, como os humanos [Bow, 84].

Humanos aprendem a partir de experiências, acumulando regras em várias formas de associação, tabelas, inequações, implicações lógicas, entre outros.

Classificação é uma forma de aprendizado que indica, a partir de atributos prévios, as classes conseqüentes. Por outro lado, argumentação é o processo de aplicar regras, equações e relações, através de uma coleção inicial de dados, para deduzir o resultado ou decisão.

Para se iniciarem o reconhecimento e a classificação de padrões se faz necessário um banco de dados inicial do tipo causa-efeito.

O conceito de classificação envolve o aprendizado de diferentes tipos de padrões e suas conclusões em uma população. A associação entre padrões e suas conclusões formam a base da inteligência artificial aplicadas em reconhecimento dos padrões [Bow, 84].

Pode-se dividir o reconhecimento de padrões em três fases:

- Aquisição de dados;
- Pré-processamento dos sinais;
- Classificador.

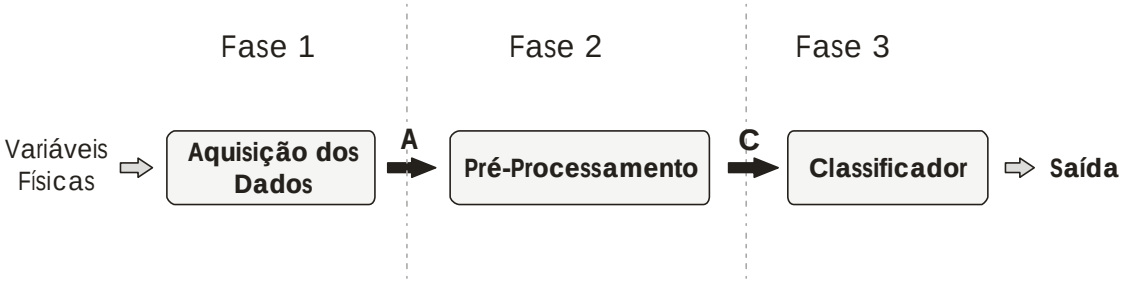


Figura 2.1 – Fases necessárias para reconhecimento de padrões

Na fase de aquisição de dados, dados analógicos do mundo físico são coletados através de conversores ou transdutores e transformados em formato digital para processamento computacional. Neste estágio, as variáveis físicas são convertidas em dados mensuráveis, indicados na figura 2.1 pelo ponto **A**. Se as variáveis físicas forem sons, o transdutor poderá ser um microfone, se a variável física for intensidade de luz, o transdutor poderá ser uma fotocélula. Dessa forma, os dados mensuráveis poderão ser utilizados na entrada da segunda fase (pré-processamento dos sinais), e são chamados de *vetores de padrão ($\underline{x}_i$)*.

Um único vetor de padrão é representado da seguinte forma:

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_r \end{bmatrix} \qquad \text{Equação 2.1}$$

Onde *r* representa a dimensão do vetor de padrão, ou seja, a quantidade de dados mensuráveis que descrevem as variáveis físicas adquiridas. Se *r* = 3, o espaço de padrões poderá ser representado graficamente.

Da mesma forma, o vetor $\underline{X}$, que é formado pelo conjunto de *m* vetores de padrão, é descrito por:

$$\underline{X} = \begin{bmatrix} \underline{x}_1^{\,T} \\ \underline{x}_2^{\,T} \\ \underline{x}_3^{\,T} \\ \vdots \\ \underline{x}_m^{\,T} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1r} \\ x_{21} & x_{22} & \cdots & x_{2r} \\ x_{31} & x_{32} & \cdots & x_{3r} \\ \vdots & & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mr} \end{bmatrix} \qquad \text{Equação 2.2}$$

Onde *T* denota transposta, *m* representa a quantidade de vetores de padrão, e *r* representa a dimensão do vetor de padrão.

Após essa etapa, o pré-processamento será usado para converter dados discretos em dados matemáticos mais apropriados à análise computacional. A saída dessa fase é indicada na figura 2.1 pela letra **C**.

O objetivo dessa fase é gerar novos dados relacionados aos dados iniciais, podendo eliminar dados insatisfatórios e/ou redundantes, para formar o *vetor de características (*$\underline{y}$*)*, que contém apenas os dados necessários à classificação.

Para o nosso caso, como o vetor de padrão $\underline{x}$ contém *r* dados temporais de uma única variável (torque da chave em função do tempo), o vetor de características $\underline{y}$ terá dimensão menor do que o vetor de padrão $\underline{x}$, pois os dados redundantes e insatisfatórios serão eliminados de forma a facilitar o trabalho do classificador.

Um único vetor de características é representado da seguinte forma:

$$\underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \qquad \text{Equação 2.3}$$

Onde $n < r$, ou seja, a dimensão do vetor de características é menor do que a dimensão do vetor de padrão.

O conjunto dos vetores de características é descrito por:

$$\underline{Y} = \begin{bmatrix} \underline{y}_1^{\,T} \\ \underline{y}_2^{\,T} \\ \underline{y}_3^{\,T} \\ \vdots \\ \underline{y}_m^{\,T} \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ y_{31} & y_{32} & \cdots & y_{3n} \\ \vdots & & & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{bmatrix} \qquad \text{Equação 2.4}$$

Onde T denota transposta, m representa a quantidade de vetores de características, e n representa a dimensão do vetor de características.

O classificador, que pertence a fase 3 descrita na figura 2.1, opera sobre o vetor de características $\underline{Y}$ de acordo com as funções de decisão.

Os vetores de padrão serão “pontos” no espaço de padrões, e padrões pertencentes à mesma classe estarão próximos e no mesmo *cluster*. Cada cluster representará uma classe distinta, ou seja, diferentes clusters representarão diferentes classes no espaço de padrões.

O classificador, composto de suas funções de decisão, irá definir a classe que cada padrão particular se encontra.

A saída do classificador poderá ser ilustrada pelo espaço de classificação. Este será M-dimensional se o classificador for utilizado para classificar M classes distintas. Para um problema de classificação simples contendo duas classes, M = 2; para classificar o alfabeto, M = 26.

Tanto o pré-processamento quanto o classificador são escolhidos pelo projetista. Os coeficientes (ou pesos) usados nas funções de decisão do classificador são calculados sobre os dados conhecidos a priori durante a fase de treinamento do classificador. Durante a fase de treinamento, os coeficientes são ajustados a partir do

resultado de cada classificação estar correto ou não. Essa etapa de adaptação dos coeficientes é obtida de forma *off-line*, durante a fase de treinamento. A fase de pré-processamento é de fundamental importância para uma classificação mais eficiente.

Conforme discutido anteriormente, o conhecimento a priori da classificação de alguns dados é necessário na fase de treinamento do classificador. Esses dados são chamados de protótipos ($\underline{z}$) e são descritos da seguinte forma:

$$\underline{z}_k = \begin{bmatrix} z_k^{\,1} \\ z_k^{\,2} \\ \vdots \\ z_k^{\,i} \\ \vdots \\ z_k^{\,Nk} \end{bmatrix} \qquad \text{Equação 2.5}$$

Onde, $k = 1, 2, \ldots, M$ indica a classe de padrão correspondente, e $i = 1, 2, \ldots, N_k$ indica o número do protótipo na classe w_k. Ou seja, M representa o número total de classes, e N_k o número total de protótipos na classe w_k.

Protótipos da mesma classe devem ter as mesmas propriedades e devem pertencer ao mesmo *cluster* no espaço de padrões. A figura 2.2 demonstra isso: os protótipos $z_1^{\,1}, z_1^{\,2}, \ldots, z_1^{\,N1}$ pertencem ao mesmo cluster w_1; os protótipos da outra classe, $z_2^{\,1}, z_2^{\,2}, \ldots, z_2^{\,N2}$, estão em outro *cluster* (w_2) e em outra região no espaço de padrões.

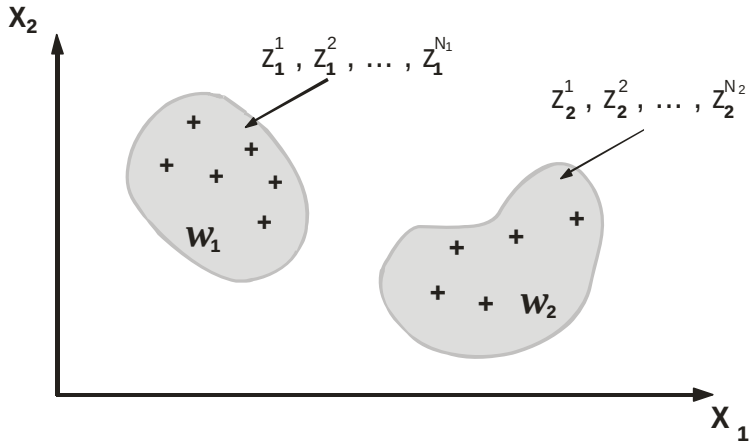


Figura 2.2 - Espaço de padrões bidimensional com duas classes.

O problema da classificação de padrões é achar a correta superfície de separação entre os clusters, capaz de tornar válido os protótipos *a priori* conhecidos nas corretas

classes. Esta região de separação também deverá classificar outras partições não utilizadas a partir dos mesmos critérios usados pelo classificador.

Padrões que não pertencerem a nenhuma classe, que por sua vez estarão em lugares diferentes no espaço de padrões, pertencem a classes não conhecidas ou não possuem classe. As distâncias entre as partições podem ser utilizadas para medir a similaridade entre as partições no espaço n-dimensional.

Algumas propriedades entre distâncias métricas podem ser utilizadas. São elas:

$$\begin{aligned} d(x,y) &= d(y,x) \\ d(x,y) &\leq d(y,z) + d(z,y) \\ d(x,y) &\geq 0 \\ d(x,y) &= 0, \; se \;\; y = x \end{aligned}$$

Equações 2.6

2.2 SUPERFÍCIE DE DECISÃO E FUNÇÕES DISCRIMINANTES

Como mencionado anteriormente, os padrões aparecem como "pontos" no espaço de padrões n-dimensional. Padrões localizados numa mesma região do espaço de padrões formam uma classe. Da mesma forma, classes diferentes aparecerão em diferentes regiões no espaço de padrões e podem ser facilmente separados utilizando superfícies de separação (n-1)-dimensional. Estas superfícies de separação são chamadas de superfícies de decisão, ou hiperplano de decisão.

Assim, obtidos os hiperplanos de separação a partir de padrões e classes de padrões conhecidos, é possível identificar novos padrões no espaço de padrões e classificá-los, ou seja, em um estudo preliminar com padrões e classes definidas, determinam-se os hiperplanos de decisão e, a partir deste, classificam-se novos padrões sem o prévio conhecimento. Desta forma, pode-se dizer que hiperplanos de decisão ruins acarretarão em classificações errôneas e equivocadas.

Quando n = 1, ou seja, quando o espaço de padrões for uma reta, a superfície de decisão é um ponto.

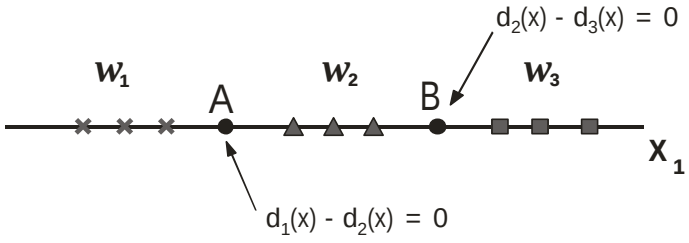


Figura 2.3 – Espaço de padrão unidimensional.

Onde **A** é um ponto que separa as classes w_1 e w_2, e o ponto **B** separa as classes w_2 e w_3.

Quando n = 2, a superfície de decisão é uma reta, descrita por:

$$w_1 x_1 + w_2 x_2 + w_3 = 0 \qquad \text{Equação 2.7.}$$

Onde w_1, w_2, e w_3 são coeficientes reais.

A figura 2.4 demonstra um espaço de padrão bidimensional separado por uma única superfície de decisão.

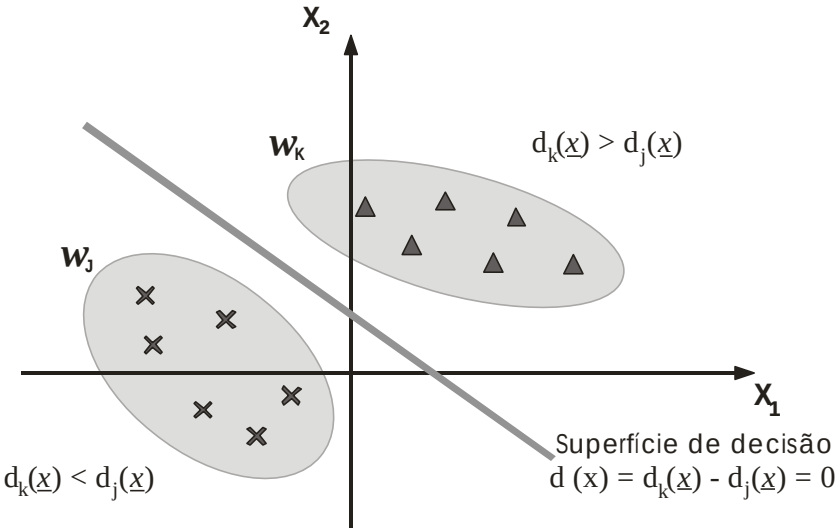


Figura 2.4 – Espaço de padrão bidimensional contendo uma única superfície de decisão.

Quando n = 3, a superfície de decisão é um plano.

Quando n = 4, a superfície será um hiperplano representado por:

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n + w_{n+1} = 0 \qquad \text{Equação 2.8.}$$

que pode ser expresso matricialmente por:

$$w^T . x = 0 \qquad \text{Equação 2.9.}$$

Onde:

$$\underline{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ w_{n+1} \end{bmatrix} \quad \text{e} \quad \underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix}$$

$\underline{w}$ e $\underline{x}$ são chamados de vetor de pesos aumentados, e vetor de padrão aumentada, respectivamente. No vetor $\underline{x}$ foi adicionado o termo $x_{n+1}=1$ para tornar a multiplicação válida.

A função que define a superfície de decisão é chamada de função discriminante $d(\underline{x})$. Na figura 2.4, $d_k(\underline{x})$ e $d_j(\underline{x})$ são valores da função discriminante para $\underline{x}$, nas classes *k* e *j* respectivamente.

Se $d(x) = d_k(x) - d_j(x) = 0$, define a superfície de separação das classes *k* e *j*.

Pode-se dizer que:

$$d_k(\underline{x}) > d_j(\underline{x}) \quad \forall \quad \underline{x} \in w_k$$
$$\forall \; j \neq k, \;\; j = 1, 2, \cdots, M$$

Equação 2.10.

Um sistema de classificação pode ser construído como mostrado na figura 2.5:

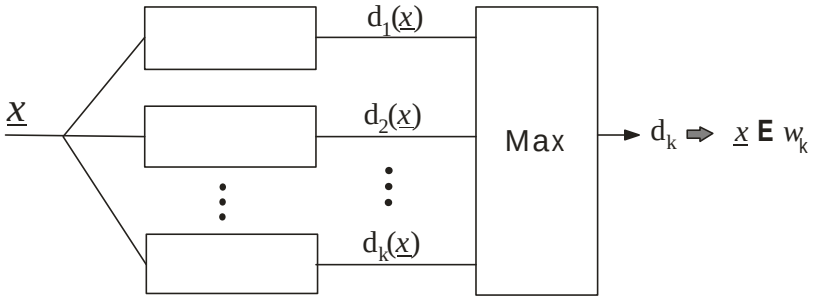


Figura 2.5 – Diagrama sistemático de um sistema classificador simples.

Para um sistema simples de classificação, contendo apenas duas classes, tem-se:

$$d(x) = d_1(x) - d_2(x) = 0$$

ou

$$d_1(x) = d_2(x)$$

que define a superfície de separação entre as duas classes.

De acordo com [Bow, 84] - para M classes diferentes, tem-se M* (M-1) /2 superfícies de separação, mas algumas das superfícies de separação são redundantes: apenas M-1 são necessárias para separar M classes. A figura 2.6 demonstra o número de superfícies de separação como função do número de classes no espaço de padrões bidimensional.

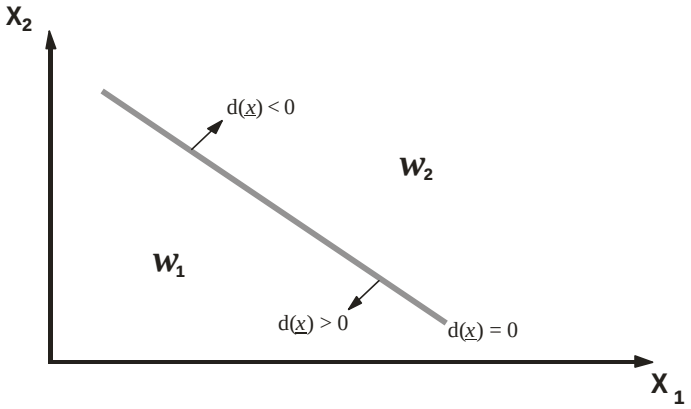


Figura 2.6.a – Espaço de Padrões subdividido por uma superfície de separação.

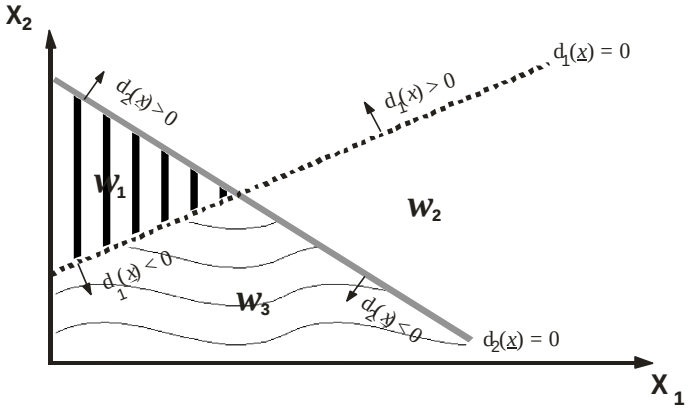


Figura 2.6.b – Espaço de Padrões subdividido por duas superfícies de separação.

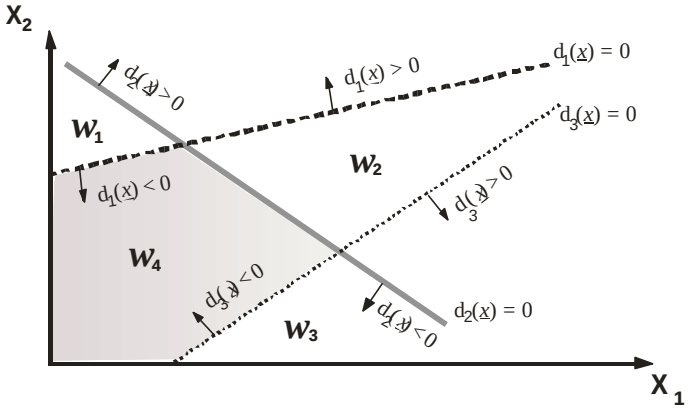


Figura 2.6.c – Espaço de Padrões subdividido por três superfícies de separação.

Na figura 2.6.a, a superfície de separação de duas classes é uma linha; sob a linha $d(\underline{x}) = 0$; abaixo dela $d(\underline{x}) > 0$; e acima dela $d(\underline{x}) < 0$.

Similar a esta, a figura 2.6.b é auto-explicativa. Note que a região indicada por $d_1(\underline{x}) < 0$ e $d_2(\underline{x}) < 0$ pertence à região w_3.

O mesmo ocorre na figura 2.6.c: padrões encontrados nas regiões formadas pelo cruzamento dos planos (seis regiões) pertencem a diferentes classes. Regiões não mencionadas no espaço de padrões são regiões indeterminadas.

2.3 REDES NEURAIS COMO FUNÇÕES DISCRIMINANTES

Na seção anterior foi demonstrado que, no espaço de padrões n-dimensional, o hiperplano de decisão, que divide o espaço em duas partes, descrito pela função discriminante é da forma:

$$d(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n + w_{n+1} = 0$$

E o vetor de características $\underline{x}^T = [x_1, \cdots, x_n]$ satisfaz uma das alternativas abaixo:

$$d(x) > 0 \qquad d(x) = 0 \qquad d(x) < 0$$

E poderá ser classificada no espaço de características.

A figura 2.7 ilustra o processo de decisão de uma rede neural:

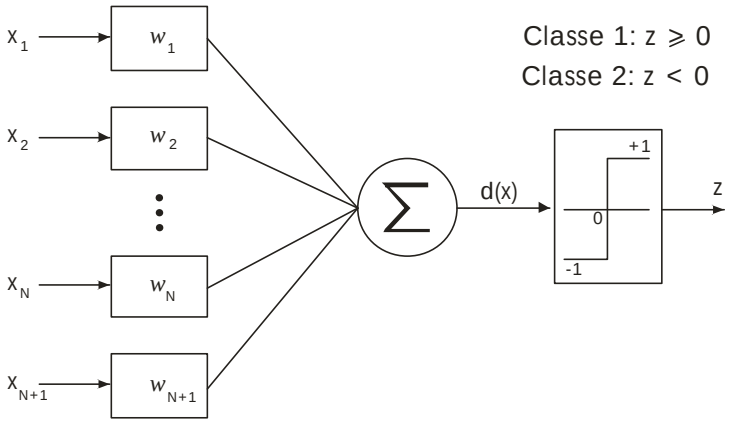


Figura 2.7 – Rede Neural como função discriminante linear.

As entradas x_i são linearmente combinadas com os pesos w_i, e o somatório resultante determina o valor de d($\underline{x}$). Por sua vez, a função de ativação irá determinar o valor da saída da rede. Esta será +1, se d($\underline{x}$) = 0, caso contrário, a saída da rede indicará -1. Assim, pode-se dizer que o identificador da classificação foi a saída *z* da rede.

A figura 2.8 ilustra uma rede multilinear com dois nós somatórios, onde a saída de cada nó indicará o seu valor discriminante linear $d_i(\underline{x})$.

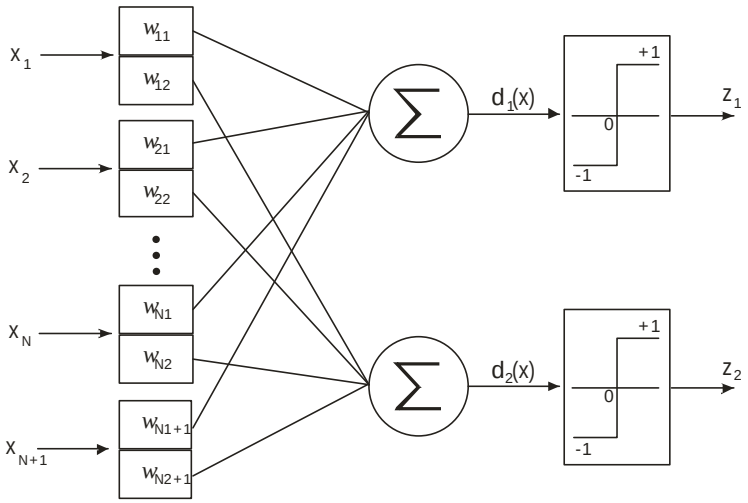


Figura 2.8 – Rede Neural como função discriminante multilinear.

A saída de cada função de ativação, z_1 e z_2, formam as saídas da rede. As combinações possíveis para as saídas dessa rede são:

$z_1 = 0,\ z_2 = 0;$

$z_1 = 0,\ z_2 = 1;$

$z_1 = 1,\ z_2 = 0;$

$z_1 = 1,\ z_2 = 1.$

As quatro combinações possíveis referem-se às quatro possíveis classes, distintas no espaço de características. A figura 2.9 ilustra as classes no espaço de padrões.

