IFSP - INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

DESENVOLVIMENTO DE FERRAMENTA COMPUTACIONAL PARA PROJETOS DE REDES NEURAIS ARTIFICIAIS

HUGO DA SILVA BERNARDES GONÇALVES

São Paulo
2013

IFSP - INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

DESENVOLVIMENTO DE FERRAMENTA COMPUTACIONAL PARA PROJETOS DE REDES NEURAIS ARTIFICIAIS

HUGO DA SILVA BERNARDES GONÇALVES

Dissertação de Mestrado apresentada ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – IFSP como parte dos requisitos para obtenção do título de Mestre em Automação e Controle de Processos.

Orientador: Prof. Dr. Alexandre Brincalepe Campo

Co-orientador: Prof. Dr. Paulo Roberto Barbosa

INSTITUTO FEDERAL
SÃO PAULO
Campus São Paulo

MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO
CAMPUS SÃO PAULO
DIRETORIA GERAL DO *CAMPUS* SÃO PAULO
Coordenadoria de Registros Escolares de Pós-Graduação

ATA DE EXAME DE DEFESA DE DISSERTAÇÃO

Nome do Programa: **Mestrado Profissional em Automação e Controle de Processos**

Nome do(a) Aluno(a) : Hugo da Silva Bernardes Gonçalves

Nome do Orientador: Prof. Dr. Alexandre Brincalepe Campo

Nome do Co-orientador: Prof. Dr. Paulo Roberto Barbosa

Título do Trabalho: "DESENVOLVIMENTO DE FERRAMENTA COMPUTACIONAL PARA PROJETOS DE REDES NEURAIS ARTIFICIAIS"

Abaixo o resultado de cada participante da Banca Examinadora

Nome completo dos Participantes Titulares da Banca	**Sigla da Instituição**	**Aprovado / Não Aprovado**
Prof. Dr. Alexandre Brincalepe Campo – Orientador	IFSP – SPO	Aprovado
Profª. Drª. Milkes Yone Alvarenga	USJT – SP	Aprovada
Prof. Dr. Ênio Carlos Segatto	IFSP – SPO	APROVADO
Nome completo do Participante Suplente da Banca	**Sigla da Instituição**	**Aprovado / Não Aprovado**
Prof. Dr. Wârderson de Oliveira Assis	IMT	
Prof. Dr. Paulo Roberto Barbosa	IFSP – SPO	

Considerando-o: [X] APROVADO
[ ] NÃO APROVADO

Assinaturas

São Paulo, 17 de abril de 2013

Presidente da Banca

Membro Interno

Membro Externo

Observações:

RESUMO

Este trabalho apresenta um conjunto de ferramentas computacionais desenvolvidas especialmente para projetos de Redes Neurais Artificiais (RNA's) com algoritmos de *retropropagação*. Estas ferramentas são construídas através de uma plataforma baseada em linguagem gráfica (LabVIEW), propondo uma alternativa de fácil interpretação para a construção de RNA's. Também é apresentada uma interface intuitiva para construção de modelos de RNA's do tipo *Perceptron* de múltiplas camadas com algoritmo de retropropagação. A interface é disponibilizada na Web, possibilitando ao usuário criar ou treinar uma RNA através da internet. É possível, desta forma, extrair os resultados ou o modelo após o treinamento da rede. A engenharia de software aplicada ao desenvolvimento da ferramenta permite a apresentação das principais configurações para o treinamento das RNA's com algortimo de *retropropagação*. Os modelos gerados são testados para verificação das ferramentas e verificação da sua facilidade de utilização.

Palavras-chave: Linguagem Gráfica, Redes Neurais Artificiais, Engenharia de Software.

ABSTRACT

This paper presents a set of computational tools developed specifically to Artificial Neural Network (ANN) projects with *back-propagation* algorithm. This tools are built by a graphical based platform (LabVIEW), proposing an easy interpretation way to build ANN's. Also it is presented an intuitive way to build ANN using Multi Layer *Perceptron* and *back-propagation*. The interface is also available on Web, giving users the capability of create and training ANN's trough the internet. It's possible extract the results or model after the network training. The software engineering applied on this tools development allows to present the essential configurations for ANN's training that uses *back-propagation* algorithm. The generated models are tested for checking tools and evaluate its ease of use.