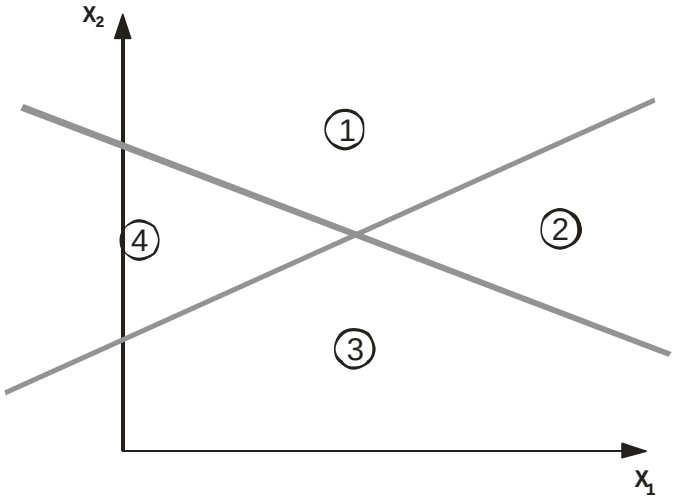


Figura 2.9 – Intersecção de dois hiperplanos de decisão formando quatro regiões distintas.

Capítulo 3

EMBASAMENTO TEÓRICO DE REDES NEURAIS ARTIFICIAIS

3.1 INTRODUÇÃO

A tecnologia das RNA's visa solucionar problemas de reconhecimento dos padrões que, geralmente, são baseados em um conjunto de informações previamente conhecido. Historicamente, pesquisadores em RNA's buscam uma compreensão das capacidades da natureza humana as quais possibilitam que as pessoas construam soluções para problemas que não sejam resolvidos através de métodos tradicionais.

As redes neurais visam, na sua maioria, solucionar problemas de inteligência artificial, modelando sistemas através de circuitos (conexões) que possam simular o sistema nervoso humano, abrangendo a capacidade que o mesmo possui de aprender e agir perante as mais adversas situações apresentadas, bem como adquirir conhecimento através da experiência e da observação.

A complexidade das estruturas elementares das redes neurais biológicas é muito maior do que a dos modelos matemáticos usados nas redes neurais artificiais, demonstrando as dificuldades encontradas para se tentar imitar o funcionamento do sistema nervoso humano. O sistema nervoso é formado por bilhões de células nervosas, enquanto que uma rede neural artificial possui de dezenas a no máximo milhares de unidades de processamento (neurônios).

Uma rede artificial pode ser vista como um conjunto de várias unidades interconectadas (similar à estrutura do cérebro), denominadas de neurônios artificiais, cada qual contendo uma pequena porção local de memória. Estes conceitos foram baseados e fundamentados nos estudos realizados nas células nervosas naturais. Portanto, busca-se aproximar ao máximo o funcionamento das redes neurais artificiais das redes neurais biológicas, na tentativa de buscar a desenvoltura com que o cérebro humano desempenha suas funções.

Alguns modelos de redes neurais artificiais possuem muitos neurônios interconectados, numa estrutura de pesos de conexão e com facilidade de adaptação, proporcionando uma estrutura paralela. A estrutura paralela é desejável pois, se algum neurônio falhar, o efeito na rede como um todo não será significante para o desempenho do sistema, desenvolvendo assim tolerância à falha.

A principio, as redes neurais podem calcular qualquer função computável que é realizada em um computador digital, ou seja, possuem a capacidade de modelar relações lineares e não lineares.

Principais características das RNA's:

- Capacidade de "aprender" através de exemplos e de generalizar este aprendizado de forma a reconhecer elementos similares, que não foram apresentados no conjunto de exemplos (treinamento);
- Bom desempenho em tarefas pouco ou mal definidas, em que falta o conhecimento explícito de como resolvê-las, o aprendizado se dá através de exemplos;
- Robustez a presença de informações falsas ou ausentes, escolha dos elementos no próprio conjunto de treinamento (integridade do conjunto de treinamento);
- No contexto de classificação de padrões, uma rede neural pode fornecer informações sobre quais padrões selecionar em função do grau de confiança apresentado (confiabilidade do conjunto de treinamento);
- Tolerância à falha.

3.2 APLICAÇÕES

Um dos principais objetivos da pesquisa sobre redes neurais artificiais é desenvolver modelos matemáticos das estruturas neurais (não necessariamente baseadas na biologia) que possam efetuar diversas funções. Na maior parte dos casos, os modelos neurais são compostos por conjuntos de elementos não lineares que operam em paralelo e que são classificados de acordo com modelos/padrões relacionados à biologia. Quando um método é criado visando utilizar aspectos de redes neurais, este começa com o desenvolvimento de um neurônio artificial ou computacional baseado no entendimento de estruturas neurais biológicas, seguidas do aprendizado de mecanismos voltados para um determinado conjunto de aplicações e o treinamento do suposto sistema.

Seguem-se, mais detalhadamente, as seguintes etapas:

- Estudo do problema;
- Desenvolvimento de modelos neurais motivados por neurônios biológicos;
- Modelos de estruturas e conexões sinápticas;

- Escolha de um algoritmo de aprendizado (um método de ajuste de pesos ou forças de conexões internodais);
- Construção de um conjunto de treinamento;
- Treinamento propriamente dito;
- Fase de testes;
- Utilização da rede.

As diferenças entre as aplicações, os algoritmos de aprendizagem e as estruturas de interconexões entre os neurônios levam os pesquisadores a desenvolver diferentes modelos (arquiteturas) de redes neurais. Do ponto de vista estrutural, a arquitetura de redes neurais pode ser classificada como estática, dinâmica ou *fuzzy*, podendo ter uma ou múltiplas camadas. Além disso, diferenças computacionais surgem devido à forma como são feitas as conexões entre os neurônios. Estas conexões podem ser *feedfoward*, *backfoward*, lateralmente conectadas, topologicamente ordenadas ou híbridas.

As aplicações de redes neurais podem ser classificadas em diversas classes como:

- Reconhecimento e classificação de padrões;
- Processamento de imagem;
- Visão computacional;
- Identificação e controle de sistemas;
- Processamento de sinais;
- Robótica;
- Filtros contra ruídos eletrônicos;
- Análise de índices e taxas do mercado financeiro;
- Previsão;
- Controle de processos.

3.3 O NEURÔNIO ARTIFICIAL

Assim como o neurônio biológico, o neurônio artificial possui um ou mais sinais de entrada e apenas um sinal de saída. As informações podem ser recebidas através de sensores ou de outros neurônios artificiais que fazem parte da rede. Estes sinais são processados e enviados para a saída. Os sinais de entrada (estímulos) devem chegar até

o neurônio simultaneamente, isto é, todas as informações devem estar disponíveis, para o neurônio artificial, ao mesmo tempo.

O processamento paralelo em computadores seqüenciais (por exemplo, os microcomputadores atuais) pode ser paradoxal, mas não o é, ocorre de fato. A simulação de um ambiente paralelo é possível, e é desta forma que ocorre esse tipo de processamento para as redes neurais. O modelo matemático simula o paralelismo da rede neural através de um algoritmo definido.

Um dos atributos de grande importância do neurônio artificial é o peso. Os pesos, também conhecidos por *pesos sinápticos*, são representados pela letra ***w*** (*weight*) e normalmente representam o grau de importância que determinada entrada possui em relação àquela determinada entrada ou neurônio.

O valor do peso é alterado em função da intensidade do sinal de entrada e, dessa forma, o peso muda o valor representativo da entrada para o neurônio correspondente (processo de aprendizagem).

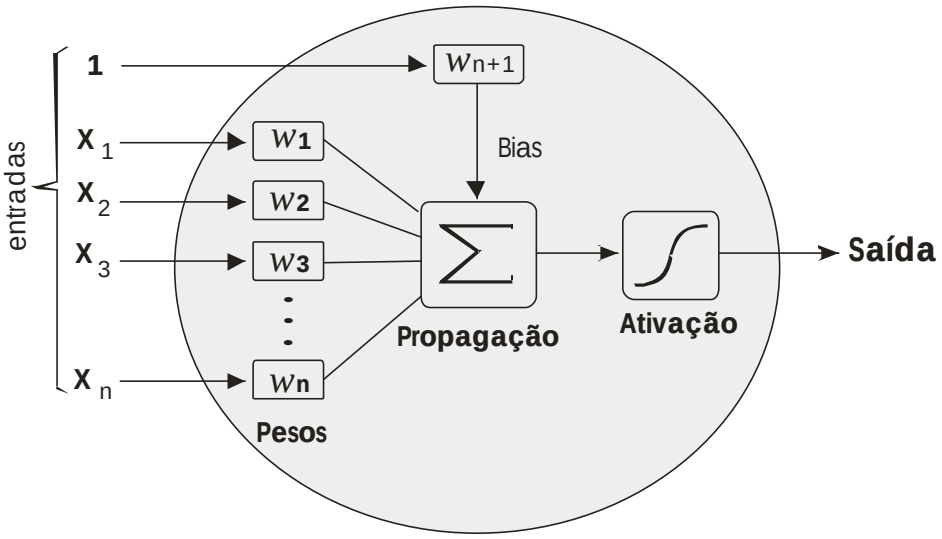


Figura 3.1 – O neurônio artificial.

Matematicamente, os pesos são vistos como um vetor de valores [w_1, w_2, ..., w_n] para um neurônio. O sinal de excitação do neurônio é resultante do somatório do produto dos sinais de entrada representado pelo vetor [x_1, x_2, ..., x_n, 1], pelo vetor de pesos aumentado [w_1, w_2, ..., w_n, w_{n+1}]. O peso w_{n+1}, também chamado de *bias*, tem a função de deslocar o somatório inicial, dentro da função de ativação do neurônio, para ativar ou não a saída.

A próxima tarefa a ser tomada pelo neurônio é de repassar o resultado do somatório para a função de ativação, que será responsável por definir o status da saída do neurônio. Existem diversas funções de ativação, lineares ou não; sua escolha está diretamente ligada à eficiência do neurônio, e consequentemente da rede neural. Por isso, deve-se dar a devida atenção a este processo. A lógica neural expõe que a intensidade dos sinais de entrada dispara, ou não, o sinal do neurônio, fazendo com que este estimule o neurônio seguinte.

Em alguns modelos simples de redes neurais, a função de ativação pode ser apenas um *buffer* do somatório das entradas ponderadas. Neste caso, a função de ativação $f(y) = 1$. Já em modelos mais elaborados, a função de ativação emprega, normalmente, a não linearidade.

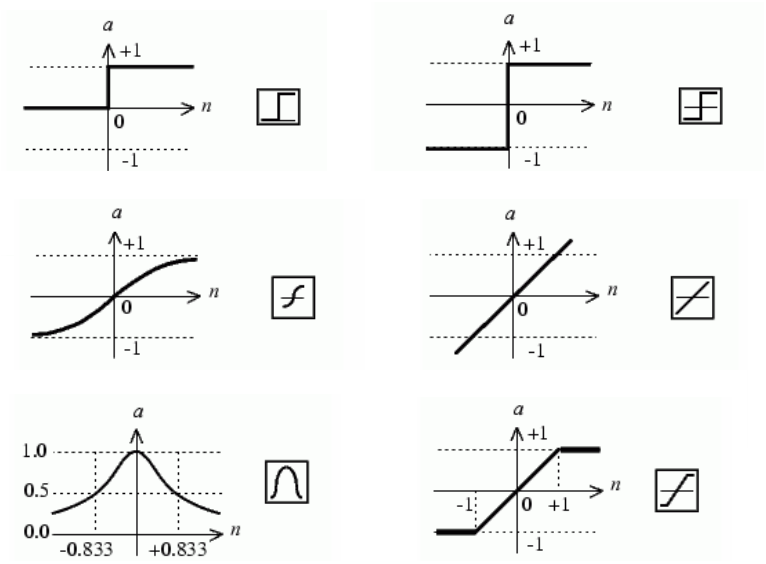


Figura 3.2 – Algumas funções de ativação disponíveis no MATLAB.

Assim como nas redes neurais biológicas, o conjunto de vários neurônios artificiais interconectados formam as redes neurais artificiais.

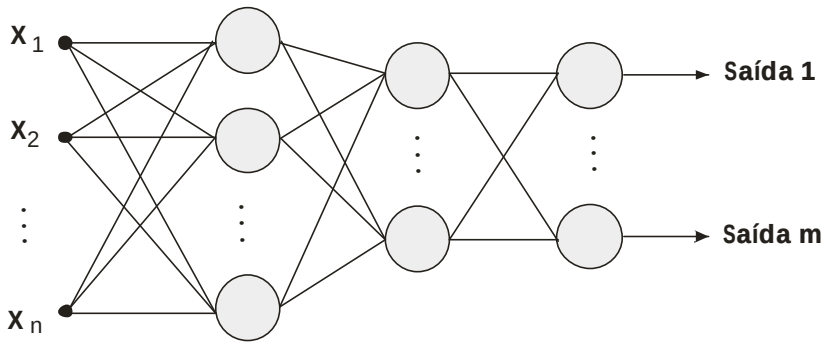


Figura 3.3 – Rede neural artificial.

3.4 ARQUITETURAS

Um outro importante aspecto, responsável diretamente sobre a eficiência da rede, é sua arquitetura, ou seja, a quantidade de neurônios e de que forma eles são conectados. Biologicamente, as redes neurais são organizadas de forma tridimensional por componentes microscópicos, contendo bilhões de neurônios. Já em uma rede neural artificial, existe uma forte restrição do número de neurônios e camadas devido ao enorme esforço computacional requerido.

Uma rede neural artificial pode ter uma ou mais camadas. As redes que possuem uma única camada possuem apenas um nó entre as entradas e saídas da rede. Esse tipo de rede é indicado para solução de problemas linearmente separáveis.

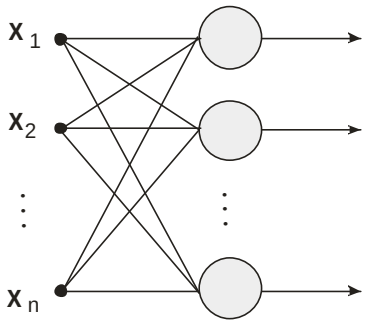


Figura 3.4 – RNA de uma única camada.

Já as redes multicamadas possuem mais de uma camada, entre as entradas e as saídas, e podem possuir as chamadas camadas ocultas ou escondidas.

As camadas ocultas têm, como característica diferenciada das demais camadas, o fato de não ter contato com o mundo externo. Os sinais são passados para os outros neurônios obedecendo às funções de transferência que cada neurônio possui.

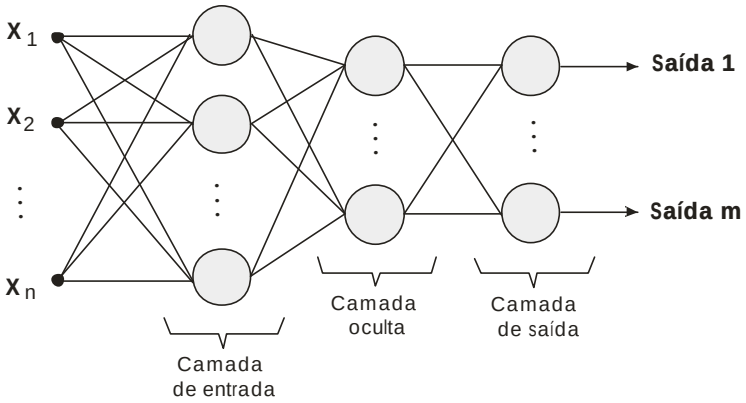


Figura 3.5 – RNA multicamada.

Para a escolha da arquitetura ideal, deve-se levar em conta que o aumento das camadas ocultas atua diretamente no aumento da não linearidade da rede e que camadas excessivamente numerosas são, muitas vezes, desnecessárias e provocam um demasiado esforço computacional. Normalmente o número de neurônios, nas camadas de entrada e saída, é determinado pela natureza dos dados disponíveis e necessários na saída da rede. Para o número de neurônios nas camadas ocultas existem diversos estudos a respeito da melhor configuração para problemas específicos mas, via de regra, não existe uma resposta trivial. Na maioria das vezes, utiliza-se um número intermediário entre as quantidades de neurônios das camadas mais próximas (camada anterior e posterior).

3.5 APRENDIZADO

As RNA's possuem a capacidade de aprender por exemplos, determinando a intensidade de conexões entre os neurônios pertencentes à rede. Logo, um conjunto de procedimentos definidos para ajustar os parâmetros de uma RNA, a fim que a mesma possa aprender uma determinada função, é chamado de algoritmo de aprendizado. A designação de uma RNA, na resolução de um determinado problema, passa inicialmente por um processo de aprendizagem, em que a rede procura extrair informações relevantes de padrões de informações apresentados a ela, modelando uma representação própria.

Para o aprendizado das redes neurais, foram propostos diversos métodos de treinamento, sendo estes subdivididos em dois paradigmas principais: aprendizado supervisionado e o não supervisionado. Para estes dois modelos existem vantagens e desvantagens que serão expostas a seguir.

3.5.1 APRENDIZADO SUPERVISIONADO

A grande maioria das RNA's tem utilizado o treinamento supervisionado. Neste tipo de aprendizado, a saída da rede neural é comparada com a saída desejada e seus pesos são ajustados de acordo com o algoritmo empregado até a minimização dos erros encontrados.

O ajuste sináptico é dependente da diferença entre o valor esperado e do sinal atual de saída. Desta maneira, o método de aprendizado tenta minimizar essa função de custo.

Com o aprendizado supervisionado, as redes neurais devem ser treinadas antes de serem usadas. O treinamento consiste na apresentação dos sinais de entrada e saída desejada (dados do tipo causa-efeito) à rede. A fase de treinamento pode consumir uma grande fatia de tempo. Em alguns sistemas esse processo pode levar semanas de processamento. O treinamento é considerado completo quando a rede alcança certo nível de desempenho definido, a priori, pelo projetista, para se evitar a especialização da rede nos dados de treinamento. Quando não há mais a necessidade de aprendizado, os pesos permanecem inalterados para sua aplicação. Alguns tipos de redes neurais permitem um treinamento contínuo, com uma taxa muito baixa de aprendizado, enquanto a mesma está em operação. Este processo auxilia as redes a se adaptarem gradualmente às condições de mudança.

O conjunto de treinamento precisa ser suficientemente grande para conter as informações necessárias para que a rede aprenda os moldes e as relações importantes entre os dados apresentados.

Após o treinamento supervisionado, a eficiência da rede deverá ser analisada por um conjunto de dados não utilizados na fase de treinamento, denominado conjunto de teste. Esta avaliação tem como objetivo assegurar que a rede foi capaz de generalizar e "aprender" com os dados apresentados na fase de treinamento, e não simplesmente memorizar o conjunto de dados do treinamento. Caso a fase de teste indique resultados indesejáveis, um novo treinamento deverá ser feito ou deve-se iniciar, novamente, a fase de treinamento.

3.5.2 APRENDIZADO NÃO SUPERVISIONADO

Na aprendizagem não-supervisionada ou auto-organizada, não há um professor externo ou um critico para supervisionar o processo de aprendizado. Em vez disso, são dadas condições para realizar uma medida independente da tarefa da qualidade da representação que a rede deve aprender, e os parâmetros livres da rede são otimizados em relação a esta medida. Uma vez que a rede tenha se ajustado às regularidades estáticas dos dados de entrada, ela desenvolve a habilidade de formar representações internas para codificar as características da entrada e, desse modo, de criar automaticamente novas classes.

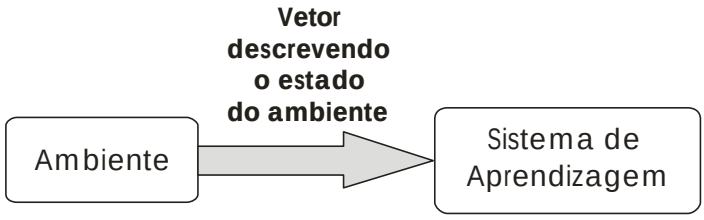


Figura 3.6 – Diagrama em blocos da aprendizagem não-supervisionada.

Para realizar a aprendizagem não-supervisionada, pode-se utilizar a regra de aprendizagem competitiva. Utiliza-se, por exemplo, uma rede neural de duas camadas – uma camada de entrada e uma camada competitiva. A camada de entrada recebe os dados disponíveis. A camada competitiva consiste de neurônios que competem entre si (de acordo com uma regra de aprendizagem) pela "oportunidade" de responder às características contidas nos dados de entrada. Na sua forma mais simples, a rede opera de acordo com uma estratégia do tipo "o vencedor leva tudo". Nesta estratégia o neurônio com a maior entrada total "ganha" a competição e se torna ligado; todos os outros neurônios, então, se tornam desligados [Haykin, 2001].

3.6 O ALGORITMO BACK PROPAGATION

O método de aprendizado *backpropagation* pode ser aplicado a qualquer rede que usufrui de uma função de ativação diferenciável e aprendizado supervisionado. Assim como a regra delta, sua otimização é baseada no gradiente descendente, que ajusta os pesos para reduzir o erro da rede.

Durante a fase de treinamento, os sinais/padrões de entrada são apresentados à rede artificial em uma determinada ordem. Cada padrão de treinamento é propagado adiante, camada após camada, até a produção do sinal/padrão de saída. A saída computada pela rede é, então, comparada com a saída desejada. Esta comparação irá gerar um valor que determinará o erro. Este erro será utilizado para calcular a função de custo de otimização e seu valor resultará ou não, no ajuste dos pesos sinápticos da rede num sentido oposto à propagação dos sinais de treinamento.

De acordo com [Haykin, 2001], o sinal de erro na saída do neurônio *j*, no padrão de treinamento *n*, é definido por:

$$e_j(n) = d_j(n) - y_j(n) \qquad \text{Equação 3.1}$$

Onde n indica o n-ésimo padrão de treinamento, $d_j(n)$ representa o valor desejado e $y_j(n)$, o valor efetivo na saída do neurônio.

Define-se o valor instantâneo da energia do erro no neurônio j como $\frac{1}{2} e_j^2(n)$. O valor instantâneo ?(n) da energia total do erro é obtido os termos $\frac{1}{2} e_j^2(n)$ de todos os neurônios da camada de saída; são os únicos neurônios "visíveis" para os quais os sinais de erro podem ser calculados diretamente.

$$x(n) = \frac{1}{2} \sum e_j^2(n) \qquad \text{Equação 3.2}$$

Considere que N represente o número total de padrões (exemplos) contidos no conjunto de treinamento. A energia média do erro quadrático é obtida somando-se os ?(n) para todos os n e, então, normalizando em relação ao tamanho do conjunto N:

$$x_{med} = \frac{1}{N} \sum_{n=1}^{N} x(n) \qquad \text{Equação 3.3}$$

A energia instantânea do erro ?(n) e, consequentemente, a energia média do erro $?_{med}$, é uma função de todos os parâmetros livres (pesos sinápticos e bias) da rede. Para um dado conjunto de treinamento, $?_{med}$ representa a *função de custo* como uma medida do desempenho de aprendizado. O objetivo do processo de aprendizagem é ajustar os parâmetros livres da rede para minimizar a função de custo $?_{med}$. Considere um método simples de treinamento cujos pesos são atualizados de padrão em padrão até formar uma *época*, isto é, uma apresentação completa do conjunto de treinamento inteiro que está sendo processado. Os ajustes dos pesos são realizados de acordo com os respectivos erros calculados para cada padrão apresentado à rede. A média aritmética destas alterações individuais de peso sobre o conjunto de treinamento é, portanto, uma estimativa da alteração real que resultaria da modificação dos pesos baseada na minimização da função de custo $?_{med}$ sobre o conjunto de treinamento inteiro.

A figura 3.7 representa o neurônio j, sendo alimentado por um conjunto de sinais funcionais produzidos por uma camada de neurônios à sua esquerda.

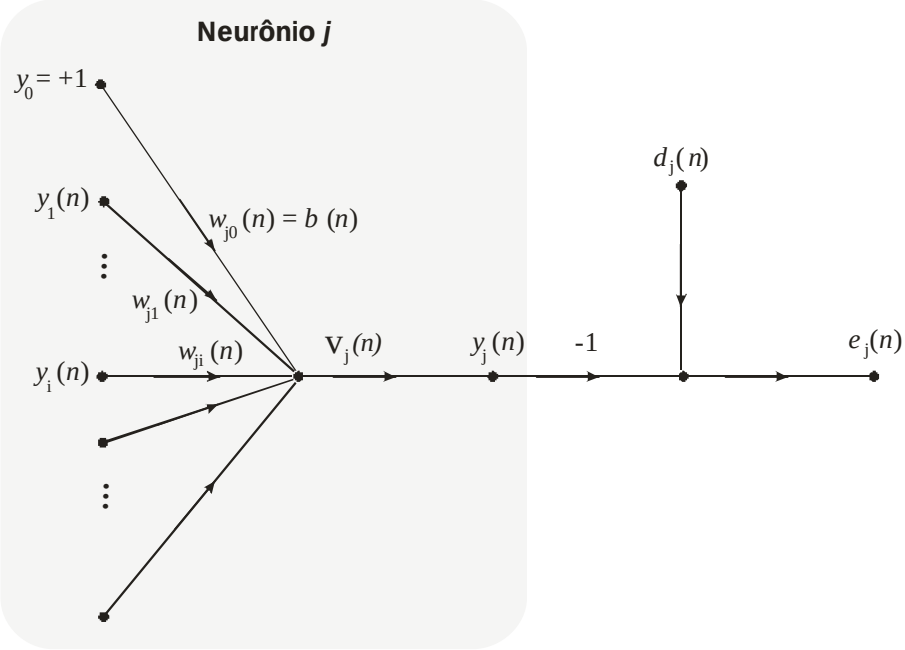


Figura 3.7 – Detalhes do neurônio de saída *j*.

Onde:

$$u_j(n) = \sum_{i=0}^{m} w_{ji}(n)\, y_i(n) \qquad \text{Equação 3.4}$$

m = número total de entradas (excluindo o bias) aplicadas ao neurônio *j*.

O peso sináptico w_{j0} (correspondendo à entrada fixa y_0 =+1) é igual ao bias b_j aplicado ao neurônio *j*. Assim, o sinal funcional $y_j(n)$ que aparece na saída do neurônio *j* na iteração *n* é:

$$y_j(n) = j_j\left(u_j(n)\right) \qquad \text{Equação 3.5}$$

O algoritmo de retro propagação aplica uma correção ? $w_{ji}(n)$ ao peso sináptico $w_{ji}(n)$, que é proporcional à derivada parcial ??(*n*)/?$w_{ji}(n)$. De acordo com a *regra da cadeia* do cálculo, pode-se expressar este gradiente como:

$$\frac{\partial x(n)}{\partial w_{ji}(n)} = \frac{\partial x(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial u_j(n)} \frac{\partial u_j(n)}{\partial w_{ji}(n)} \qquad \text{Equação 3.6}$$

A derivada parcial ??(*n*) /?$w_{ji}(n)$ representa um *fator de sensibilidade*, determinando a direção de busca no espaço de pesos, para o peso sináptico w_{ji}.

Realizando as devidas substituições na equação 3.6, para o neurônio *j* na última camada, tem-se:

$$\Delta w_{ji}(n) = h \; d_j(n) \; y_i(n) \qquad \text{Equação 3.7}$$

Onde o *gradiente local* $d_j(n)$ é definido por:

$$d_j(n) = e_j(n) \; j_j'(u_j(n)) \qquad \text{Equação 3.8}$$

E para o neurônio *j* em camada oculta:

$$d_j(n) = j_j'(u_j(n)) \sum_k d_k(n) \; w_{kj}(n) \qquad \text{Equação 3.9}$$

Onde *k* denota o neurônio de saída e *j* o neurônio oculto.

O gradiente local aponta para as modificações necessárias nos pesos sinápticos. De acordo com a equação 3.8, o gradiente local $d_j(n)$ para o neurônio de saída j é igual ao produto do sinal de erro $e_j(n)$ correspondente para aquele neurônio pela derivada $f_j'(?_j(n))$ da função de ativação associada.

Das equações acima, nota-se que um fator-chave envolvido no cálculo do ajuste de peso $? w_{ji}(n)$ é o sinal de erro $e_j(n)$ na saída do neurônio *j*. Neste contexto, podemos identificar dois casos distintos, dependendo de onde se encontra o neurônio *j*. No caso 1, o neurônio *j* é um nó de saída. Neste caso, cada nó de saída da rede é suprido com uma resposta desejada particular, fazendo com que o cálculo do sinal de erro associado seja direto. No caso 2, o neurônio *j* é um nó oculto. Apesar dos neurônios ocultos não serem acessíveis diretamente, eles compartilham a responsabilidade por qualquer erro cometido na saída da rede. A questão, entretanto, é saber como penalizar ou recompensar os neurônios ocultos pela sua parcela de responsabilidade. Este problema é resolvido retro propagando os sinais de erro através da rede.

Na aplicação do algoritmo de retro propagação, distinguem-se dois passos distintos de computação. O primeiro passo é conhecido como passo para frente, ou propagação, e o segundo como passo para trás, ou retro propagação.

No passo para frente, os pesos sinápticos se mantêm inalterados em toda a rede e os sinais funcionais da rede são calculados individualmente, neurônio por neurônio. O sinal funcional que aparece na saída do neurônio j é calculado como:

$$y_j(n) = \varphi_j(u_j(n)) \qquad \text{Equação 3.10}$$

Onde $?_j(n)$ é o campo local induzido do neurônio j, definido por:

$$u_j(n) = \sum_{i=0}^{m} w_{ji}(n)\, y_i(n) \qquad \text{Equação 3.11}$$

Onde m é o número total de entradas (excluindo o bias) aplicadas ao neurônio j, e $w_{ji}(n)$ é o peso sináptico que conecta o neurônio i ao neurônio j, e $y_i(n)$ é o sinal de entrada do neurônio j, que é à saída do neurônio i. Se o neurônio j estiver na primeira camada oculta da rede, $m = m_0$ e o índice i se refere ao i-ésimo terminal de saída da rede, para o qual escrevemos:

$$y_i(n) = x_i(n) \qquad \text{Equação 3.12}$$

Onde $x_i(n)$ é o i-ésimo elemento do vetor de padrão de entrada. Se, por outro lado, o neurônio j estiver na camada de saída da rede, $m = m_L$ e o índice j se refere ao j-ésimo terminal de saída da rede, para escrevemos:

$$y_j(n) = o_j(n) \qquad \text{Equação 3.13}$$

Onde $o_j(n)$ é o j-ésimo elemento do vetor de saída. Esta saída é comparada com a resposta desejada $d_j(n)$, obtendo-se o sinal de erro $e_j(n)$ para o j-ésimo neurônio da saída. Assim, a fase de propagação começa na primeira camada oculta, com a apresentação do vetor de entrada, e termina na camada de saída calculando o sinal de erro de cada neurônio desta camada.

O passo de retro propagação, por outro lado, começa na camada de saída passando-se o d (gradiente local) de cada neurônio. Este processo recursivo permite que os pesos sinápticos sofram modificações de acordo com a equação 3.7. Para um neurônio localizado na camada de saída, o d é simplesmente igual ao sinal de erro

daquele neurônio multiplicado pela primeira derivada da sua não-linearidade. Assim, utilizamos à equação 3.7 para calcular as modificações dos pesos de todas as conexões que alimentam a camada de saída. Dados os d para os neurônios da camada de saída, utilizamos, a seguir, a equação 3.9 para calcular os d para todos os neurônios na penúltima camada, e consequentemente as modificações dos pesos de todas as conexões que a alimentam. A computação é recursiva e contínua, camada por camada, propagando as modificações para todos os pesos sinápticos da rede. Note que para a apresentação de cada exemplo de treinamento, o padrão de entrada está fixo (“preso”) durante todo o ciclo, englobando o passo de propagação seguido pelo passo de retro propagação.

Capítulo 4

ESTUDO DO PRÉ-PROCESSAMENTO DOS SINAIS

4.1 VISÃO GERAL

Cada vez mais, cenários complexos que há algum tempo só existiam em filmes de ficção científica estão se tornando realidade devido ao incrível aumento da capacidade computacional e de comunicação de dados. Reconhecimento de faces, busca por conteúdo de imagens e vídeo, sistemas automáticos de vigilância eletrônica, monitoramento de tráfego, detecção precoce de tumores são apenas alguns dos recentes resultados de projetos da comunidade científica na área de processamento digital de sinais e reconhecimento de padrões.

O processamento digital dos dados é um dos grandes responsáveis pelo enorme sucesso e robustez desses projetos inovadores. Cada vez mais eficientes, são capazes de aumentar a confiabilidade e comprimir os dados sem perda de informação. Por isso, o processamento digital dos sinais, em reconhecimento dos padrões, representa um papel de grande importância para o sistema de classificação.

Esta seção descreve, então, uma forma de pré-processamento de sinais digitais através da análise de wavelets, que auxiliará o sistema classificador eliminando os ruídos presentes no sinal de entrada. A seguir é apresentada, de forma sucinta, a técnica empregada.

4.2 WAVELETS

Wavelets ou transformada de wavelets nada mais é do que um conjunto de ferramentas matemáticas utilizadas para decomposição, filtragem ou compressão de sinais ou imagens. Nas últimas décadas, a crescente utilização dessa técnica se dá, sobretudo, ao avanço computacional e tecnológico.

A transformada de wavelets surgiu da necessidade de se extrair as informações omitidas na transformada de Fourier. Wavelets surgiu para não só decompor curvas em diferentes sinais, mas sim dizer exatamente quando esses sinais ocorrem.

A seguir, segue um comparativo entre as transformadas que justifica tal avanço.

4.2.1 ANÁLISE DE FOURIER

A análise de Fourier é uma das ferramentas mais conhecidas e utilizadas para analisar sinais temporais. Ela é capaz de quebrar o sinal original em senóides ou cossenóides de diferentes freqüências e amplitudes. Um outro grande diferencial da análise de Fourier está em transformar um sinal no domínio do tempo para o domínio da freqüência.

Figura 4.1 – Análise de Fourier.

Para muitos sinais, a análise de Fourier é extremamente útil, pois a freqüência dos sinais contém grande importância. Então por que precisamos de outras técnicas de análise, como wavelets?

A análise de Fourier tem uma séria desvantagem. Ao transferir um sinal no domínio do tempo para o domínio da freqüência, informações importantes do tempo são perdidas. Quando olhamos a transformada de Fourier de um sinal, é impossível dizer quando um evento particular ocorre.

Se as propriedades do sinal não se modificam sobre o tempo, temos um sinal estacionário, que não contém informações importantes sobre particularidades. Entretanto, muitos sinais contêm sinais não-estacionários ou características transitórias como descontinuidades, pulsos, começo e fim de eventos, etc. Estas características contêm a parte mais importante do sinal, e a análise de Fourier não consegue identificá-las.

4.2.2 JANELAMENTO DA ANÁLISE DE FOURIER (Short-Time Fourier Analysis)

Para corrigir tal deficiência, Dennis Gabor (1946) adaptou a transformada de Fourier para analisar apenas uma pequena parte do sinal no domínio do tempo – utilizando a técnica de janelamento do sinal. A adaptação de Gabor, chamada de Short-

Time Fourier Transform (STFT), mapeia o sinal em uma função bidimensional: no tempo e na freqüência.

Figura 4.2 – Janelamento da análise de Fourier (STFT).

O janelamento da análise de Fourier (STFT) representa um comportamento limitado entre o tempo e a freqüência do sinal. Contém algumas informações de quando e quais freqüências ocorrem no sinal. Entretanto, você só pode obter estas informações com uma precisão limitada, e tal precisão é determinada pelo tamanho da janela.

Você só pode ter uma boa relação entre tempo e freqüência, quando utilizar um tamanho particular para a janela de análise, e esta janela será do mesmo tamanho para todas as freqüências. Muitos sinais requerem uma abordagem mais flexível, uma vez que se faz necessário variar o tamanho da janela para se determinar mais precisamente as freqüências no tempo.

4.2.3 ANÁLISE DE WAVELETS

A análise de wavelet representa o próximo passo lógico: a técnica de janelamento com variação do tamanho por regiões. A análise de wavelet utiliza grandes regiões quando queremos obter informações em baixa freqüência, e pequenas regiões quando queremos obter informações em alta freqüência.

Figura 4.3 – Análise de Wavelets.

O desenho abaixo ilustra o contraste dos diferentes domínios mencionados anteriormente.

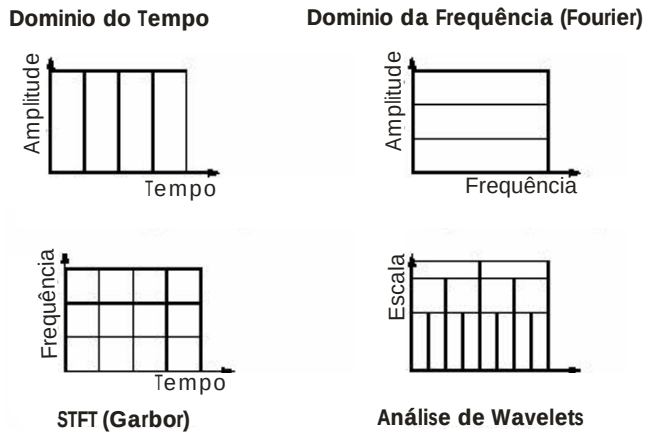


Figura 4.4 – Comparativo gráfico dos diferentes domínios.

Uma outra diferença na análise de wavelet está em utilizar o domínio da escala-tempo, em que escala é o fator que relaciona o tamanho da curva wavelet escolhida.

4.2.3.1 O QUE É A ANÁLISE DE WAVELETS?

Wavelet é uma forma de onda (uma curva) com duração limitada e média igual a zero. Uma comparação de ondas senoidais e wavelets:

Senóides não tem tamanho limitado, e estendem-se ao infinito; são simétricas e de fácil predição.

Wavelets tem duração limitada; tendem a ser irregulares e assimétricas.

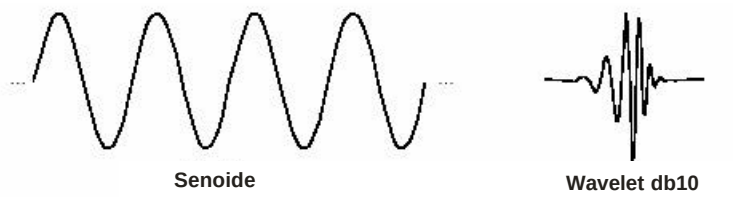


Figura 4.5 – Comparativo entre um sinal senoidal e uma wavelet mãe.

A análise de Fourier consiste em quebrar o sinal original em várias senóides de diferentes freqüências. Similar à análise de Fourier, a análise de wavelet quebra o sinal original em várias curvas de wavelets de diferentes escalas e translações.

4.2.3.2 O QUE A ANÁLISE DE WAVELETS PODE FAZER?

Uma das maiores vantagens é a de conseguir fazer uma análise local – analisar uma área em particular dentro de um grande sinal.

Considere um sinal senoidal com pequenas descontinuidades, similares a ruídos.

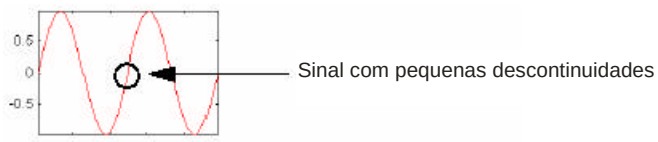


Figura 4.6 – Sinal com pequenas descontinuidades.

O resultado da análise de Fourier pode ser observado abaixo:

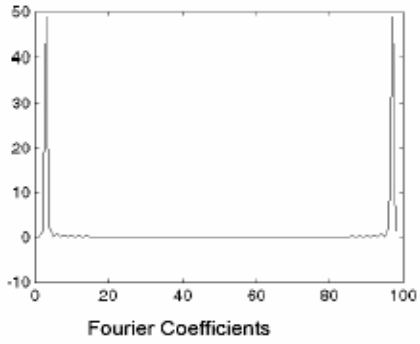


Figura 4.7 – Resultado da análise de Fourier.

Os dois picos representam as freqüências das senóides puras que constituem o sinal original, e são omitidas informações de quando ocorrem as particularidades.

Já os coeficientes da análise de wavelet são capazes de detectar além das freqüências existentes, o *quando* exatamente ocorrem particularidades.

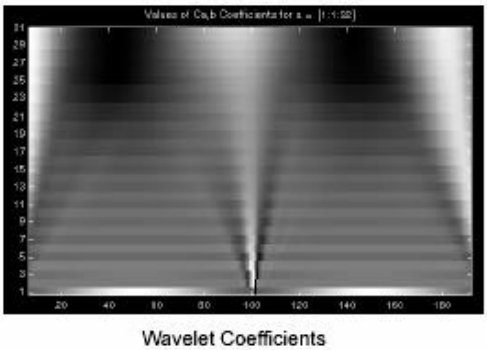


Figura 4.8 – Resultado da análise de Wavelets.

A análise de wavelet é capaz de revelar outros sinais e aspectos como: descontinuidades em grandes derivadas; pontos de quebra dos sinais; início e fim do sinal; e outras particularidades. Por outro lado, por utilizar meios diferentes de analisar sinais, é possível retirar ruídos e comprimir os dados sem perda de informação.

4.2.4 FILTRO WAVELETS

A utilização da análise de wavelets para a filtragem de sinais e imagens vem se destacando em reconhecimento de padrões pela robustez apresentada, principalmente a partir da década de 90. As *wavelets mãe* ortonormais e continuamente diferenciáveis, concebidas por Ingrid Daubechies (1987), tornaram-se um marco na análise de wavelets [Silva, 2000].

Abaixo segue exemplo de aplicação:

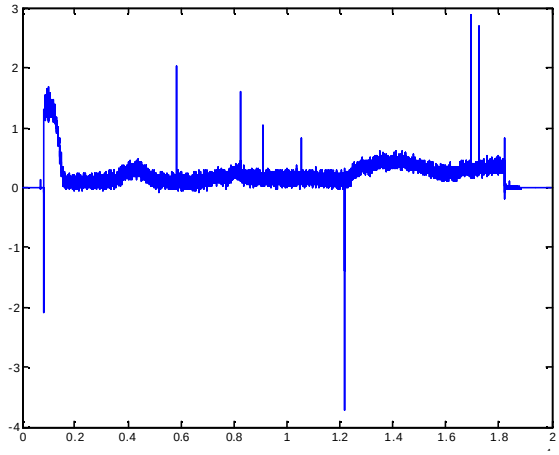


Figura 4.9 – Sinal sem filtragem.

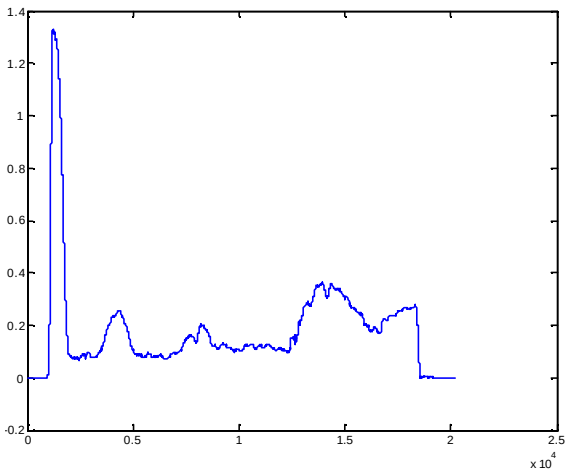


Figura 4.10 – Sinal pós filtro wavelets.

Capítulo 5

DESENVOLVIMENTO

5.1 ESTUDO DAS CURVAS

O banco de dados inicial é composto de 165 curvas, entre abertura e fechamento da chave. As curvas possuem cerca de 20.000 pontos cada e uma taxa de amostragem de 0.5 ms (milisegundos).

A tabela 5.1 ilustra a quantidade de curvas existentes no banco de dados para cada tipo de teste (cada teste representa um estado de funcionamento do seccionador):

Tabela 5.1 – Quantidade de curvas em cada tipo de padrão.

	ABERTURA		FECHAMENTO	
	Nome	**Qtde**	**Nome**	**Qtde**
1	Teste 7	27	Teste 6	27
2	Teste 27	3	Teste 8	3
3	Teste 30	3	Teste 11	3
4	Teste 31	3	Teste 12	3
5	Teste 32	3	Teste 13	3
6	Teste 33	3	Teste 14	3
7	Teste 34	3	Teste 15	3
8	Teste 35	3	Teste 16	3
9	Teste 36	3	Teste 17	3
10	Teste 37	3	Teste 18	3
11	Teste 44	3	Teste 25	3
12	Teste 45	3	Teste 26	3
13	Teste 46 A	3	Teste 46 F	3
14	Teste 47 P1 A	3	Teste 47 P1 F	3
15	Teste 47 P2 A	3	Teste 47 P2 F	3
16	Teste 47 P3 A	3	Teste 47 P3 F	3
17	Teste 48 A	3	Teste 48 F	3
18	Teste 49 A	3	Teste 49 F	3
19	Teste 53 A	3	Teste 53 F	3
20			Teste 56 F	3
	Total	**81**	**Total**	**84**

As figura 5.1 ilustra uma curva de fechamento (teste 6) e a figura 5.2 uma curva de abertura (teste 7) contidas no banco de dados inicial:

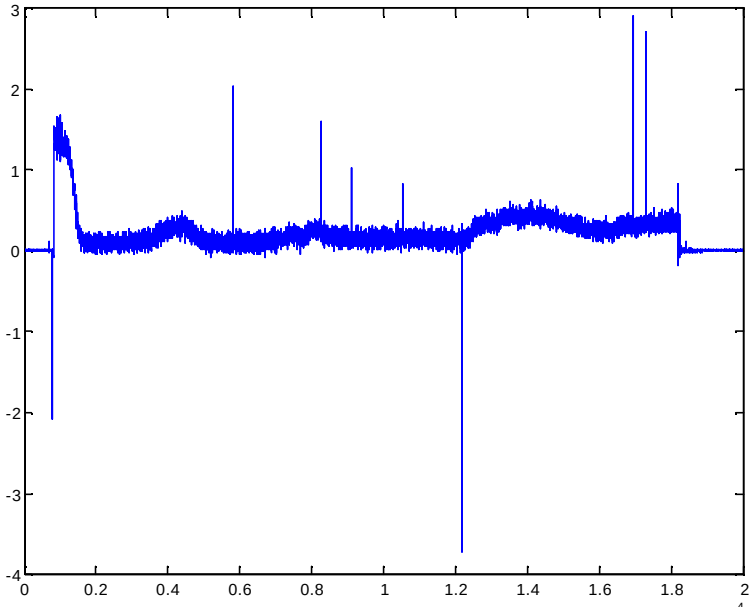


Figura 5.1 – Teste 6, curva de torque do motor de acionamento, na manobra de fechamento da chave seccionadora.

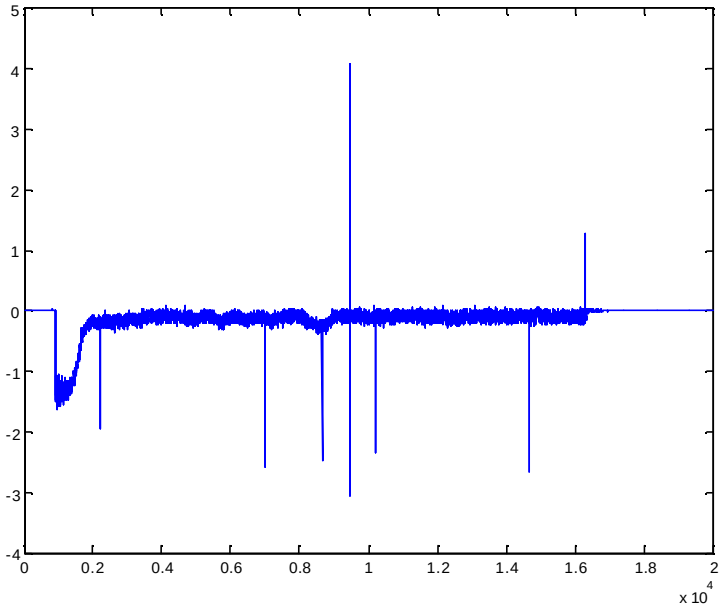


Figura 5.2 – Teste 7, curva de torque do motor de acionamento, na manobra de abertura da chave seccionadora.