Keywords: Graphical Programming Language, Artificial Neural Network, Software Engineering

LISTA DE TABELAS

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

As redes neurais artificiais (RNA's) são utilizadas em diversas aplicações comerciais e industriais como, por exemplo, em modelagem e controle de processos, reconhecimento de caracteres, reconhecimento de imagens, classificação de dados, controle inteligente. É possível observar, também, como as aplicações de RNA's nas áreas de engenharias biológica e agrícola cresceram no período de 2003 a 2008 (HUANG, 2009).

Há uma grande variedade de RNA's tais como: as redes de múltiplas camadas do tipo *feed-forward*, as redes com funções de bases radiais (*BRF*), os mapas auto-organizáveis de Kohonen e as redes recorrentes. Cada uma destas redes possui o seu algoritmo de treinamento como o de *retropropagação*, para o *Perceptron* de múltiplas camadas e a clusterização de dados para as *BRF's* e as redes de Kohonen (HUANG, 2009).

Com o crescimento das aplicações com computação distribuída é possível encontrar implementações de RNA's utilizando este tipo de processamento (PLAGIANAKOS, D e VRAHATIS, 2006). Além disso, os recursos de processamento distribuído ou computação em *nuvem* vem sendo cada vez mais oferecidos através da internet a pesquisadores e estudantes (DIKAIAKOS, KATSAROS, *et al.*, 2009).

O presente documento apresenta uma alternativa para construção de modelos de Redes Neurais Artificiais (RNA's) implementada em liguagem gráfica LabVIEW. A ferramenta computacional foi desenvolvida com o objetivo de fornecer a capacidade de configuração de modelos de RNA's, permitindo o aproveitamento dos resultados em sistemas de acordo com a finalidade do usuário que irá utilizá-la. Após a implementação, o sistema é testado e os resultados obtidos serão aplicados em um sistema existente para validação das fórmulas empregadas. A seguir são demonstradas propostas de desenvolvimento de plataformas para construção de modelos de RNA's.

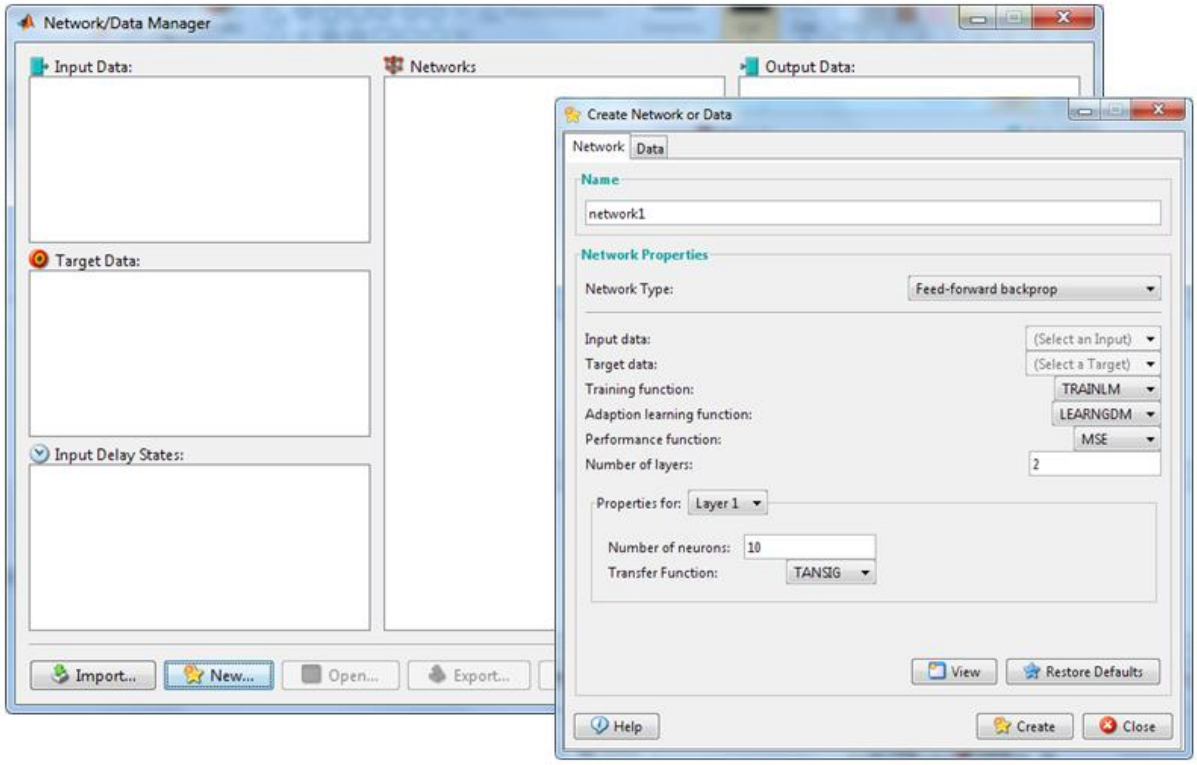
1.1 Softwares para construção de modelos de redes neurais artificiais

Existem diversas ferramentas computacionais que disponibilizam uma interface para configuração e treinamento de RNA's. Entretanto cada ferramenta destas possui características diferentes e com parâmetros de entrada específicos. Em diversos casos estas ferramentas não oferecem o produto do treinamento para que este seja utilizado em outros sistemas de forma simples e direta.

Nestes casos ainda existe a necessidade de adquirir licenças destas ferramentas além de ser necessário familiarizar-se com a linguagem visual específica em cada uma delas.

Todo este processo é, muitas vezes, trabalhoso, sendo que o objetivo principal do usuário, que é utilizar uma RNA's para uma análise ou aplicação, acaba demorado e difuso. A seguir é apresentado um modelo de ferramenta que pode ser adquirida mediante o uso de plataformas pré-existentes, neste caso o MATLAB, e a aquisição dos respectivos módulos ou *toolbox* para utilização de Redes Neurais Artificiais. Na Figura 1 está apresentada uma parte da interface deste módulo que permite também a utilização de comandos em formato de texto. Nesta ferramenta, caso o usuário queira acrescentar algum recurso como a importação de um arquivo, terá que digitar comandos em formato de texto.

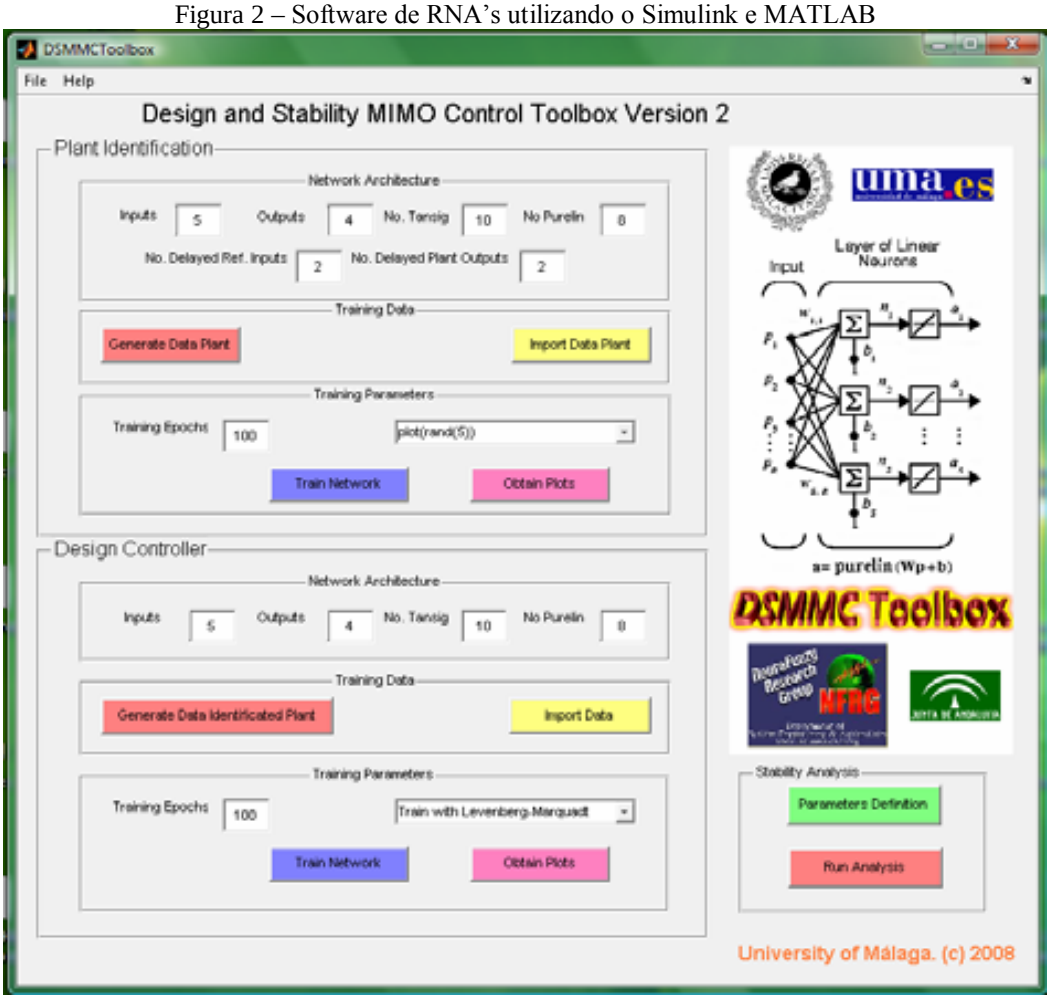
Figura 1 – Visual das telas do toolbox para o MATLAB