Nota-se, nas curvas sem pré-processamento, a existência de muitos ruídos e que o inicio de cada curva se dá em tempos diferentes. Para eliminar tais transtornos, faz-se necessário o uso de um pré-processamento inicial que elimine os ruídos e padronize o inicio e o final de cada curva.

5.2 DESENVOLVIMENTO DO PRÉ-PROCESSAMENTO

Como descrito anteriormente, a fase de pré-processamento está diretamente ligada à eficiência do classificador. A correta manipulação dos dados otimizará os resultados do classificador.

Para os dados de torque, de abertura e de fechamento da chave seccionadora, o pré-processamento deverá, respectivamente:

1. Eliminar os ruídos presentes;
2. Padronizar o início e o final de cada curva;
3. Determinar se a curva é de Abertura ou de Fechamento;
4. Obter os segmentos significativos de cada curva (diferentes para abertura e fechamento);
5. Reduzir os segmentos para 30 pontos (que será à entrada da rede).

A figura, a seguir, descreve os procedimentos da fase de pré-processamento, em diagrama de blocos:

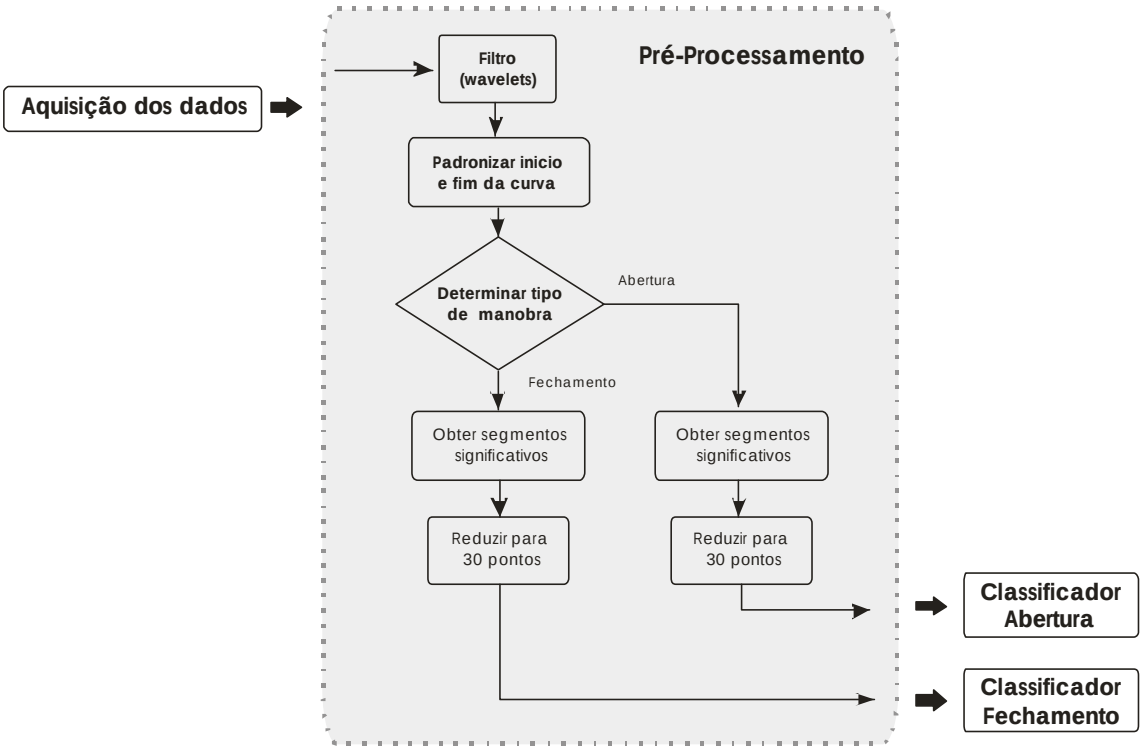


Figura 5.3 – Diagrama de blocos do pré-processamento.

5.2.1 ELIMINANDO OS RUÍDOS

Para eliminar os ruídos presentes, utilizou-se a analise de wavelets como um filtro, conforme demonstrado no capítulo anterior. Os resultados podem ser observados nas figuras seguintes:

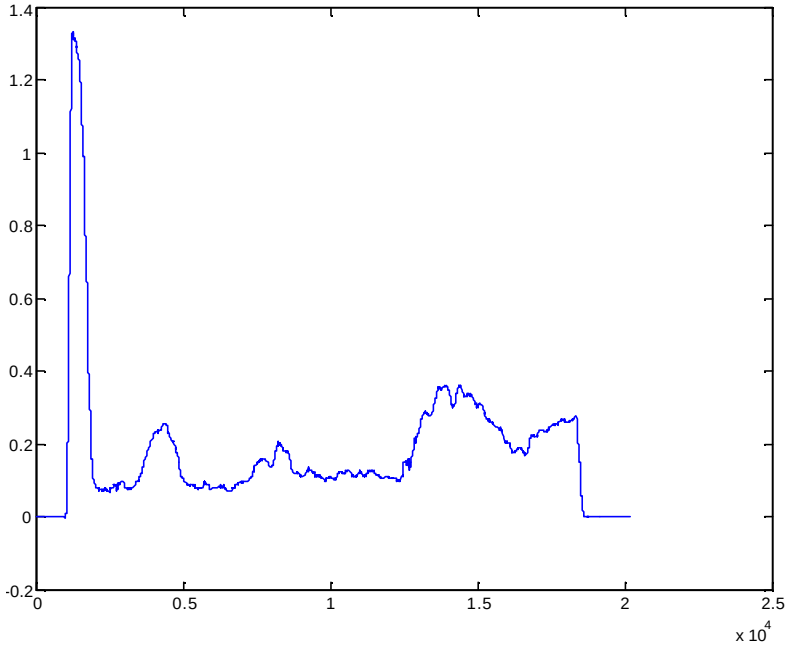


Figura 5.4 – Curva de Fechamento pós-análise de wavelets.

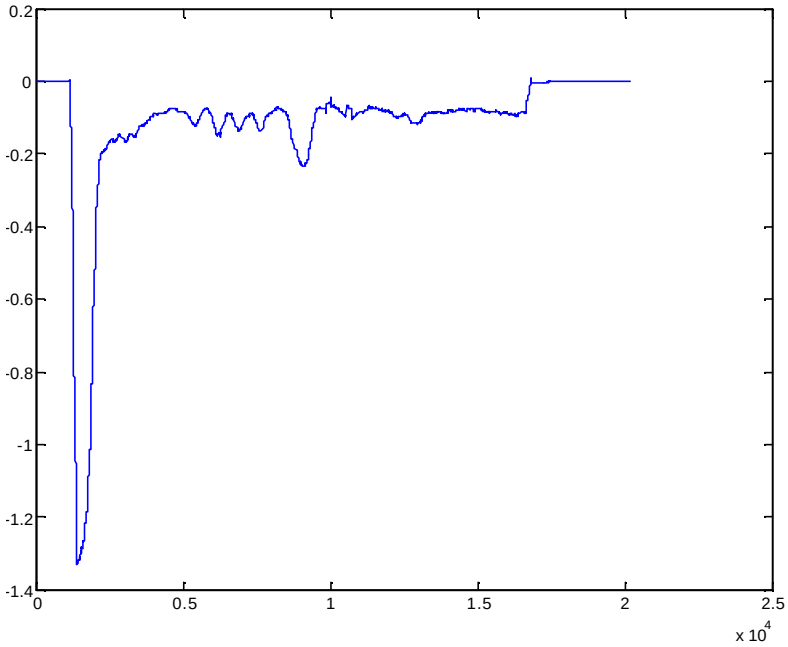


Figura 5.5 – Curva de Abertura pós-análise de wavelets.

5.2.2 PADRONIZANDO AS CURVAS

A padronização do início e do final de cada curva se faz necessário para se obter uma referência de início e fim de cada evento.

Determinou-se o inicio do evento o primeiro ponto mais próximo (em módulo) de 0,05.

O final de cada evento se dará no ponto 18.000 subseqüente ao início do evento. Ou seja, cada curva terá 18.000 pontos.

As figuras, a seguir, ilustram os resultados obtidos:

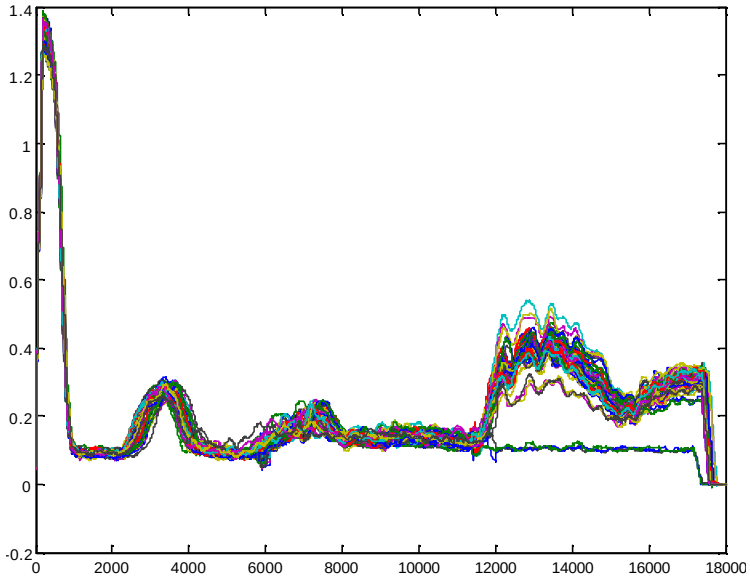


Figura 5.6 – Curvas de fechamento padronizadas.

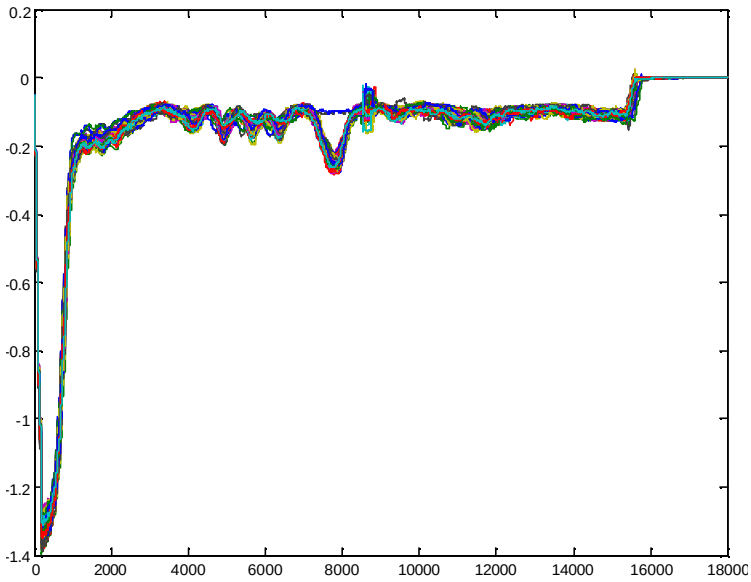


Figura 5.7 – Curvas de abertura padronizadas.

5.2.3 DETERMINANDO O TIPO DE FUNCIONAMENTO

Como o movimento das manobras é oposto (abertura e fechamento da chave seccionadora), os dados coletados também os são.

De acordo com o banco de dados utilizados, as curvas que possuem valores positivos são de fechamento, e as curvas que possuem valores negativos são curvas de abertura. Este conceito foi utilizado para determinar se o evento é de fechamento ou de abertura da chave seccionadora.

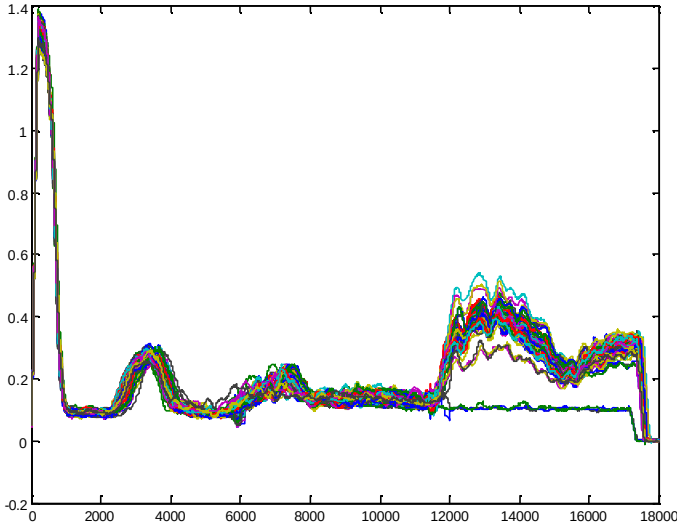


Figura 5.8 – Curvas de fechamento = valores positivos.

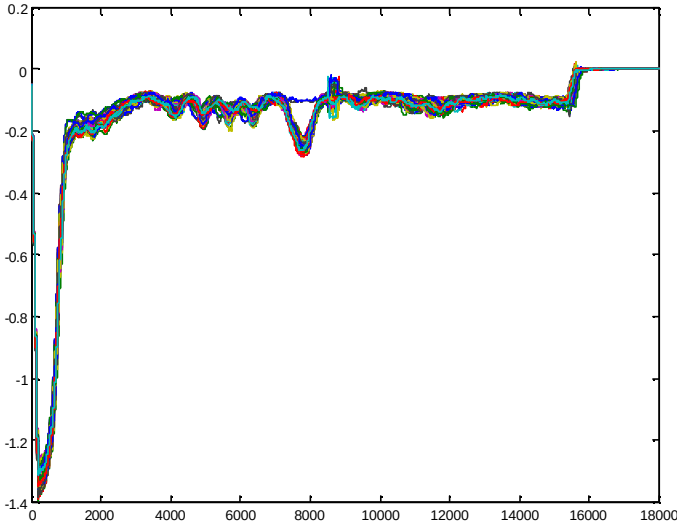


Figura 5.9 – Curvas de abertura = valores negativos.

Para simplificar a visualização das curvas a partir deste ponto, todas as curvas de abertura serão convertidas para valores em módulo, pois a informação contida em sua natureza já foi extraída.

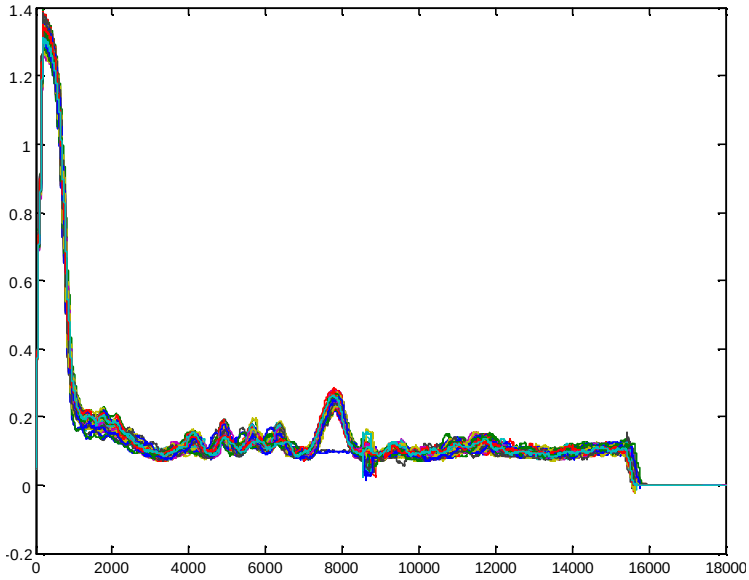


Figura 5.10 – Curvas de abertura em módulo.

5.2.4 OBTENDO OS SEGMENTOS SIGNIFICATIVOS

Pela análise gráfica, observa-se que muitos pontos são comuns a todas as curvas, tanto para fechamento quanto para abertura da chave. O intuito dessa fase será obter os segmentos mais significativos em cada tipo de curva (abertura ou fechamento), para aumentar a precisão do classificador.

A escolha será feita somente em regiões que o padrão de cada teste seja válido, ou seja, que exista um padrão para cada tipo de teste. Isso se faz necessário, pois há possibilidade de se encontrar regiões que embora contenham valores diferentes em cada tipo de teste, não existe lógica ou não são parecidos para testes idênticos.

Para as curvas de fechamento da chave, o segmento significativo foi extraído dos pontos 12.001 até o ponto 17.500, conforme ilustra a figura abaixo:

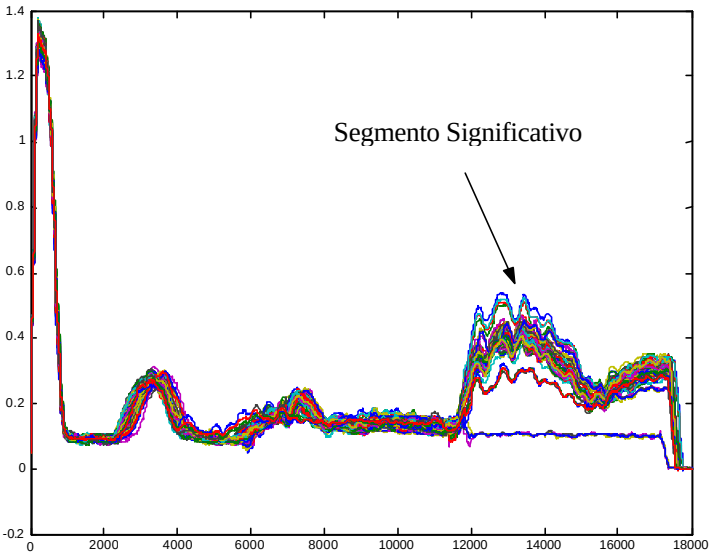


Figura 5.11 – Segmento significativo para as curvas de fechamento.

Para as curvas de abertura da chave, o primeiro segmento significativo foi extraído do ponto 1.001 até o ponto 2.000, e o segundo do ponto 4.000 até 8.500, conforme figura abaixo:

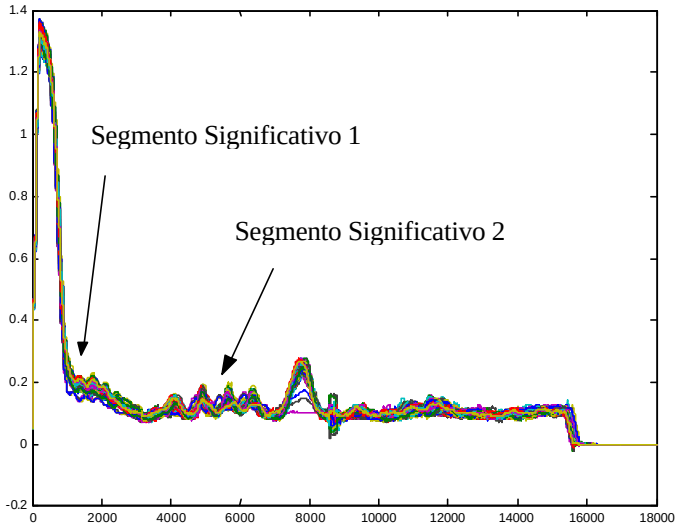


Figura 5.12 – Segmentos significativos para as curvas de abertura (em módulo).

Assim, as curvas que inicialmente tinham cerca de 20.000 pontos, foram resumidas em 5.500 pontos (ditos significativos).

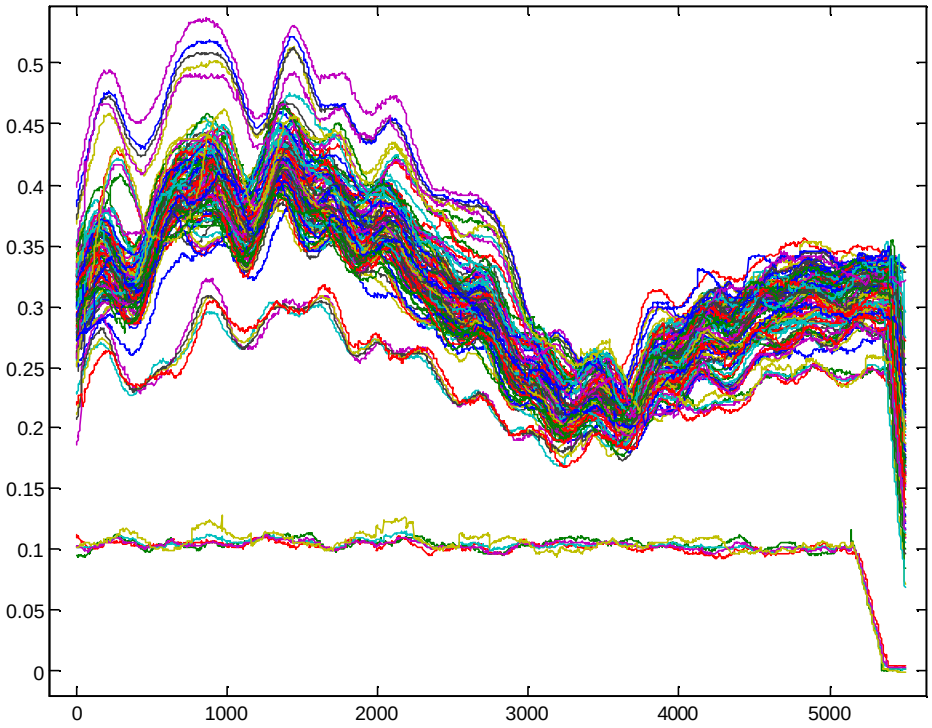


Figura 5.13 – Segmento significativo de fechamento, 5.500 pontos.

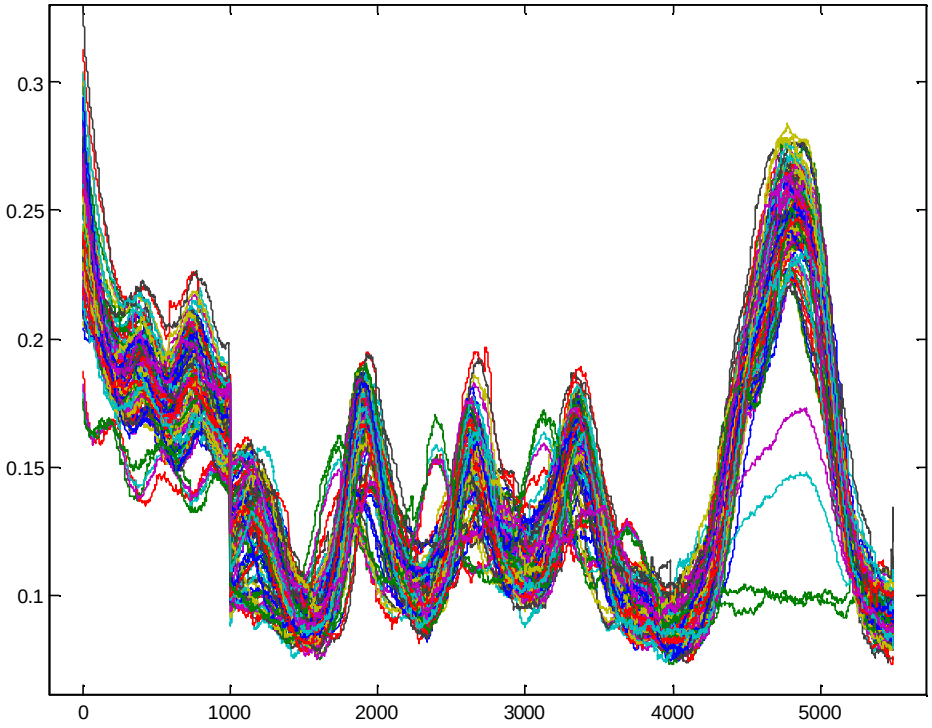


Figura 5.14 – Segmento significativo de abertura, 5.500 pontos.

5.2.5 REDUZINDO OS SEGMENTOS SIGNIFICATIVOS

De acordo com o propósito deste trabalho, o classificador a ser utilizado será uma RNA e, a priori, foi definido que a entrada da rede seria composta de 30 neurônios. Então, devem-se reduzir os segmentos significativos para 30 pontos. Os novos pontos foram obtidos através da média aritmética de 21 pontos a cada intervalo de 185 pontos do segmento significativo.