Fonte: print screen do toolkit do MATLAB executado em sistema operacional Windows

Alternativas de utilização das ferramentas do MATLAB em sistemas de identificação e controle podem ser criadas como visto em CANETE, PEREZ e SAZ, 2008. Neste trabalho os autores também se utilizam dos recursos de interconexão do MATLAB com a ferramenta Simulink para a construção dos modelos em blocos. A seguir na Figura 2, uma parte da interface produzida neste trabalho.

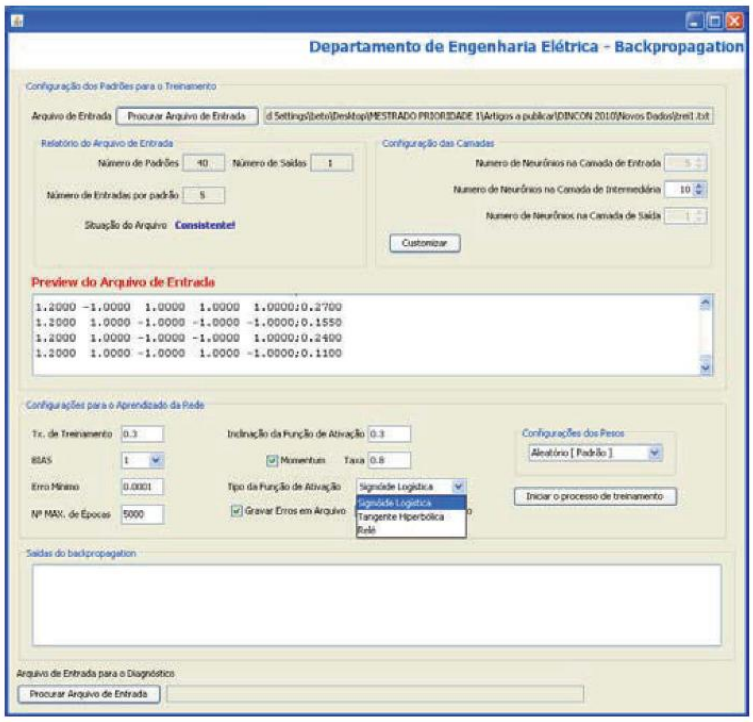
Figura 2 – Software de RNA's utilizando o Simulink e MATLAB



Fonte: CANETE, PEREZ e SAZ, 2008

Em CAMPOS, LOTUFO, *et al.*, 2010 é possível verificar a construção de um software para treinamento de redes *feed-forward* com algoritmo de *retropropagação*. A ferramenta é desenvolvida em linguagem de programação JAVA utilizando o paradigma de programação orientada a objeto difundido na década de 90 que pressupõe uma organização de software em termos de coleção de objetos discretos incorporando estrutura e comportamento próprios. A seguir na Figura 3, é apresentada uma das telas da ferramenta desenvolvida em linguagem de programação JAVA sendo executado em sistema operacional Windows:

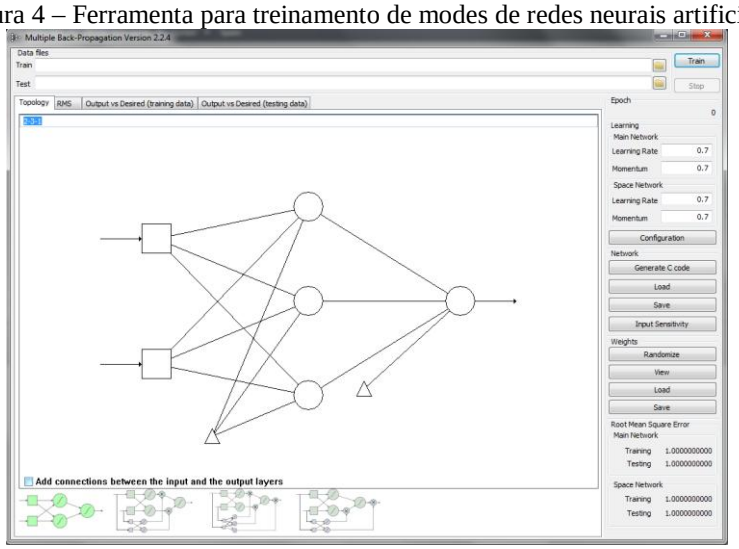
Figura 3 – Software para treinamento de RNA's desenvolvido em linguagem JAVA



Fonte: CAMPOS, LOTUFO, *et al.*, 2010

Em outro trabalho, como é o caso do programa Multiple Back-Propagation (LOPES e RIBEIRO, 2001), também foi desenvolvido uma ferramenta para configurações de Redes Neurais Artificiais que executa em sistema operacional Windows. A imagem do software está na Figura 4:

Figura 4 – Ferramenta para treinamento de modes de redes neurais artificiais



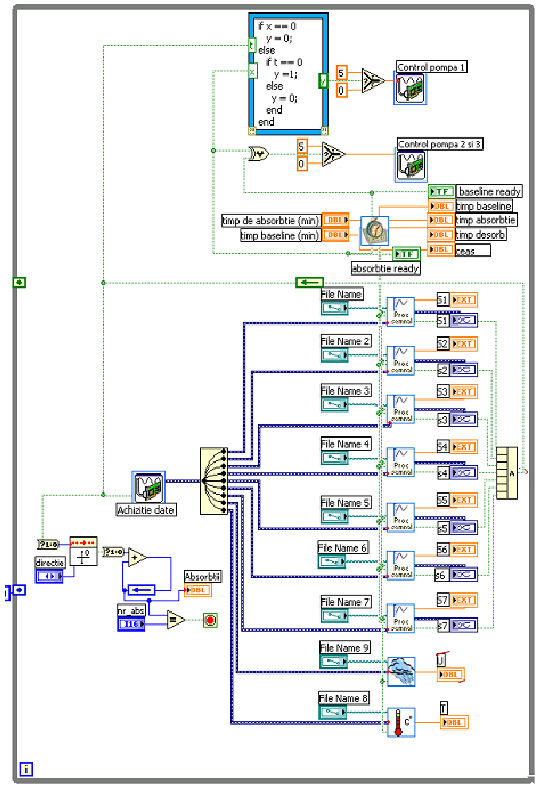
Fonte: print screen do software *MBP* executado em sistema operacional Windows

A ferramenta desenvolvida por Lopes e Ribeiro permite algumas configurações mas para utilizá-la é necessário construir um arquivo com os dados de entrada e saída desejados seguindo um padrão pré-determinado. Os desenvolvedores também produziram trabalhos para aperfeiçoar o processamento das operações com auxílio de processadores gráficos (LOPES e RIBEIRO, 2011), com o objetivo de melhorar o desempenho dos algoritmos para treinamento de redes neurais artificiais.

Em função dos algoritmos de RNA's necessitarem de rápido processamento devido a quantidade de operações que realizam, diversos trabalhos tem sido desenvolvidos com o objetivo de executar estes algoritmos em micro-chips.

Em TISAN, CIRSTEA, *et al.*, 2010, é possível encontrar uma alternativas de implementação dos algoritmos em chips como o FPGA, onde é proposto um sistema artificial de aprendizagem de olfato. O ambiente de programação em LabVIEW é utilizado para aquisição e armazenamento dos sinais originados dos ensaios. Estes ensaios são posteriormente inseridos em uma rede que é implementada em FPGA utilizando a linguagem *VHDL*. Na Figura 5 a seguir, é apresentado o código utilizado para a aquisição dos sinais:

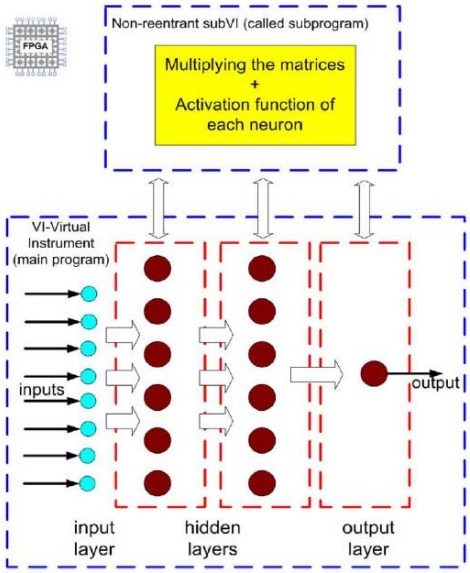
Figura 5 – Código em LabVIEW (executado em Windows) utilizado para aquisição dos sinais a serem utilizados no treinamento de uma rede neural artificial