As figuras a seguir ilustram os resultados:

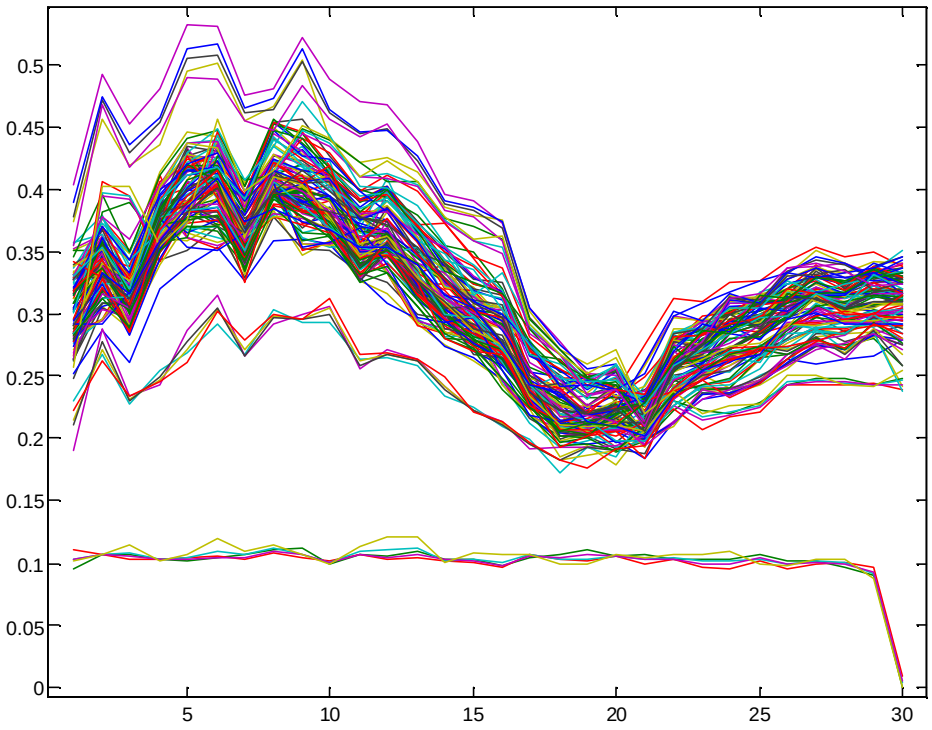


Figura 5.15 – Curvas de fechamento, reduzidas a 30 pontos.

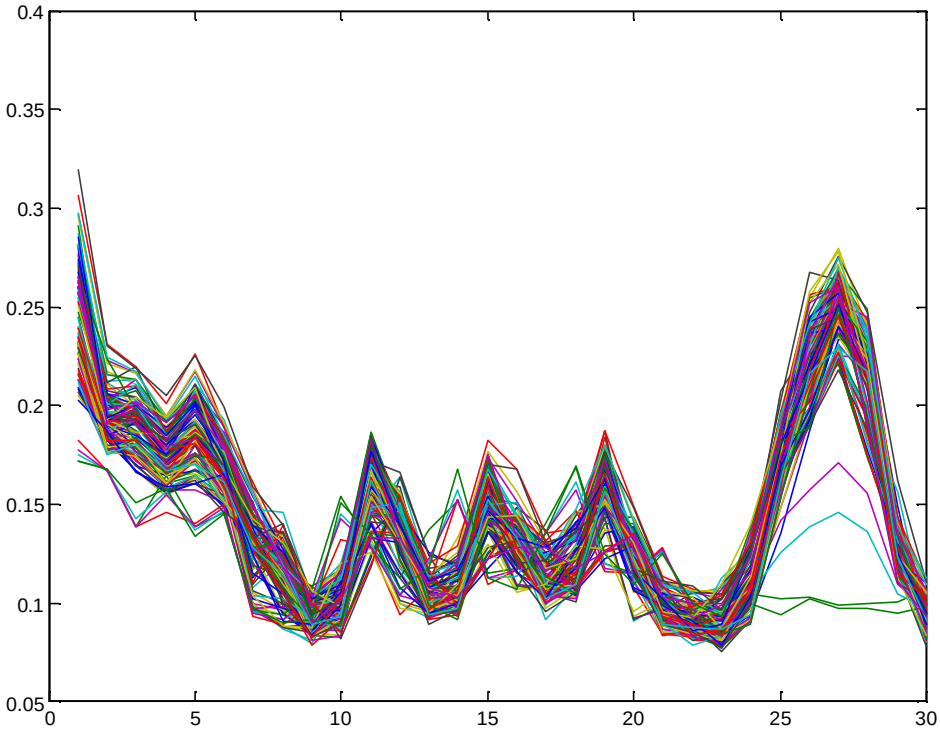


Figura 5.16 – Curvas de abertura, reduzidas a 30 pontos.

5.3 DESENVOLVIMENTO DOS BANCOS DE DADOS

Após o pré-processamento dos dados, o desenvolvimento dos bancos de dados necessários para o treinamento e validação da rede neural será o próximo passo deste estudo.

De acordo com [Haykin, 2001], a respeito da quantidade de curvas necessárias para a formação dos bancos de dados, a base de dados, de treinamento deverá ser aproximadamente 80% do total de dados, e os 20% restantes deverá ser utilizado para validar o modelo. Dessa forma, faz-se necessário aumentar a base de dados inicial.

Conforme a tabela 5.1, excluindo os testes 6 e 7 (que são os testes para funcionamento normal para fechamento e abertura, respectivamente), temos 3 curvas para cada tipo de teste. O objetivo do aumento da base de dados será fornecer duas novas curvas de treinamento.

Para cada teste com três curvas, o banco de dados foi aumentado e separado da seguinte forma:

Curva 1: Curva real, validação;

Curva 2: Curva real, treinamento;

Curva 3: Curva real, treinamento;

Curva 4: Média aritmética da Curva 2 e da Curva 3, treinamento.

Curva 5: Média aritmética das três primeiras curvas, treinamento.

Para os testes de funcionamento normal de abertura e fechamento da chave, que contêm cada uma 27 curvas, foram escolhidas aleatoriamente cinco curvas para serem utilizadas na validação, as 22 restantes foram utilizadas para treinamento.

As figuras abaixo ilustram os bancos de dados de treinamento e de validação, para fechamento e abertura da chave seccionadora, resultantes:

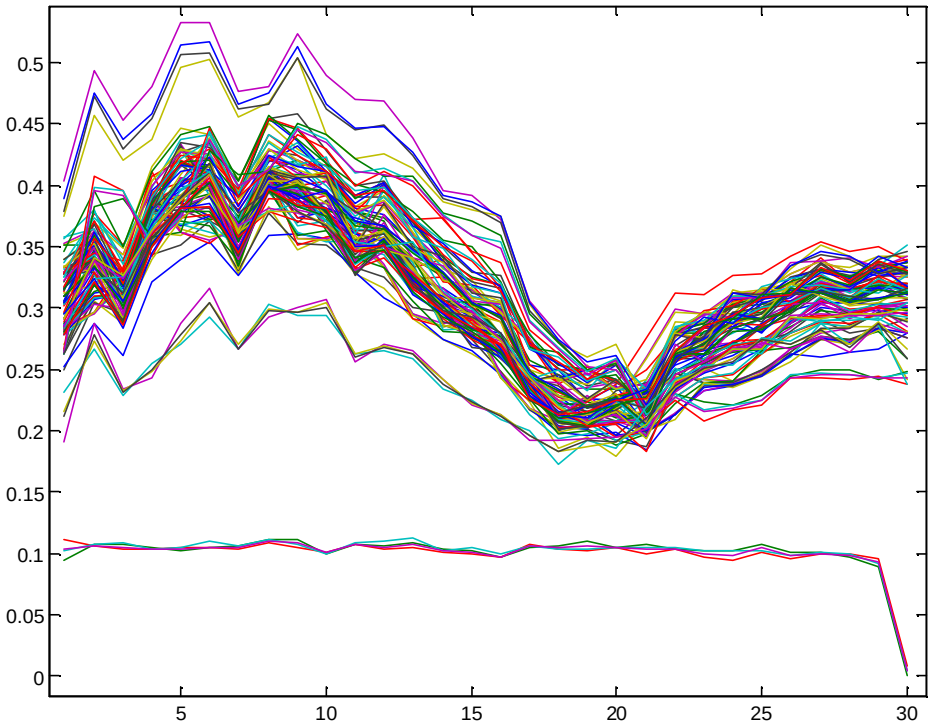


Figura 5.17 – Dados de treinamento, para fechamento da chave.

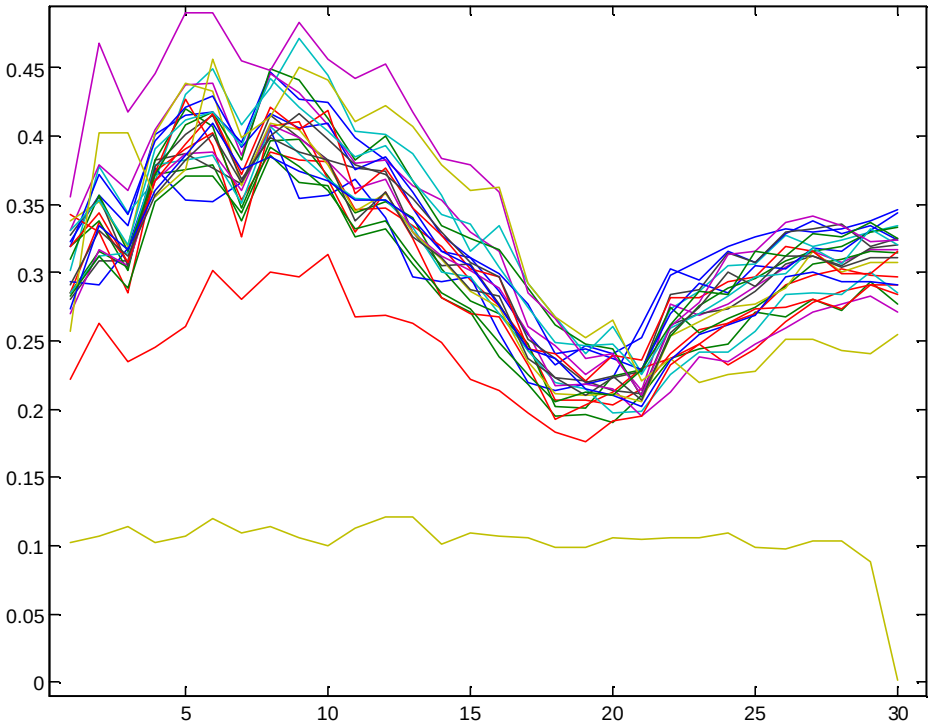


Figura 5.18– Dados de validação para fechamento da chave.

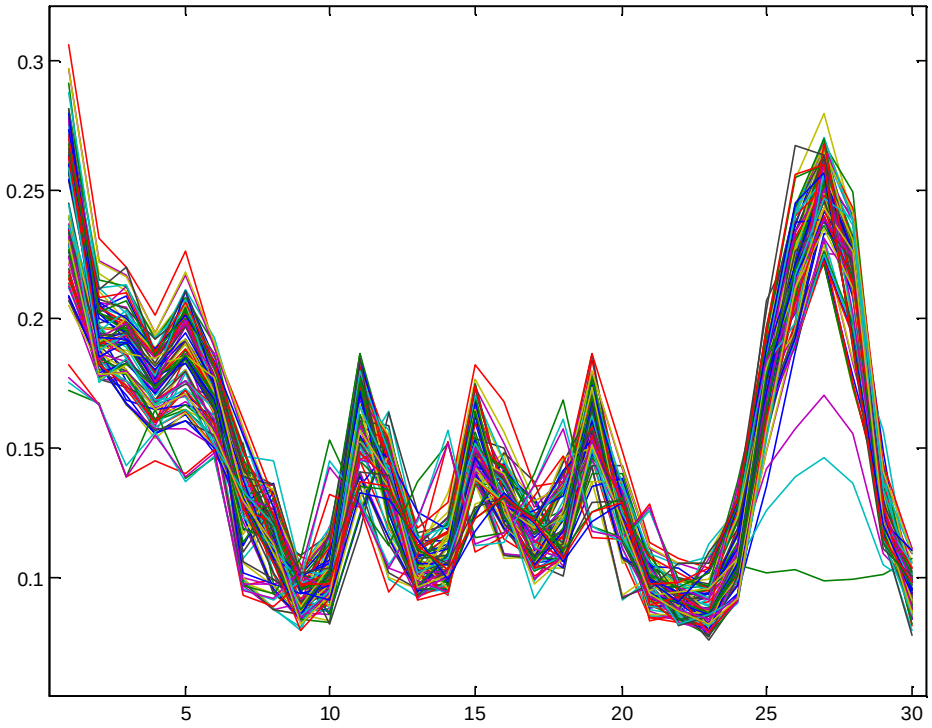


Figura 5.19 – Dados de treinamento para abertura da chave.

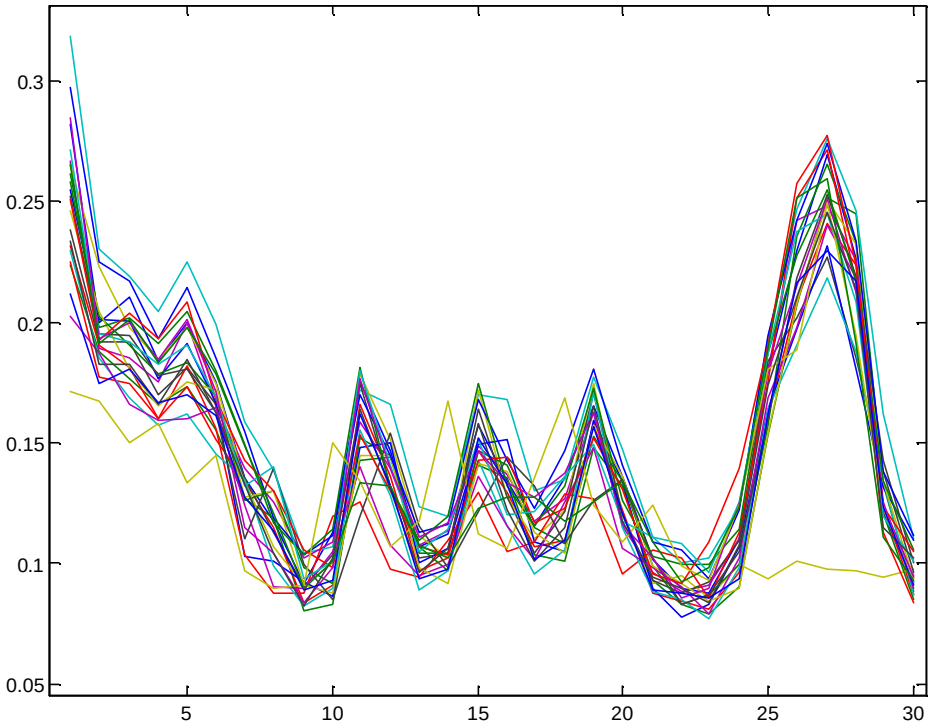


Figura 5.20 – Dados de validação para abertura da chave.

Para o treinamento supervisionado da rede neural, faz-se necessário um banco de dados do tipo *causa-efeito*. Anteriormente foi apresentado o banco de dados de treinamento (*causa*). Seu *efeito* constitui um outro banco de dados, chamado de *target*, que indica para rede qual o resultado esperado dos neurônios de saída para cada sinal de entrada.

Como será necessário classificar 18 tipos de defeito para abertura da rede, mais o funcionamento normal, foi definido que a saída da rede deverá ter 18 neurônios, cada um representativo de um tipo de curva. Ou seja, se a rede identificar que o sinal de entrada é uma curva representativa tipo 1, o neurônio 1 deverá ser ativado enquanto que os outros deverão permanecer não ativados. Se for detectado que o sinal é de funcionamento normal, não deverá ser ativado nenhum neurônio. Assim a base de dados poderá ser implementada considerando -1 o valor de neurônio não ativo, e 1 para neurônio ativo. Esse banco de dados foi denominado de target de abertura.

Da mesma forma, foi desenvolvido o target para o fechamento da chave, desta vez com 19 tipos de curvas a serem classificadas.

5.4 DESENVOLVIMENTO DO SISTEMA CLASSIFICADOR

Para o sistema classificador adotado existem duas estruturas diferentes de RNA's, dependentes do tipo de manobra efetuada pela chave seccionadora. A seguir tem-se o desenvolvimento das redes, o tipo de treinamento e demais particularidades.

5.4.1 A ESTRUTURA DAS REDES

Como descrito anteriormente, será necessário o treinamento de duas redes neurais: uma para fechamento da chave, e outra para a abertura. Ambas as redes deverão ter 30 pontos de entrada.

Para a rede de abertura, a saída da rede deve ter 18 neurônios, um para cada tipo de sinal a ser classificado, ou seja, se o sinal de entrada for representativo de uma curva do tipo 1, o neurônio 1 deverá ser ativado (neurônio 1 = +1), enquanto que os demais deverão permanecer não ativados (demais neurônios = -1). Para uma curva representativa de funcionamento normal, todos os neurônios da saída da rede deverão estar não-ativados (todos em -1), ou seja, a rede não reconheceu padrão de defeitos.

Para a rede de fechamento, a última camada deverá ter 19 neurônios, um para cada tipo de sinal a ser classificado, da mesma forma como a rede de abertura.

Para ambas as redes a camada intermediária, ou camada oculta, será composta de 25 neurônios.

A estrutura das redes pode ser observada na figura abaixo:

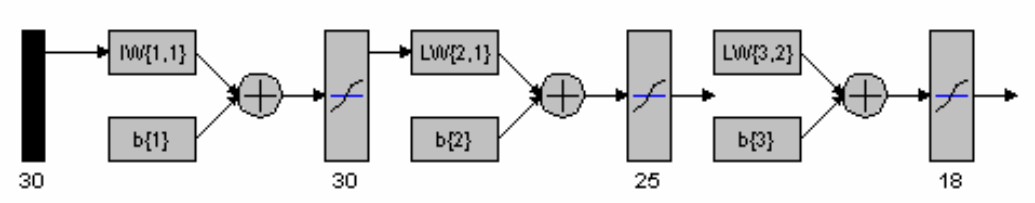


Figura 5.21 – Rede de abertura, 30/25/18.

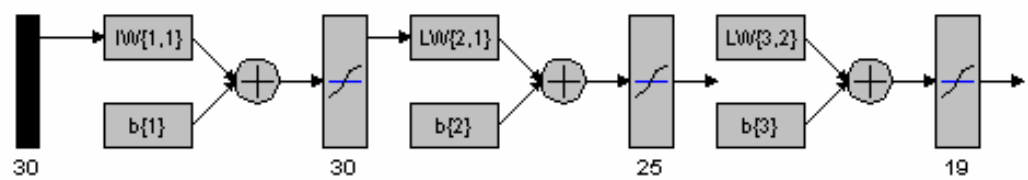


Figura 5.22 – Rede de fechamento, 30/25/19.

Em todas as camadas, a função de ativação utilizada foi a *tansig*, cujo gráfico é ilustrado a seguir:

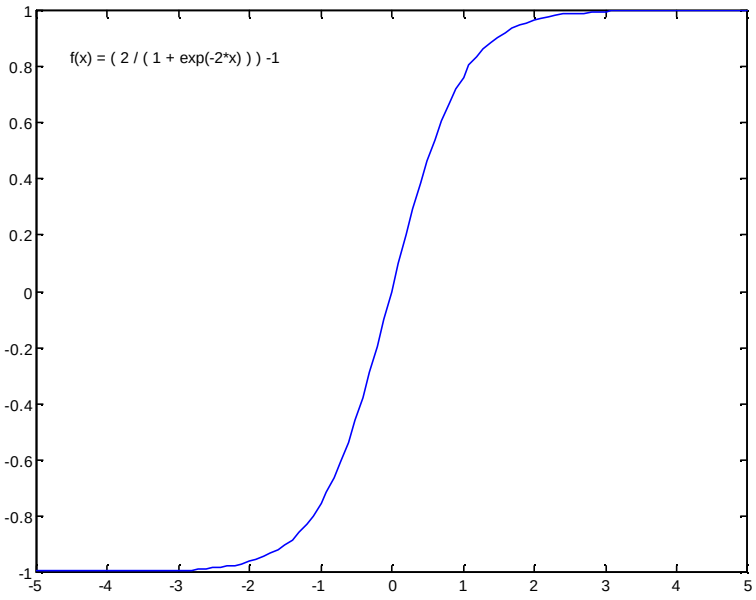


Figura 5.23 – *tansig* como função de ativação em todas as camadas.

5.4.2 TREINAMENTO DAS REDES

O treinamento das redes foi feito utilizando-se o algoritmo *back-propagation*, ou *algoritmo de retro propagação de erro*, no ambiente MATLAB R13.

Para o treinamento de cada rede, os bancos de dados de treinamento e de target foram utilizados para a minimização da função de custo através do algoritmo. Os treinamentos foram feitos a cada 10 épocas, devido aos grandes esforços computacionais requeridos, até atingir erros de aderência médios na casa de 10^{-7}. Em todas as épocas de treinamento utilizou-se a *Bayesian Regularization* para otimizar a capacidade de generalização da rede.

Para o processamento dos treinamentos utilizou-se um computador Pentium 4 2,4 MHz, com 512Mb de RAM, e o tempo de treinamento médio foi de 25 minutos a cada 10 épocas.

Capítulo 6

TESTES E RESULTADOS

Para a validação das redes treinadas anteriormente, comprovando sua capacidade de "aprendizado" dos padrões independentemente dos dados de simulação, deve-se efetuar a fase de teste das redes. Essa fase consiste em apresentar às redes um novo banco de dados para simulação (dados não utilizados nos treinamentos). A eficiência de "aprendizado" dos padrões, pela rede, está diretamente ligada a esta fase. Resultados ruins indicarão que a rede não foi capaz de generalizar e, provavelmente, a rede está especializada apenas em classificar os dados de treinamento, como uma memória, que não contempla o objetivo deste sistema.

Abaixo seguem os resultados obtidos para a simulação das redes com dados excluídos dos treinamentos, conforme mencionado no Capítulo 4 - Formação dos Bancos de Dados.

ABERTURA DA CHAVE

O *target* a ser alcançado pela simulação dos dados de testes de abertura, para a rede treinada, é o seguinte:

Tabela 6.1 – Valores esperados das saídas dos neurônios para o teste da rede de abertura.

		Curvas Reais Utilizadas no Teste																						
		7_1	7_6	7_11	7_16	7_21	27	30	31	32	33	34	35	36	37	44	45	46A	47P1	47P2	47P3	48A	49A	53A
Saída dos Neuronios	1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	2	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	3	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	4	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	5	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	6	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	8	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	10	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1
	11	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
	12	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1
	13	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
	14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1
	15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
	16	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1
	17	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
	18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1

Resultados obtidos através da simulação da rede com os dados de testes:

Tabela 6.2 – Valores obtidos nas saídas dos neurônios, para o teste da rede de abertura.

Saída dos Neurônios	Curvas Reais Utilizadas no Teste																							
	7_1	7_6	7_11	7_16	7_21	27	30	31	32	33	34	35	36	37	44	45	46A	47P1	47P2	47P3	48A	49A	53A
1	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
2	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
3	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	0,8	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
4	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
5	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
6	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
7	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
8	-1,0	-0,9	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
9	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	0,7	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
10	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
11	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
12	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
13	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0
14	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0
15	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0
16	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0
17	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0
18	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0

Uma justificativa plausível, pelo erro cometido ao simular o teste 7_16, é que a curva não é representativa, de fato, do funcionamento normal de abertura da chave. Durante a fase de treinamento, 22 curvas do teste 7 foram apresentadas à rede, os erros médios de aderência da rede ficaram próximos de 10^{-7} e ,na fase de testes, quatro das cinco curvas foram corretamente classificadas. Apenas uma curva, das 27 apresentadas à rede, não foi classificada corretamente. Para garantir as deduções acima, a figura a seguir ilustra as curvas do teste 7 utilizadas na fase de testes.