Fonte: TISAN, CIRSTEA, *et al.*, 2010

No trabalho de KOWALSKA e KAMINSKI, 2011, é apresentada a implementação de uma rede neural de múltiplas camadas para um estimador de velocidade em um sistema de *driver* duas massas. O algoritmo é implementado em um chip FPGA utilizando a linguagem LabVIEW. O modelo com 2 camadas intermediárias, é desenvolvido especificamente para a aplicação que teve de levar em consideração questões como o espaço que o algoritmo ocupa no chip FPGA, já que este possui memória e tamanho de variáveis determinados. Segundo KOWALSKA e KAMINSKI, 2011, o uso do código em LabVIEW como linguagem de alto nível para desenvolver RNA's oferece vantagens como a rápida prototipação, rápida verificação, e a possibilidade de implementação em FPGA. A seguir na Figura 6 é apresentado o diagrama com a representação da RNA e sua interação no chip FPGA.

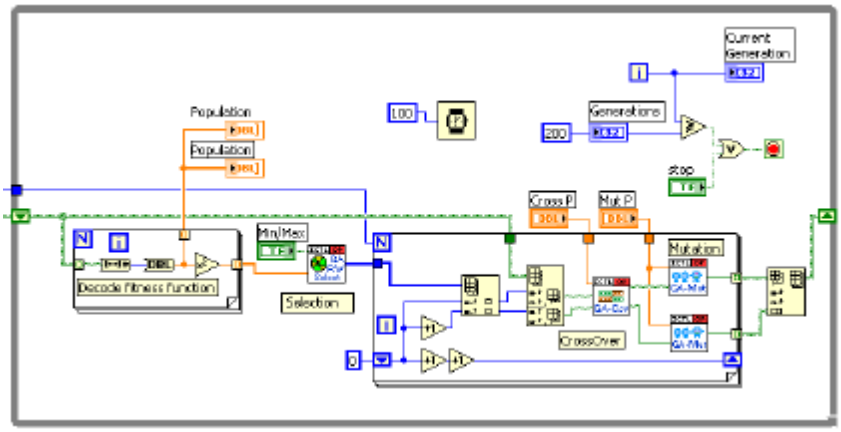
Figura 6 – Diagrama da implementação da RNA com duas camadas intermediárias



Fonte: KOWALSKA e KAMINSKI, 2011

Nos trabalhos de PONCE e GUTIERREZ, 2010, é possível encontrar um conjunto de algoritmos de RNA's especialmente desenvolvido para aplicações em LabVIEW. Entretanto neste trabalho não há uma interface padronizada para inserção dos dados de treinamento em um algoritmo como o *Perceptron* de múltiplas camadas, por exemplo. A seguir na Figura 7, a imagem de um código para algoritmos genéticos utilizando estas ferramentas:

Figura 7 – Código em LabVIEW para implementação de RNA's



Fonte: PONCE e GUTIERREZ, 2010

1.2 Objetivos

Observou-se que, apesar de existirem alternativas para implementações de RNA's não se tem uma interface intuitiva e padronizada para o treinamento das redes sendo possível extrair os modelos para utilizá-los rapidamente em outros sistemas. As ferramentas existentes possuem características muito específicas que exigem muito conhecimento e estudo para operá-las.

Optou-se por desenvolver uma ferramenta com maior capacidade para integração em sistemas de reconhecimento de padrão e sistemas de controle, por exemplo, utilizando a linguagem gráfica LabVIEW. Neste contexto, foi desenvolvida uma alternativa para utilização das ferramentas e também dos conceitos de RNAs utilizando redes do tipo *Perceptron* de Múltiplas Camadas com algoritmo de *retropropagação*. Foram identificadas as principais características de modelos de redes neurais artificiais e com isso determinados os parâmetros para as configurações dos modelos. A interface deve ser intuitiva e, portanto, a ferramenta deve ser baseada nas principais referências utilizadas para o estudo de RNA's com o algoritmo de *retropropagação*. Ao final do desenvolvimento a ferramenta foi disponibilizada em ambiente Web, no qual o usuário pode acessá-la através da internet.

1.3 Etapas do trabalho

As etapas deste trabalho constituem-se de:

a) Pesquisa sobre os algoritmos de configuração e treinamento de RNAs;

b) Definição dos parâmetros de entrada e saída da ferramenta computacional;

c) Elaboração e programação da interface para a interação com a ferramenta;

d) Construção dos algoritmos para configuração e treinamento de RNAs;

e) Validação das funções;

f) Teste dos resultados em um sistema de controle;

g) Avaliação dos resultados e considerações em relação à proposta.

2 REDES NEURAIS ARTIFICIAIS (RNA'S)

Neste capítulo é abordada a origem dos conceitos de RNA's e a evolução dos cálculos para os algoritmos existentes.

2.1 Primeiros estudos de RNA's

O termo Redes Neurais se deve ao fato de os primeiros estudos de redes neurais artificiais terem sido inspirados no processo de interpretação e aprendizado do cérebro humano.

Com a motivação de se obter resultados semelhantes aos do cérebro humano é que se iniciaram os primeiros estudos de RNA's. A rede artificial é normalmente implementada utilizando-se componentes eletrônicos ou é simulada por programação em um computador digital. Entretanto o estudo de Redes Neurais Artificiais está relacionado a fornecer alternativas onde os enfoques numéricos convencionais não são adequados. As RNA's são apenas inspiradas no conhecimento atual sobre os processos nervosos e biológicos e não necessariamente buscam ser realísticos em todos os detalhes. A palavra *neural* é empregada hoje, devido a razões históricas, já que os primeiros pesquisadores vieram de áreas da Psicologia e Biologia. (NASCIMENTO e YONEYAMA, 2008)

As Redes Neurais Artificiais executam suas tarefas de forma **paralela**, com grande número de processamentos simples, mas com alto grau de interconexão entre eles. A seguir, uma definição formal proposta por Hetch-Nielson:

> Uma rede neural artificial é uma estrutura que processa informação de forma paralela e distribuída e que consiste de unidades computacionais (as quais podem possuir uma memória local e podem executar operações locais) interconectadas por canais unidirecionais chamados de **conexões.** Cada unidade computacional possui uma única conexão de saída, que pode ser dividida em quantas conexões laterais se fizer necessário, sendo que cada uma destas conexões transporta o mesmo sinal, o sinal de saída da unidade computacional. Este sinal de saída pode ser contínuo ou discreto. O processamento executado por cada unidade computacional pode ser definido arbitrariamente, com a restrição de que ele deve ser completamente local, isto é, deve depender somente dos valores atuais dos sinais de entrada que chegam até a unidade

> computacional via as conexões e dos valores armazenados na memória local da unidade computacional.(HETCH-NIELSON, 1990)

Observa-se neste trecho que uma rede neural artificial possui processamento paralelo, distribuído e é composta por unidades computacionais que são, por sua vez, interconectadas a outras unidades.

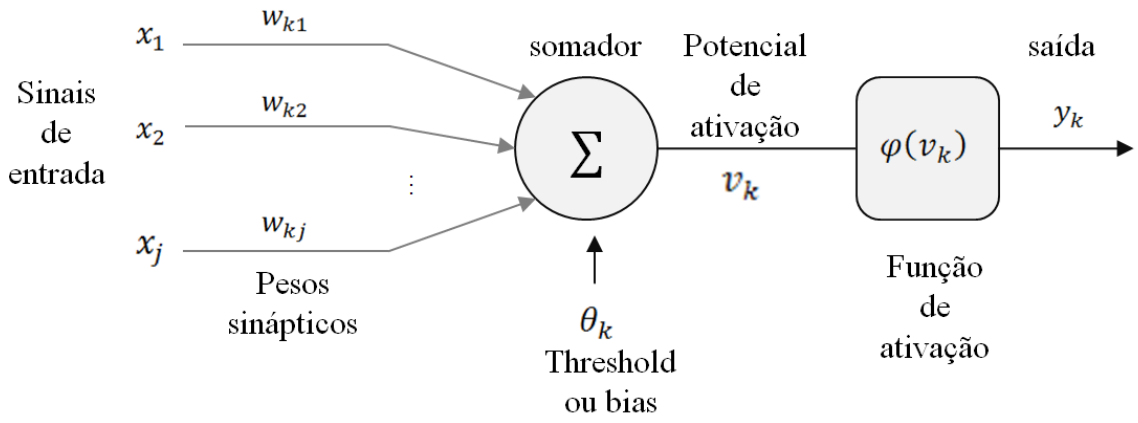
2.2 O Neurônio artificial

Neste capítulo é apresentado o neurônio sob uma visão computacional.