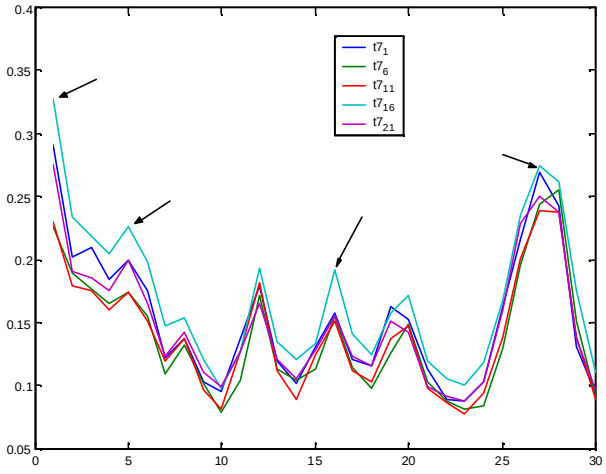


Figura 6.1 – Curvas de teste da rede para o teste 7.

De fato, o teste 7_16 não é representativo e deve ser descartado. Sua exclusão do banco de testes da rede induzirá um novo treinamento pois, de acordo com a *Bayesian Regularization* utilizada durante a fase de treinamento, o banco de dados de testes tem influência no ajuste dos pesos sinápticos da rede.

Após a exclusão do teste 7_16, a rede foi treinada novamente e os novos resultados da simulação dos dados de testes podem ser observados na tabela a seguir:

Tabela 6.3 – Valores obtidos nas saídas dos neurônios para o teste da rede de abertura após o novo treinamento.

		Curvas Reais Utilizadas no Teste																					
		7_1	7_6	7_11	7_21	27	30	31	32	33	34	35	36	37	44	45	46A	47P1	47P2	47P3	48A	49A	53A
Saída dos Neurônios	1	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	2	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	3	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	0,9	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	4	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	5	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	6	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	7	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	8	-1,0	-0,9	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	9	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	0,7	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	10	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	11	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	12	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	13	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	14	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0
	15	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0
	16	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0
	17	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0
	18	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0

De acordo com os resultados obtidos pela simulação do sistema classificador de abertura da chave seccionadora, a rede neural foi validada. Percebe-se, após o novo treinamento, uma ligeira melhora nos resultados obtidos (ver resultados das simulações do teste 31). Embora os resultados dos testes 31 e 37 não serem ideais, representam um elevado grau de proximidade com o padrão.

O sistema classificador foi capaz de classificar corretamente 100% dos dados de testes.

FECHAMENTO DA CHAVE

O *target* a ser alcançado pela simulação dos dados de testes de fechamento, para a rede treinada, é o seguinte:

Tabela 6.4 – Valores esperados das saídas dos neurônios para o teste da rede de fechamento.

Saída dos Neurônios	Curvas Reais Utilizadas: Teste																								
	6_1	6_6	6_11	6_16	6_21	8	11	12	13	14	15	16	17	18	25	26	46	47P1	47P2	47P3	48	49	53	56
1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
4	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
8	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
10	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
11	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1
12	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
13	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1
14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1
16	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
17	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1
18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
19	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1

Resultados obtidos através da simulação da rede com os dados de testes:

Tabela 6.5 – Valores obtidos nas saídas dos neurônios para o teste da rede de fechamento.

	Curvas Reais Utilizadas: Teste																								
	6_1	6_6	6_11	6_16	6_21	8	11	12	13	14	15	16	17	18	25	26	46	47P1	47P2	47P3	48	49	53	56
1	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
2	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
3	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
4	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
5	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
6	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
7	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
8	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
9	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
10	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
11	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-0,9	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
12	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
13	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
14	-1,0	-1,0	-1,0	-0,6	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0	-1,0
15	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-1,0	-1,0
16	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0	-0,9
17	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0	-1,0
18	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0	-1,0
19	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	1,0

De acordo com os resultados obtidos pela simulação do sistema classificador de fechamento da chave seccionadora, a rede neural foi validada. O sistema classificador foi capaz de classificar corretamente 100% dos dados de testes.

Capítulo 7

DESENVOLVIMENTO DO PROGRAMA SIMULADOR

Para a conclusão deste estudo, o programa SIMULADOR foi projetado para automatizar todos os cálculos contidos no pré-processamento e para simular as saídas das redes. O programa foi desenvolvido para auxiliar o usuário e/ou outros programas que possam ser beneficiados pelos resultados obtidos, agrupando em um só software todas as etapas desse estudo. O programa foi implementado em linguagem C, pelo compilador Borland C/C++ - v5.02.

O diagrama de blocos, a seguir, descreve o funcionamento do programa.

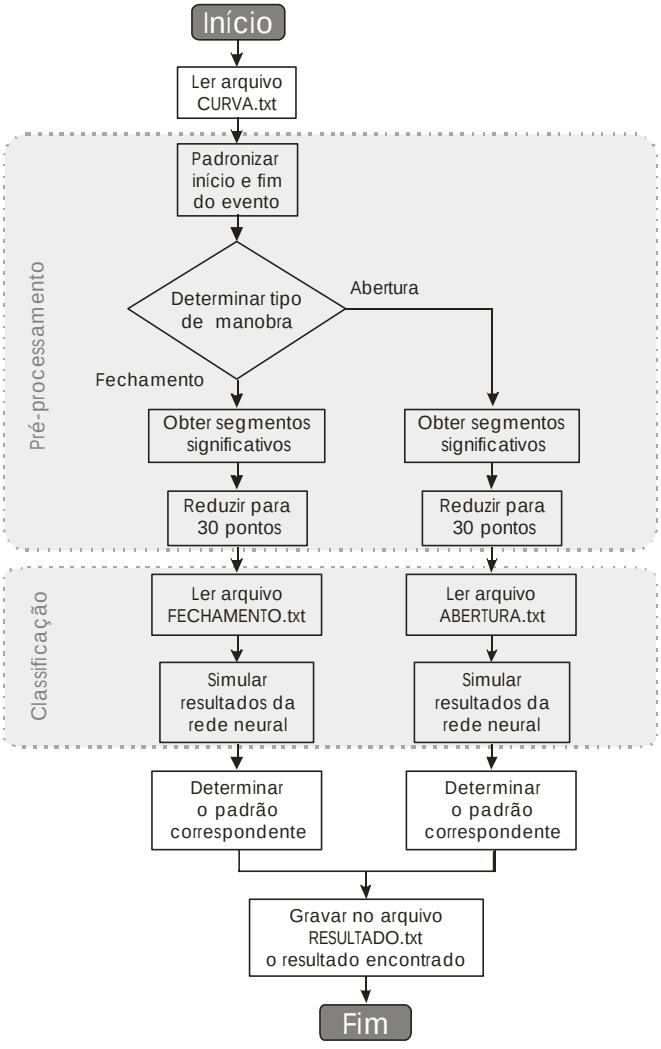


Figura 7.1 – Diagrama de blocos do programa SIMULADOR.

O arquivo “CURVA.txt” contém os dados de torque do motor de acionamento do seccionador, após a filtragem dos dados. Os arquivos “ABERTURA.txt” e “FECHAMENTO.txt” contêm os pesos sinápticos obtidos após o treinamento das redes, e o arquivo “RESULTADO.txt” contém o resultado obtido na classificação.

O usuário poderá acompanhar os resultados obtidos, em cada etapa do programa, através de uma janela de texto.

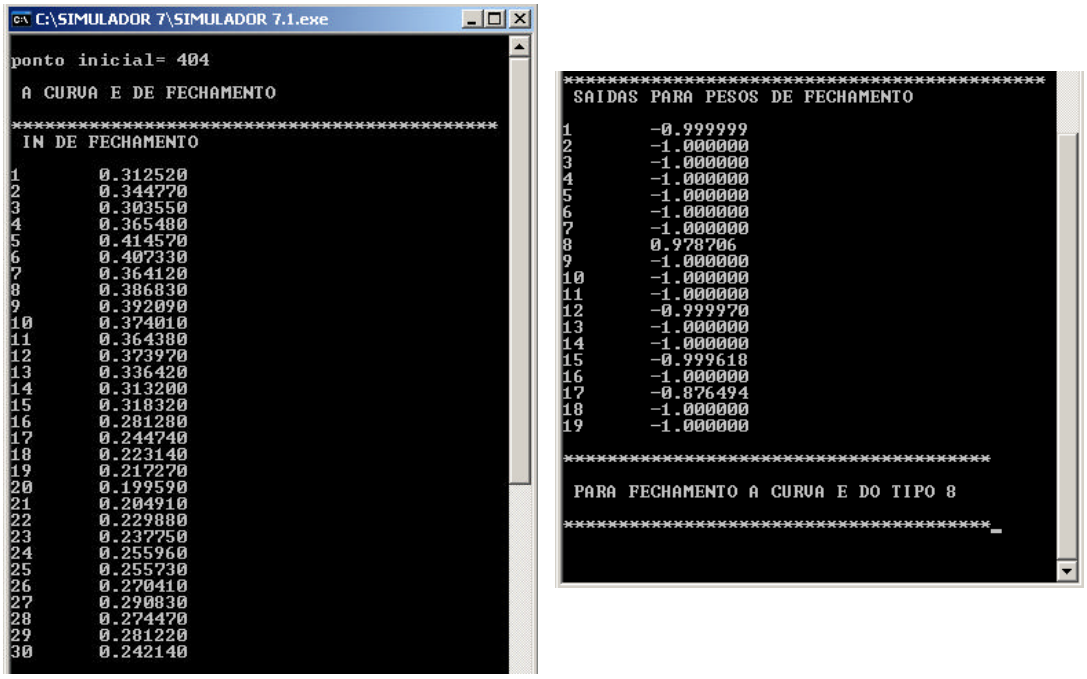


Figura 7.2 – Resultados obtidos pelo programa SIMULADOR.

As linhas de comando do programa SIMULADOR encontram-se no ANEXO deste trabalho.

Capítulo 8

CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi apresentado o projeto e desenvolvimento de uma ferramenta computacional capaz de classificar o funcionamento de uma chave seccionadora de potência, através da curva de torque fornecida pelo motor de acionamento. As técnicas de processamento digital de sinais e os algoritmos de inteligência artificial, empregadas no projeto, determinaram excelentes resultados e comprovaram a eficiência dos classificadores.

As curvas de torque foram obtidas através de diversos ensaios com a chave seccionadora, totalizando 165 curvas entre as manobras de abertura e fechamento, com uma taxa de amostragem de 0,5 ms (milisegundos). Durante o pré-processamento dos sinais as curvas, com 20.000 pontos, foram filtradas, padronizadas, resumidas em segmentos significativos e reduzidas a 30 pontos, respectivamente.

O sistema classificador composto de duas redes neurais, treinadas a partir do algoritmo *backpropagation* e da *Bayesian Regularization*, possuem três camadas e cada neurônio de saída representa um padrão diferente. Em cada saída da rede foi possível medir o grau de similaridade com o respectivo padrão, determinando o tipo de funcionamento correspondente.

O sistema projetado foi capaz de classificar corretamente 100% dos dados de testes apresentados. O protótipo foi capaz de reconhecer os 19 estados de operação propostos, para as manobras de abertura da chave seccionadora, e os 20 estados para as manobras de fechamento.

O programa SIMULADOR proposto executa todos os cálculos da fase de pré-processamento e simula os resultados das redes neurais treinadas, seu uso *off-line* indica de maneira simples ao usuário o estado de funcionamento da chave de potência.

A utilização do programa SIMULADOR em subestações de potência auxiliará o monitoramento das chaves seccionadoras à distância e indicará o estado de operação da chave, diminuindo o tempo de reparo e o desligamento desnecessário da linha.

Para o sistema proposto os dados temporais dos ensaios foram omitidos, tais como: temperatura ambiente, umidade relativa do ar, entre outros.

Para trabalhos futuros planeja-se a implantação em campo do programa SIMULADOR, a fim de ratificar os resultados obtidos e testar a robustez do sistema.

Referências Bibliográficas

[Aguirre, 2000] L. A. Aguirre - *"Introdução à Identificação de Sistemas: Técnicas Lineares e Não Lineares Aplicadas a Sistemas Reais"*, Editora UFMG, 2000.

[Américo, 1999] E. Américo, T. Huarsaya - *"Aprendizado Bayesiano para Redes Neurais"*, Tese de Mestrado, DEE-PUC-RIO, 1999.

[Bow, 1984] S. T. Bow - *"Pattern Recognition"* – Marcel Dekker, Inc., 1984.

[Demuth, 2002] H. Demuth, M. Beale - *"Neural Network Toolbox User's Guide – for use with MATLAB"*, Version 4 - MathWorks Inc., 2002.

[Falk, 2002] T. H. Falk, H. M. Oliveira, R. F. G. Távora - *"Decomposição de Wavelets sobre Corpos Finitos"*, Revista da Sociedade Brasileira de Telecomunicações, volume 17, número 1, 2002.

[Farag, 1998] W.A. Farag, V.H. Quintana & G. Lambert Torres - *"A Genetic-Based Neuro-Fuzzy Approach for Modeling and Control of Dynamical Systems"*, IEEE Trans. on Neural Networks, Vol. 9, No. 5, pp. 756-767, 1998.

[Field, 2001] M. W. Field, W. K. Leung - *"A Complexity Analysis of The Back Propagation Algorithm"*, Publicado em *"Problems in Applied Mathematics and Computational Intelligence"*, World Scientific and Engineering Socity Press, 2001.

[Haykin, 2001] S. Haykin – *"Neural Networks: a Comprehensive Foundation"*, 2ed, Prentice-Hall, 1999. (Edição em Português: *"Redes Neurais - Princípios e Prática"*, Bookman, 2001).

[Hertz, 1991] J. Hertz, A. Krogh, R. G. Palmer - *"Introduction to the Theory of Neural Computation"*, Addison-Wesley, 1991.

[Holland, 1991] J. H. Holland - *"Adaptation in Natural and Artificial Systems"*, The MIT Press, 1992.

[Insfran, 1999] A.H.S. Insfran, A.P.A. Silva, G. Lambert Torres - *"Alarm Processing based on Associative Neural Memories with Explanation Capability"*, International Journal on Engineering Intelligent Systems, Vol. 7, No.2, pp. 109-115, 1999.

[Liberty, 2000] J. Liberty - *"C++"*, Editora Berkeley, 2000.

[Looney, 1997] C.G. Looney - *"Pattern Recognition Using Neural Networks: Theory and Algorithms for Enginneers and Scientists"* - Oxford University Press, 1997.

[Medsker, 1995] L.R. Medsker - *"Hybrid Intelligent Systems"*, Kluwer Academic Publishers, 1995.

[Minsky, 1988] M. L. Minsky, S. A. Papert - *"Perceptrons: An Introduction to Computational Geometry"*, 3º edition, The MIT Press, 1988.

[Misiti, 2002] M. Misiti, Y Misiti, G. Oppenheim, J. M. Poggi - *"Wavelets Toolbox User's Guide – for use with MATLAB"*, Version 2 - MathWorks Inc., 2002.

[Niebur, 1996] D. Niebur, G. Lambert Torres - *"Artificial Neural Networks for Power Systems: State of The Art"*, Publicado em: *Neural Networks Applications in Power Systems*, por T.S. Dillon & D. Niebur, CRL Pub., UK, pp. 37-110, 1996.

[Sheridan, 1974] T. B. Sheridan, W. R. Ferrel - *"Man-Machine Systems Information, Control and Decision Models of Human Performance"*, The MIT Press, 1974.

[Rezaeia, 2001] G. R. Rezaeia, R. A. Borzooeia, M. R. Molaeib, M. M. Zahedib - *"Classification of Generalized Groups of Order 2 and 3"*, Publicado em *"Problems in Applied Mathematics and Computational Intelligence"*, World Scientific and Engineering Socity Press, 2001.

[Silva, 2000] A. V. Silva, J. Eyng - *"Wavelets e Wavelets Packets"* - *Seminário de Visão Computacional – CPGCC*, 2000.
Disponível em: http://www.inf.ufsc.br/~visao/2000/Wavelets/#2
Acesso em: 15 de setembro de 2004.

[Tenenbaum, 2000] A. Tenenbaum - *"Estruturas de Dados Usando C"*, Makron Books, 2000.

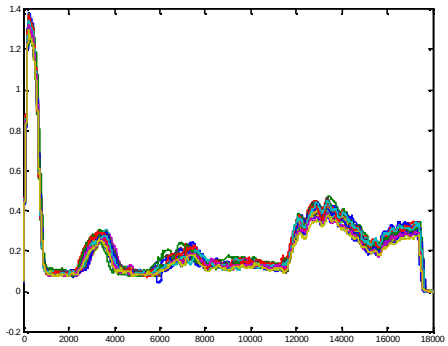
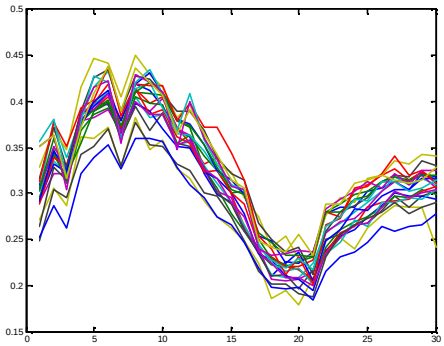
Anexo 1

ESTUDO DOS PADRÕES

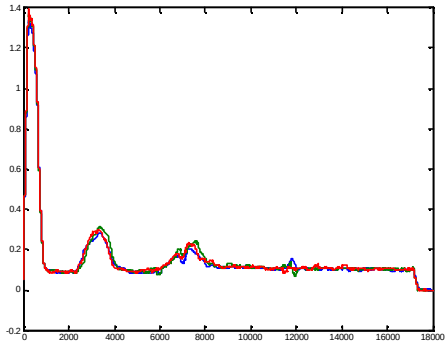
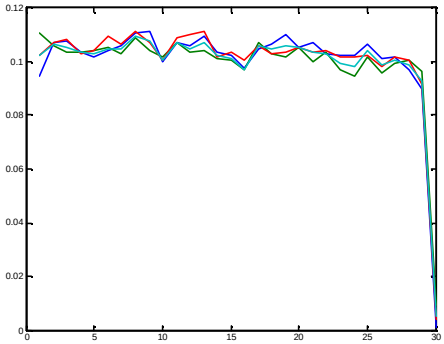
Esta seção apresenta um comparativo entre as curvas obtidas para cada teste e sua respectiva representação para o classificador. Esta comparação tem por objetivo ilustrar o comportamento dos padrões após o pré-processamento dos dados.

1. FECHAMENTO

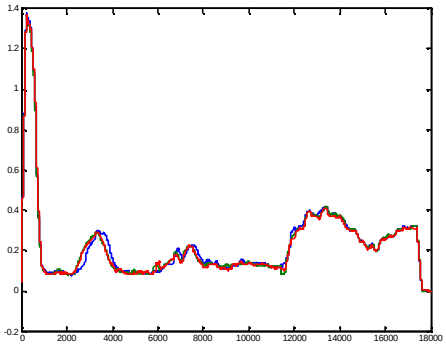
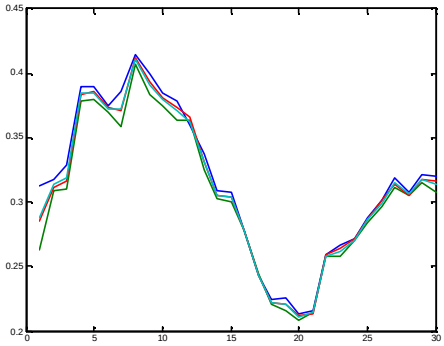
- **TESTE 6** – OPERAÇÃO DE FECHAMENTO NORMAL

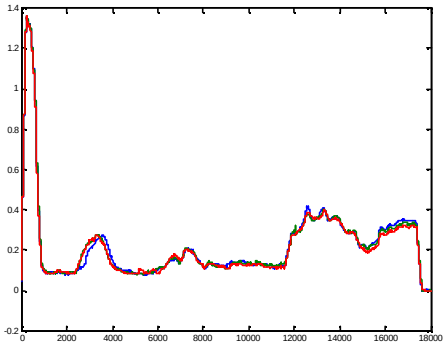
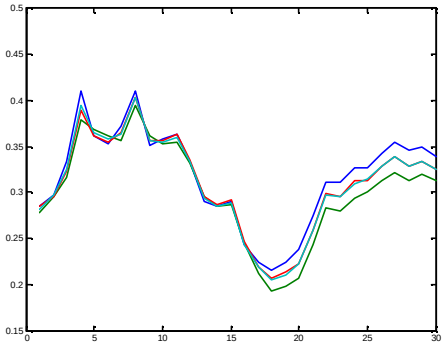
- **TESTE 8**

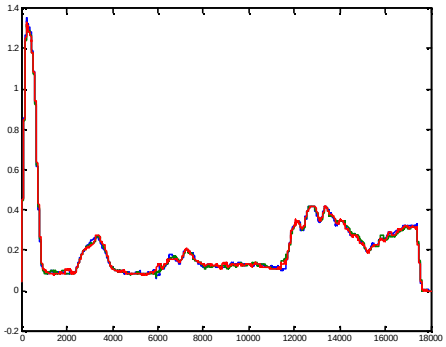
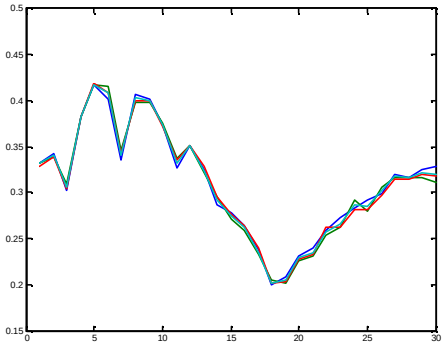
- **TESTE 11**

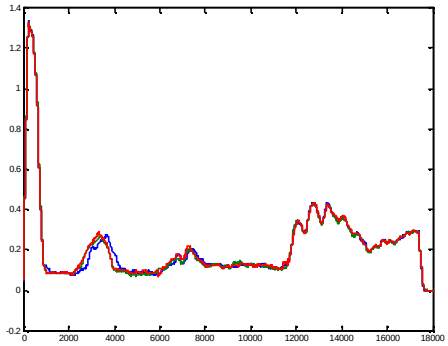
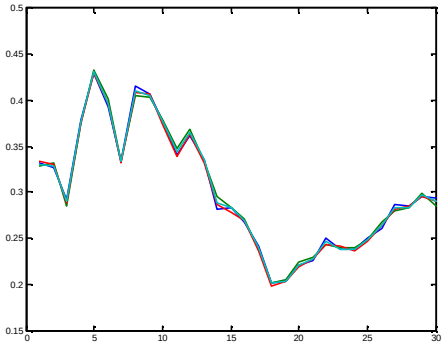
- **TESTE 12**

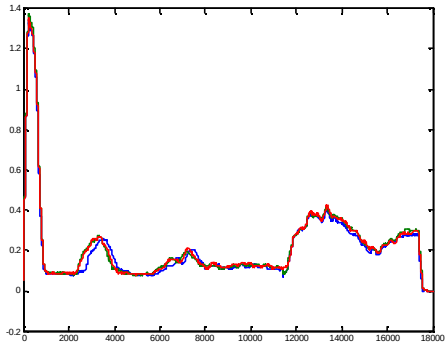
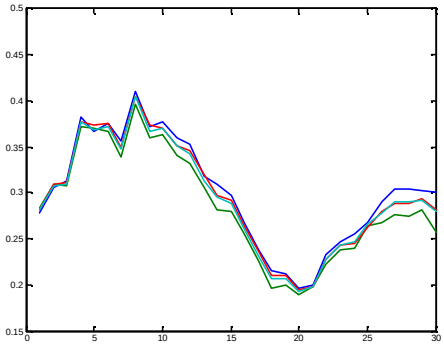
- **TESTE 13**

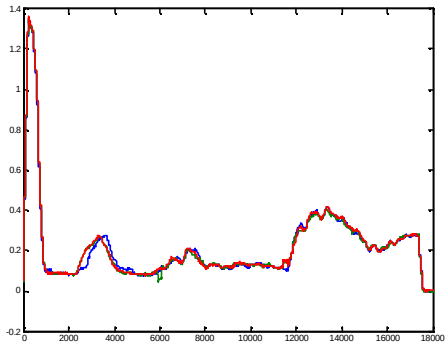
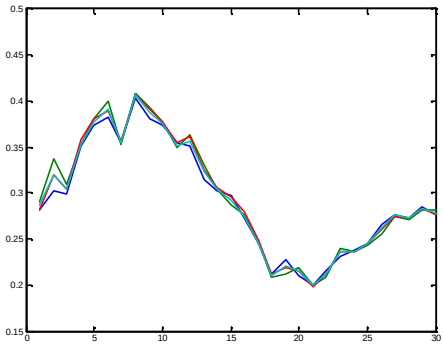
- **TESTE 14**

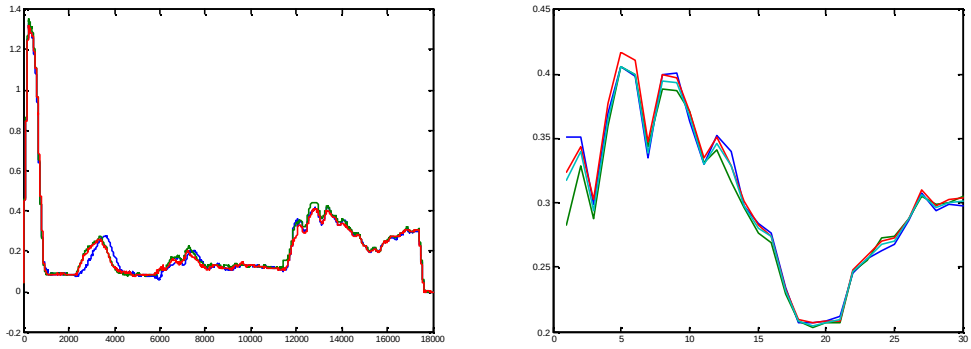
- **TESTE 15**

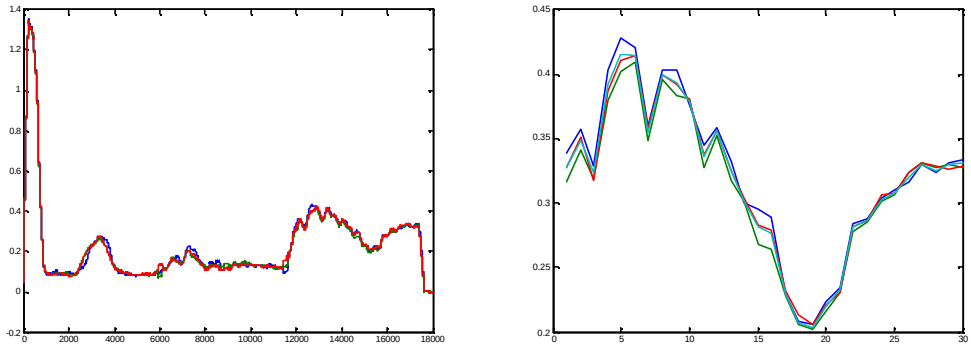
- **TESTE 16**

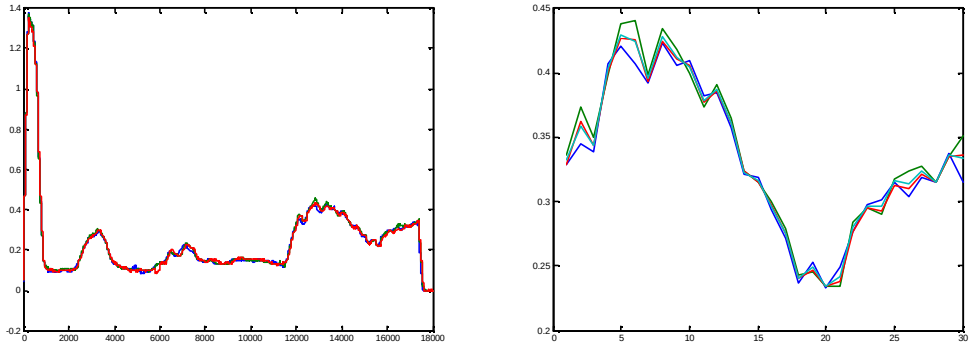
- **TESTE 17**



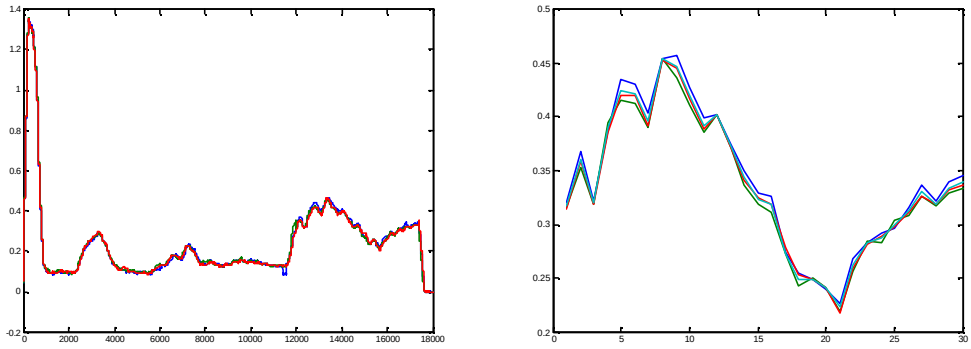
- **TESTE 18**



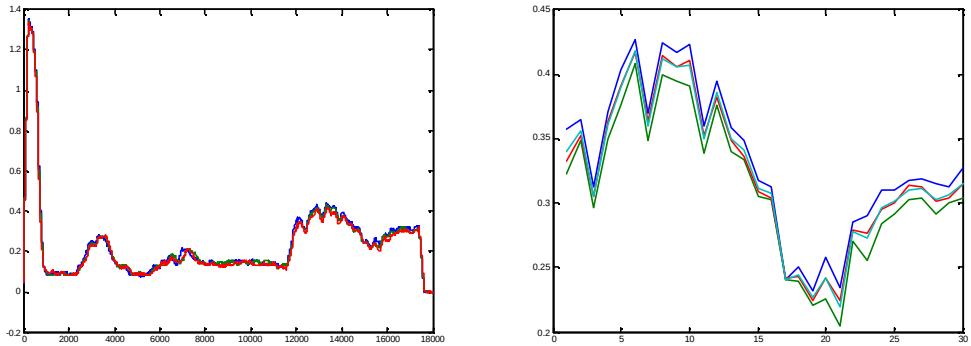
- **TESTE 25**



- **TESTE 26**

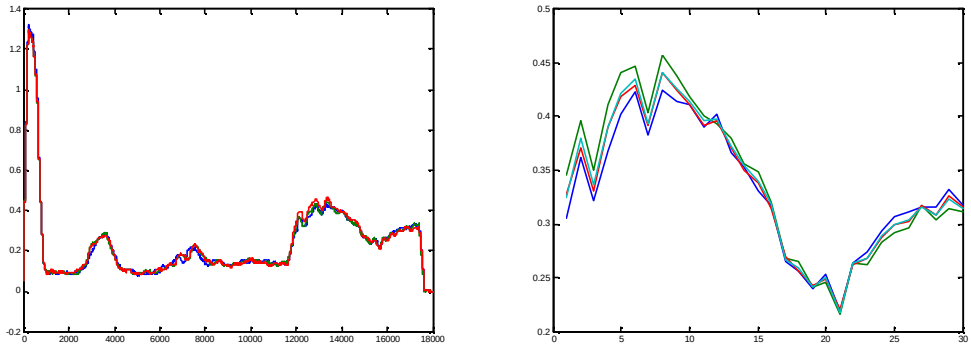


- **TESTE 46 – FECHAMENTO**

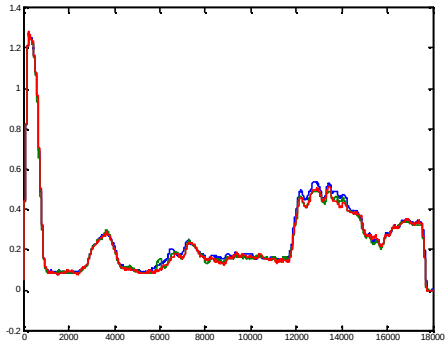
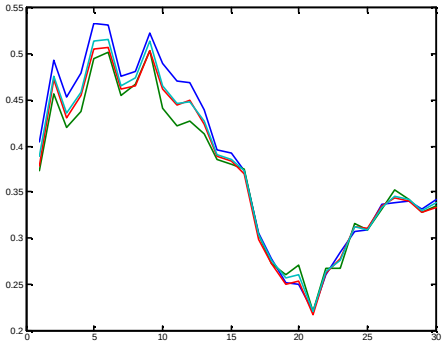


- **TESTE 47**

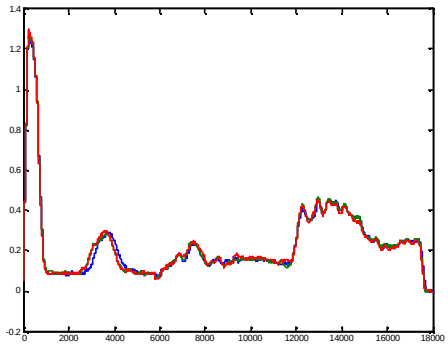
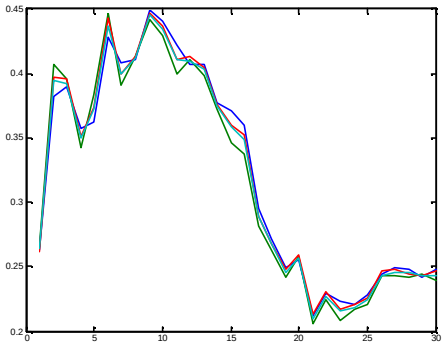
✓ **PROCEDIMENTO 1 – FECHAMENTO**



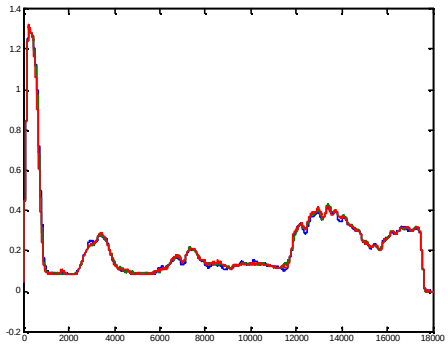
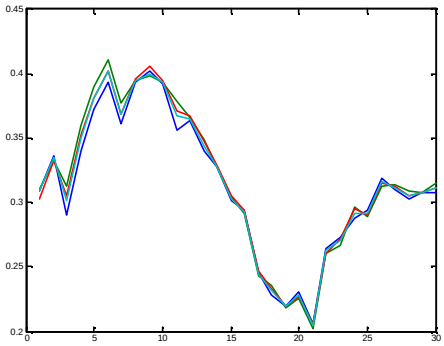
✓ **PROCEDIMENTO 2 – FECHAMENTO**

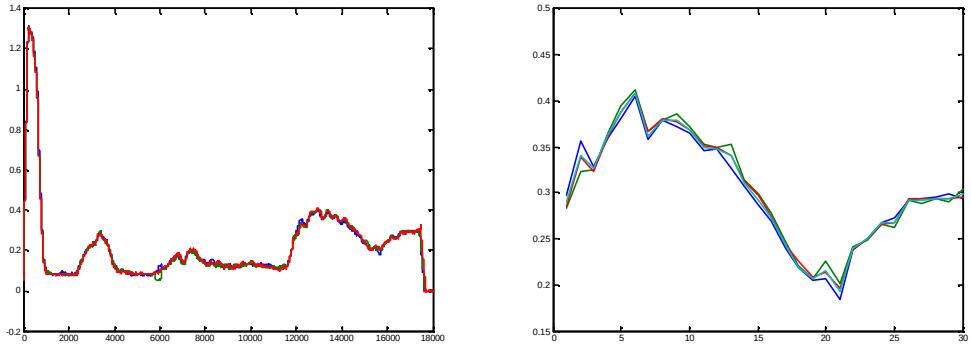
✓ **PROCEDIMENTO 3 – FECHAMENTO**

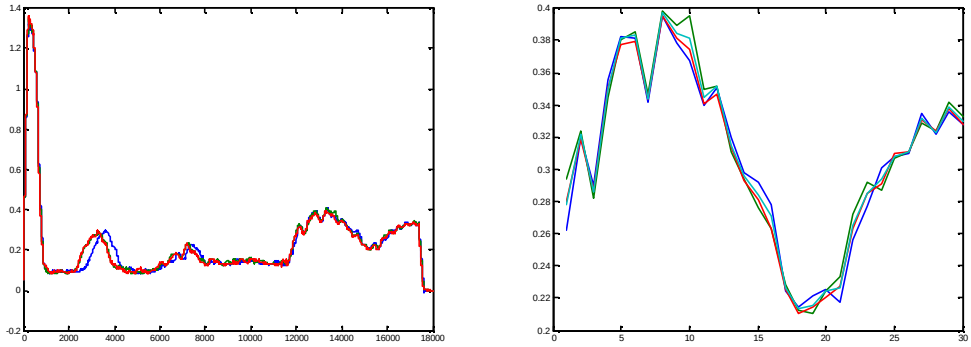
- **TESTE 48 – FECHAMENTO**

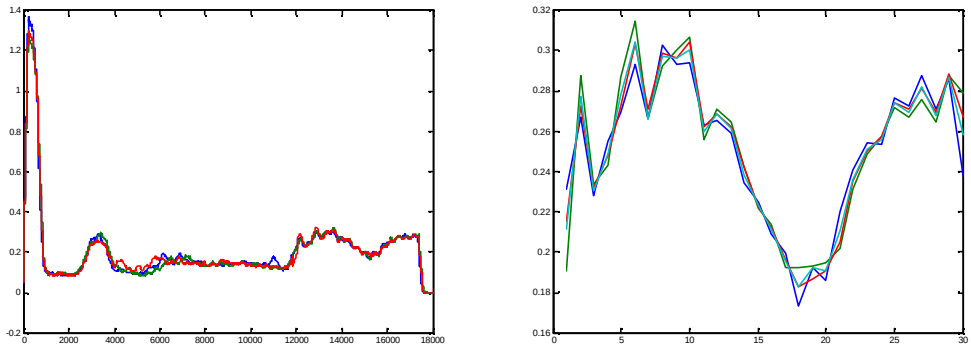
- **TESTE 49 – FECHAMENTO**



- **TESTE 53 - FECHAMENTO**

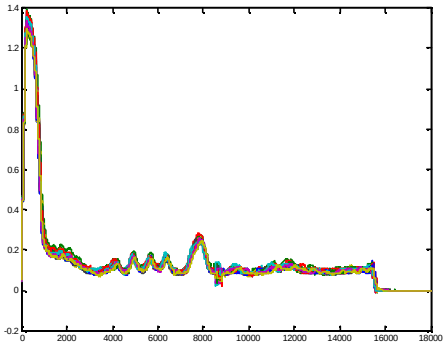
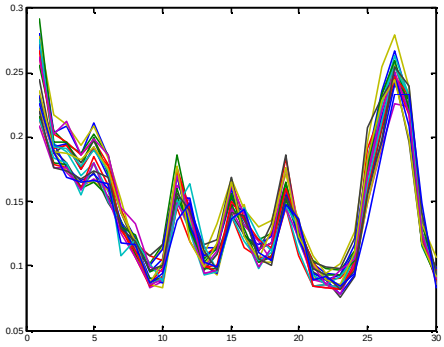


- **TESTE 56 – FECHAMENTO**

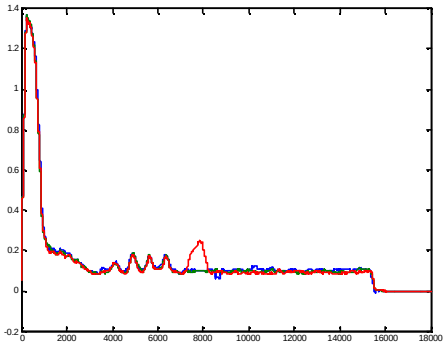
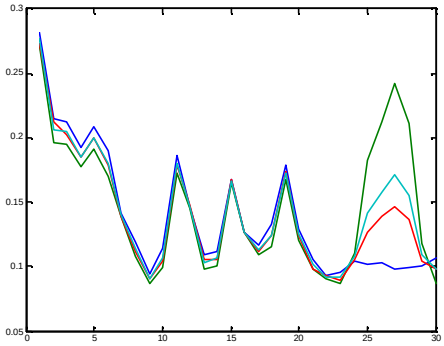


2. ABERTURA

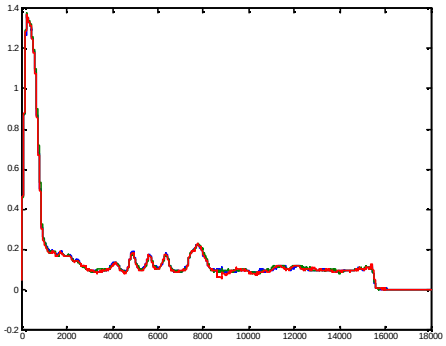
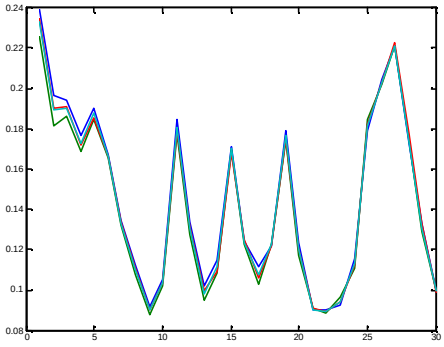
- **TESTE 7** – OPERAÇÃO DE ABERTURA NORMAL

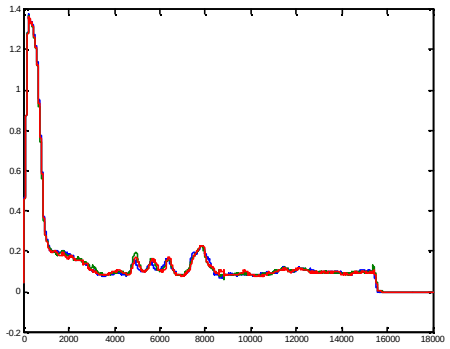
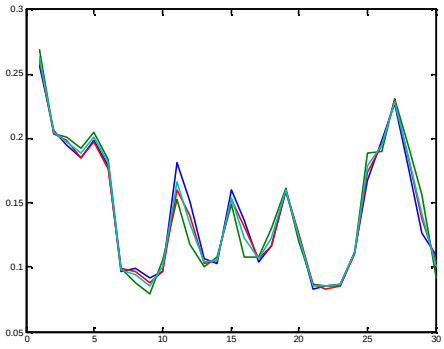
- **TESTE 27**

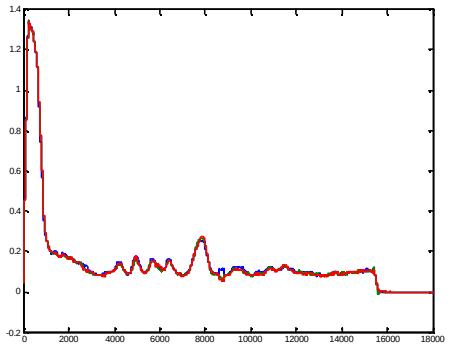
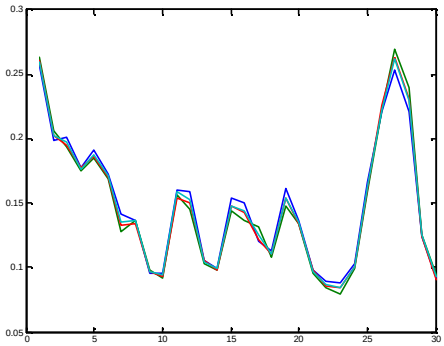
- **TESTE 30**

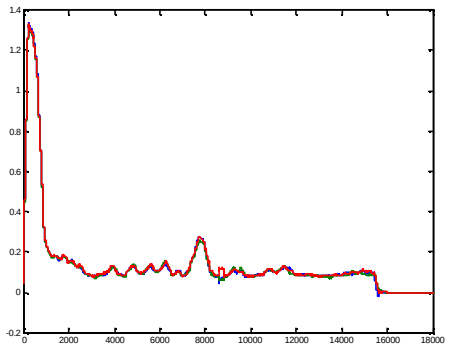
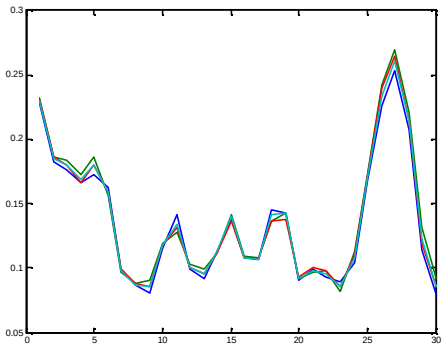
- **TESTE 31**

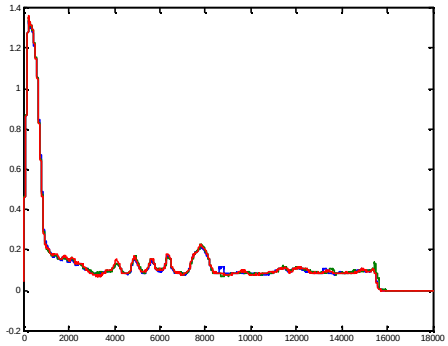
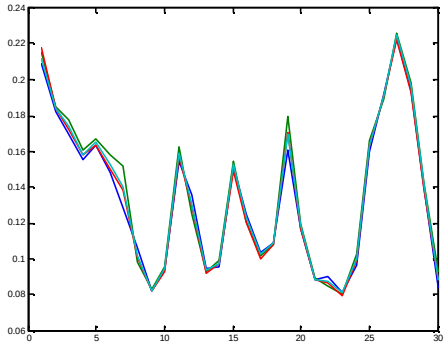
- **TESTE 32**

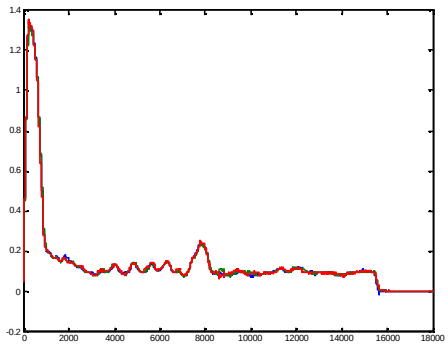
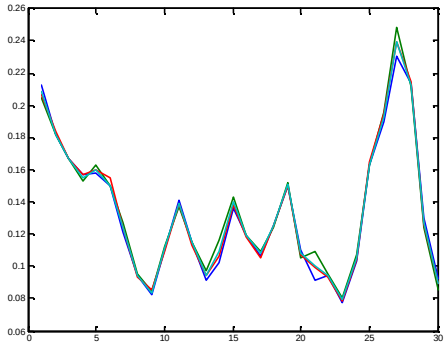
- **TESTE 33**

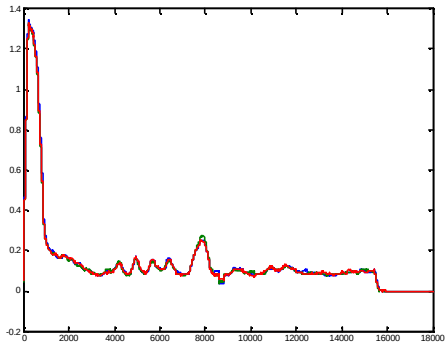
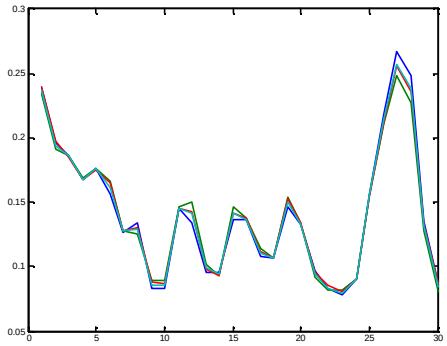
- **TESTE 34**

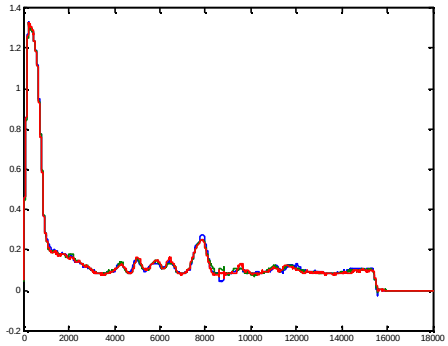
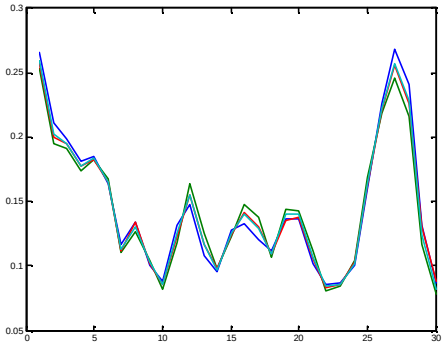
- **TESTE 35**

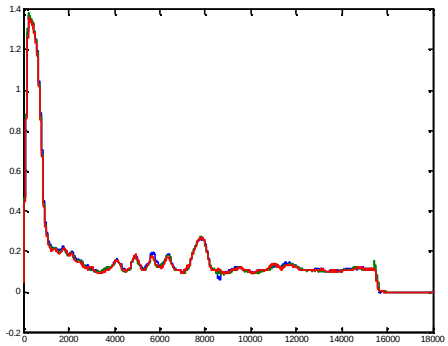
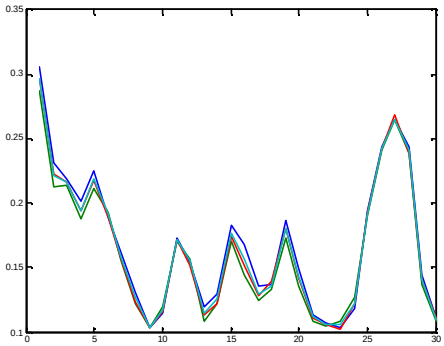
- **TESTE 36**

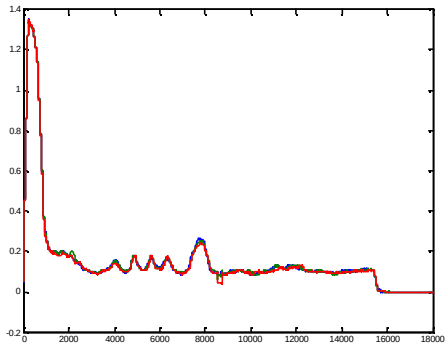
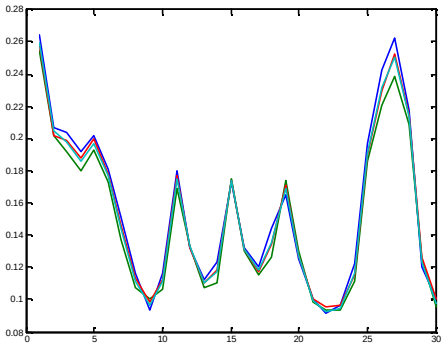
- **TESTE 37**

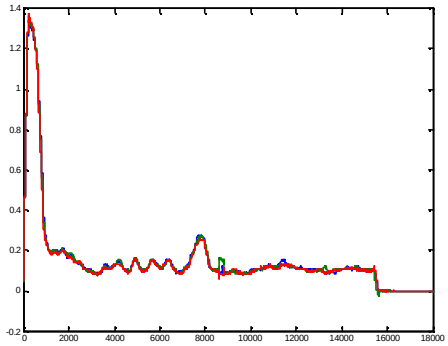
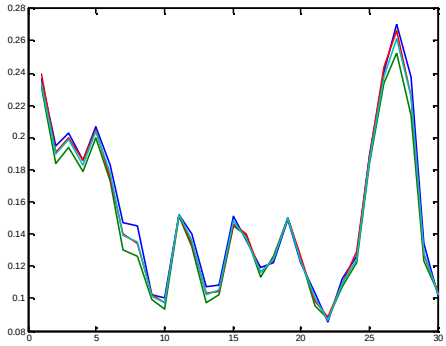
- **TESTE 44**

- **TESTE 45**

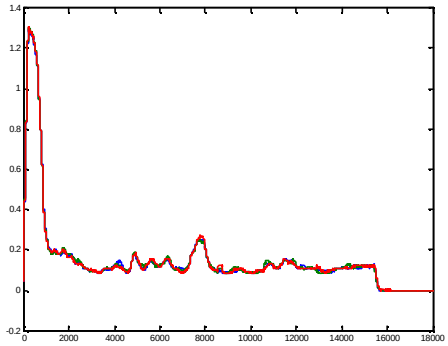
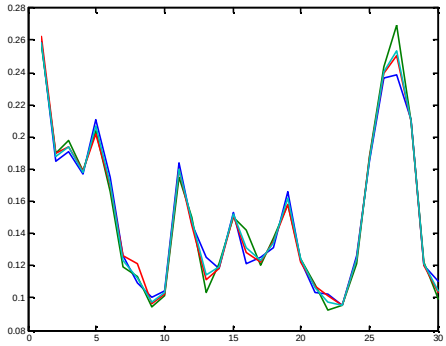



- **TESTE 46 – ABERTURA**

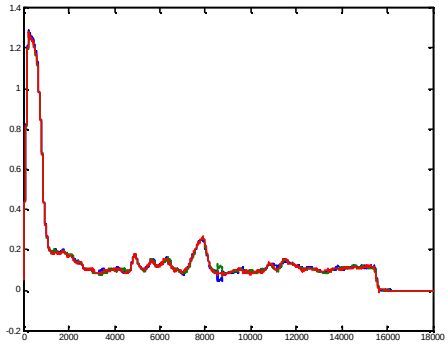
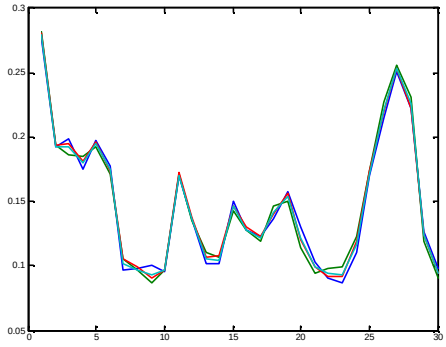



- **TESTE 47**

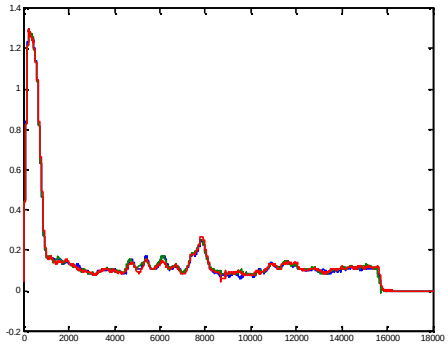
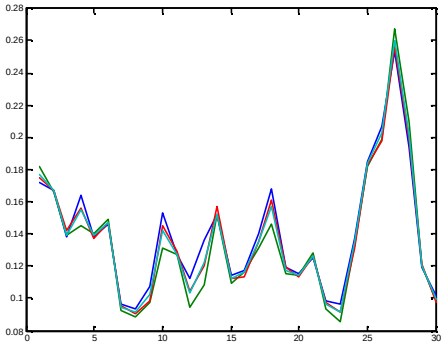
✓ **PROCEDIMENTO 1 – ABERTURA**

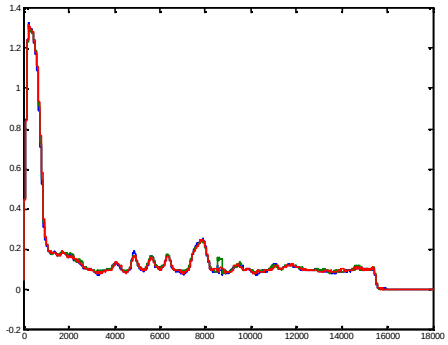
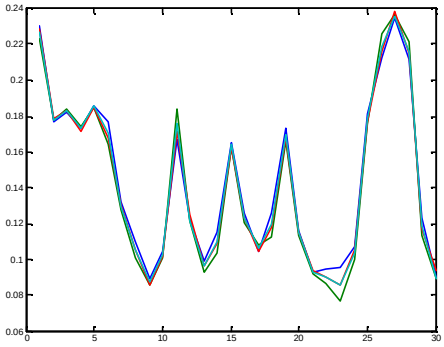
✓ **PROCEDIMENTO 2 – ABERTURA**

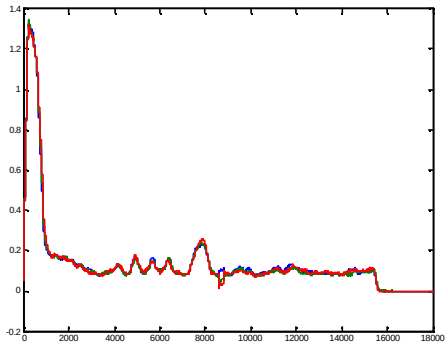
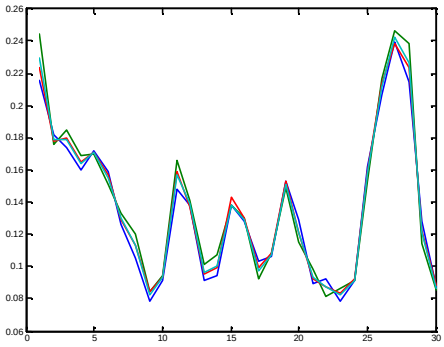
✓ **PROCEDIMENTO 3 – ABERTURA**

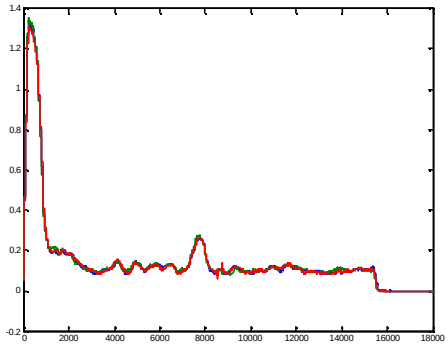
- **TESTE 48 – ABERTURA**

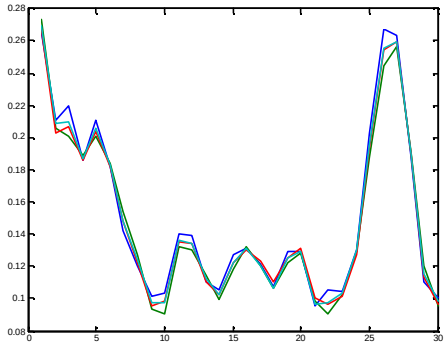



- **TESTE 49 – ABERTURA**

- **TESTE 53 – ABERTURA**





Anexo 2

LINHAS DE COMANDO DO PROGRAMA SIMULADOR

```
/*
                                PROGRAMA SIMULADOR v1
                          ESTE PROGRAMA E PARTE DA DISSERTACAO:

    "SISTEMA DE DETECÇÃO DE FALHAS DE MANOBRAS EM SECCIONADORES DE ALTA
    TENSÃO BASEADO EM PROCESSAMENTO DIGITAL DE SINAIS E RNA"

                              MESTRANDO: ANDRE COLEN CARRASCO
                              ORIENTADOR: GERMANO LAMBERT TORRES

                          UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
                                FEVEREIRO DE 2005
*/
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>

void main (void);
void ler_curva(float*,int);
void padroniza(float[21000],float*,int*);
void reduz(float*,float*,int);
void ler_pesos(int[4],float[30][30],float[25][30],float[20][25],float*,float*,float*,int);
void calcula(int[4],float[30],float[30][30],float[25][30],float[20][25],float[30],float[25],float[20],
double*);
void resultado(double[20],int,int);

/*****************************************************************************/
void main (void)
{
  float curva[21000],cp[18000];
  float in_a[30],in_f[30];
  float iw11[30][30],iw21[25][30],iw32[20][25],b1[30],b2[25],b3[20];
  double saida_a[20],saida_f[20];
  int i,estrutura[4],aber,fech,tipo;
```

```
  clrscr();
  ler_curva(curva,1);
  padroniza(curva,cp,&tipo);

  if (tipo == 1){
        printf("\n\n A CURVA E DE ABERTURA");
        reduz(cp,in_a,1);
        ler_pesos(estrutura,iw11,iw21,iw32,b1,b2,b3,1);
        calcula(estrutura,in_a,iw11,iw21,iw32,b1,b2,b3,saida_a);
        aber=estrutura[3];
        printf("\n\n*********************************************\n SAIDAS PARA PESOS DE
ABERTURA\n");

        for(i=0;i<aber;i++){
              printf("\n%d\t%lf",i+1,saida_a[i]);
        }
        resultado(saida_a,aber,tipo);
        getch();

  }else{
        printf("\n\n A CURVA E DE FECHAMENTO");
        reduz(cp,in_f,2);
        ler_pesos(estrutura,iw11,iw21,iw32,b1,b2,b3,2);
        calcula(estrutura,in_f,iw11,iw21,iw32,b1,b2,b3,saida_f);
        fech=estrutura[3];
        printf("\n\n*********************************************\n SAIDAS PARA PESOS DE
FECHAMENTO\n");

        for(i=0;i<fech;i++){
              printf("\n%d\t%lf",i+1,saida_f[i]);
        }
        resultado(saida_f,fech,tipo);
        getch();
  }
}

/*****************************************************************************/
void ler_curva(float curva[21000],int tipo)
{
  FILE *fp;
```

```
  int i=0;

  if (tipo==1){

        fp=fopen("\\CURVA.TXT","rt");

        if (!fp){
              printf("\n erro 01\n\n  O arquivo CURVA.TXT nao existe,\nou nao esta na raiz.");
                    getch();
                    exit(0);
        }
  }
  if (tipo==2){

        fp=fopen("\\NORMAL_A.TXT","rt");

        if (!fp){
              printf("\n erro 02\n\n  O arquivo NORMAL_A.TXT nao existe,\nou nao esta na raiz.");
              getch();
              exit(0);
              }
  }
  if (tipo==3){

        fp=fopen("\\NORMAL_F.TXT","rt");

        if (!fp){
              printf("\n erro 03\n\n  O arquivo NORMAL_F.TXT nao existe,\nou nao esta na raiz.");
              getch();
              exit(0);
        }
  }

 fscanf(fp,"%f",&curva[i]);

while(!feof(fp)){
        i++;
        fscanf(fp,"%f",&curva[i]);
 }
fclose(fp);
}
```

```
/*****************************************************************************/
void padroniza(float curva[21000],float cp[18000],int *tipo)
{
  int i=1,j,ponto_inicial=0;
  float a,media=0;

  for (j=0;j<5000;j++){
        media=media+curva[j];
  }
  //media=media/5000;

  a=0.05;

  while((ponto_inicial==0)&&(i<=4000)) {

        if( ( fabs(curva[i-1]) < a )&&( fabs(curva[i]) >= a )&&( fabs(curva[i+1]) >= a ) ){

              if ( fabs( fabs(curva[i-1]) - a ) >= fabs( fabs(curva[i]) - a ) ){

              ponto_inicial=i;

               }else{
                     ponto_inicial=i-1;
               }
      }
      i++;
  }

  printf("\nponto inicial= %d",ponto_inicial);

  if (i>=3900){
        printf("\n\n erro 04\n\nO PROGRAMA NAO FOI CAPAZ DE PADRONIZAR\nO INICIO DA
CURVA");

        getch();
        exit(0);
   }

  if ( media <= 0 ){
```

```
    *tipo=1;
    for (j=0;j<18000;j++){
            cp[j]=-(curva[ponto_inicial+j]);
    }

  }else{

   *tipo=2;
    for (j=0;j<18000;j++){
            cp[j]=curva[ponto_inicial+j];
    }
  }
}

/*****************************************************************************/
void reduz(float cp[18000],float in[30],int a)
{
  int i=0,j=0,m=0;
  float soma=0;

  if (a==1){
        printf("\n\n*********************************************\n IN DE ABERTURA\n");

        while (j<6){
                soma=0;
                for(m=-10;m<11;m++){
                        soma=soma+cp[i+1000+m];
                }

                in[j]=soma/21;

                i=i+185;
                j++;
        }
         i=0;
         while (j<30){

                soma=0;
                for(m=-10;m<11;m++){
                        soma=soma+cp[i+4000+m];
                }
```

```
            in[j]=soma/21;

            i=i+185;
            j++;
            }
  }
  i=0;
  j=0;

  if (a==2){
            printf("\n\n*****************************************\n IN DE
FECHAMENTO\n");

            while (j<30){
                        soma=0;
                        for(m=-10;m<11;m++){
                                    soma=soma+cp[i+12000+m];
                        }
            in[j]=soma/21;
            i=i+189;
            j++;
            }
  }
 for (j=0;j<30;j++){
 printf("\n%d \t%f",j+1,in[j]);
}
}

/*************************************************************************/
void ler_pesos(int estrutura[4],float iw11[30][30],float iw21[30][30],float iw32[20][25],float b1[30],float
b2[25],float b3[20],int tipo)
{
  FILE *fp;
  int linha,coluna,entradas,c1,c2,c3;

  if (tipo == 1){

            fp=fopen("\\PESOS_A.TXT","rt");

            if (!fp){
                        printf("\n O arquivo PESOS_A nao existe, ou não está na raiz.");
```

```
                getch();
                exit(0);
            }
    }

    if (tipo == 2){

            fp=fopen("\\PESOS_F.TXT","rt");

            if (!fp){
                printf("\n O arquivo PESOS_F nao existe, ou não está na raiz.");
                getch();
                exit(0);
            }
    }

    fscanf(fp,"%d %d %d %d",&entradas,&c1,&c2,&c3);
    estrutura[0]=entradas;
    estrutura[1]=c1;
    estrutura[2]=c2;
    estrutura[3]=c3;

//   printf("\n\nENTRADAS= %d \nCAMADA 1= %d\nCAMADA 2= %d\nCAMADA 3=
%d\n",entradas,c1,c2,c3);
/*    entradas = numero de entradas
      c1 = numero de neuronios na primeira camada
      c2 = segunda
      c3 = terceira
*/
//printf("\n**********************************************\n\n IW11");

    for (linha=0; linha<c1 ;linha++){     /* iw11  */

//          printf("\n\n Linha %d\n",linha);

            for (coluna=0; coluna<entradas ;coluna++){
                fscanf(fp,"%f",&iw11[linha][coluna]);
//                printf("%3f\t",iw11[linha][coluna]);
          }
    }
```

```
// printf("\n*********************************************\n\n IW21");

  for (linha=0; linha<c2 ;linha++){     /* iw21  */

//          printf("\n\n Linha %d\n",linha);

            for (coluna=0; coluna<c1 ;coluna++){
                        fscanf(fp,"%f",&iw21[linha][coluna]);
//                       printf("%f\t",iw21[linha][coluna]);
            }
  }

// printf("\n*********************************************\n\n IW32");

  for (linha=0; linha<c3 ;linha++){     /* iw32  */

//          printf("\n\n Linha %d\n",linha);

            for (coluna=0; coluna<c2 ;coluna++){
                        fscanf(fp,"%f",&iw32[linha][coluna]);
//                       printf("%f\t",iw32[linha][coluna]);
             }
  }

// printf("\n\n*********************************************\n\n B1\n");

  for (linha=0; linha<c1 ;linha++){
            fscanf(fp,"%f",&b1[linha]);
//          printf("\n%f",b1[linha]);
  }

// printf("\n\n*********************************************\n\n B2\n");

  for (linha=0; linha<c2 ;linha++){
            fscanf(fp,"%f",&b2[linha]);
//          printf("\n%f",b2[linha]);
  }

// printf("\n\n*********************************************\n\n B3\n");

  for (linha=0; linha<c3 ;linha++){
```

```
            fscanf(fp,"%f",&b3[linha]);
//          printf("\n%f",b3[linha]);
  }
  fclose(fp);
}

/*****************************************************************************/
void calcula(int estrutura[4],float in[30],float iw11[30][30],float iw21[25][30],float iw32[20][25],float
b1[30],float b2[25],float b3[20],double saida[20])
{
  int linha,coluna,c1,c2,c3,entradas;
  double soma1[30],soma2[25],soma3[20],saida1[30],saida2[25];

  entradas=estrutura[0];
  c1=estrutura[1];
  c2=estrutura[2];
  c3=estrutura[3];

  // CAMADA 1

  for(linha=0; linha<c1 ;linha++){

            soma1[linha]=0;

            for(coluna=0; coluna<entradas ;coluna++){

            soma1[linha]=soma1[linha]+(iw11[linha][coluna]*in[coluna]);
    }
    soma1[linha]=soma1[linha]+b1[linha];
    saida1[linha]=( 2/(1+exp(-2*soma1[linha])) )-1;
  }

  // CAMADA 2

  for(linha=0; linha<c2 ;linha++){

            soma2[linha]=0;

            for(coluna=0; coluna<c1 ;coluna++){

            soma2[linha]=soma2[linha]+(iw21[linha][coluna]*saida1[coluna]);
```

```
    }
    soma2[linha]=soma2[linha]+b2[linha];
    saida2[linha]=( 2/(1+exp(-2*soma2[linha])) )-1;
  }

  // CAMADA 3

  for(linha=0; linha<c3 ;linha++){

          soma3[linha]=0;

          for(coluna=0; coluna<c2 ;coluna++){

          soma3[linha]=soma3[linha]+(iw32[linha][coluna]*saida2[coluna]);
    }
    soma3[linha]=soma3[linha]+b3[linha];
    saida[linha]=( 2/(1+exp(-2*soma3[linha])) )-1;
  }
}

/*****************************************************************************/
void resultado(double saida[20],int tamanho,int tipo_curva)
{
  int i,tipo;
  double max=-10;
  char *tipoc, *nome;
  FILE *fp;


  for(i=0;i<tamanho;i++){

          if(saida[i]>max){
                  max=saida[i];
                  tipo=i+1;
           }
  }

  if (max <= 0.3){

          printf("\n\n***************************************");
          printf("\n\n NAO FOI DETECTADO ERRO, A CURVA E NORMAL\n\n");
```

```
        printf("****************************************");

        if (tipo_curva ==1){
                nome=" ABERTURA";
                tipo=0;
         }

        if (tipo_curva ==2){
                nome=" FECHAMENTO";
                tipo=0;
         }

}else{

        if (tipo_curva ==1){

                 printf("\n\n****************************************");
                 printf("\n\n PARA ABERTURA A CURVA E DO TIPO %d\n\n",tipo);
                 printf("****************************************");
                 nome="ABERTURA";

        }else{

                printf("\n\n****************************************");
                printf("\n\n PARA FECHAMENTO A CURVA E DO TIPO %d\n\n",tipo);
                printf("****************************************");
                nome=" FECHAMENTO";

        }
}

if (tipo==0) tipoc="00";
if (tipo==1) tipoc="01";
if (tipo==2) tipoc="02";
if (tipo==3) tipoc="03";
if (tipo==4) tipoc="04";
if (tipo==5) tipoc="05";
if (tipo==6) tipoc="06";
if (tipo==7) tipoc="07";
if (tipo==8) tipoc="08";
if (tipo==9) tipoc="09";
```

```
   if (tipo==10) tipoc="10";
   if (tipo==11) tipoc="11";
   if (tipo==12) tipoc="12";
   if (tipo==13) tipoc="13";
   if (tipo==14) tipoc="14";
   if (tipo==15) tipoc="15";
   if (tipo==16) tipoc="16";
   if (tipo==17) tipoc="17";
   if (tipo==18) tipoc="18";
   if (tipo==19) tipoc="19";
   if (tipo==20) tipoc="20";

   fp=fopen("\\RESULTADO.TXT","wt");
   fputs(tipoc,fp);
   fputs(nome,fp);
   fclose(fp);
}
```