2.2.1 Modelo de um Neurônio (artificial)

Um neurônio artificial é uma unidade computacional que será a base para a formação das redes neurais artificiais. Na Figura 8 são identificadas as partes do modelo neuronal:

Figura 8 – Neurônio computacional



Fonte: elaborado pelo autor

O modelo apresentado na Figura 8 possui três partes:

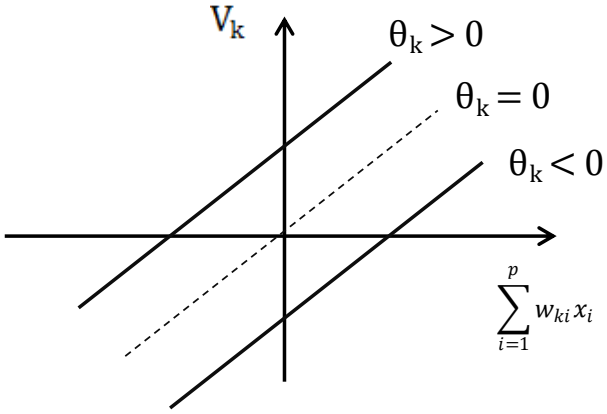
a) Os elos de conexão ou as *sinapses*, onde é aplicado ao sinal de entrada x_j, e um *peso* ou *força* representado por w_{kj}. Estes pesos podem possuir valores negativos, diferente do que ocorre em uma sinapse do cérebro.

b) Um *somador* que soma os sinais de entrada após eles serem ponderados pelos respectivos pesos.

c) Uma *função de ativação*, onde é aplicada uma função que restringe a saída do neurônio.

Este modelo inclui também um *bias* θ_k que tem o efeito de aumentar ou diminuir a entrada da função de ativação, dependendo se ele é positivo ou negativo. A Figura 9 mostra a saída do neurônio em função da variação do *bias.*

Figura 9 – Saída produzida com a presença de um *bias*



Fonte: adaptado de HAYKIN, 2001

Sendo o campo local induzido v_k representado por,

$$V_k = \theta_k + \sum_{i=1}^{p} w_{ki} x_i \quad (1)$$

A saída y_k do neurônio dada por:

$$y_k = \varphi\left(\theta_k + \sum_{i=1}^{p} w_{ki} x_i\right) \quad (2)$$

Sendo $\varphi_j(v_k)$ correspondente à função de ativação sobre a saída v_k.

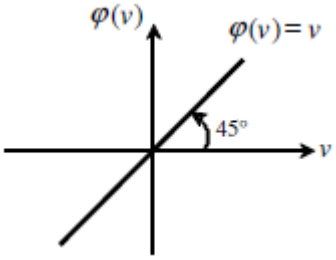
2.2.2 Tipos de Função de Ativação

Dentre os tipos mais comuns de função de ativação tem-se:

a) *Função Linear e Linear por partes*

Na *Função Linear*, a saída da função $\varphi(\boldsymbol{v_k})$ é o próprio $\boldsymbol{v_k}$. É representada no gráfico da Figura 10:

Figura 10 – *Função linear*

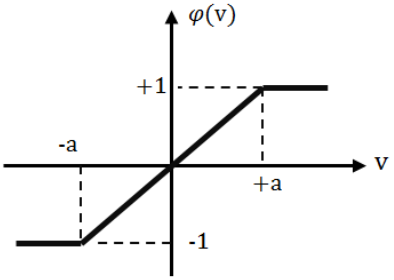


Fonte: elaborado pelo autor

Na função *Linear por partes*, se a operação for mantida sem entrar em saturação, comporta-se como um combinador linear. Se o fator de amplificação for infinitamente grande, se reduz à *Função de Limiar*. A seguir, na Figura 11, a representação da *Função Linear por Partes* que apresenta a relação da equação 3:

$$\varphi(v)=\begin{cases} 1 \quad se & v \geqslant +a \\ v \quad se & -a<v<a \\ -1 \quad se & v \leqslant -a \end{cases} \qquad (3)$$

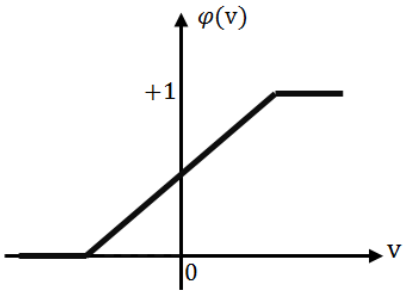
Figura 11 – Gráfico da *função linear* por partes passando pela origem



Fonte: elaborado pelo autor

Na Figura 12 vemos outro exemplo, apenas com valores positivos:

Figura 12 – *Função linear* por partes sem passar pela origem



Fonte: elaborado pelo autor

Neste modelo tem-se:

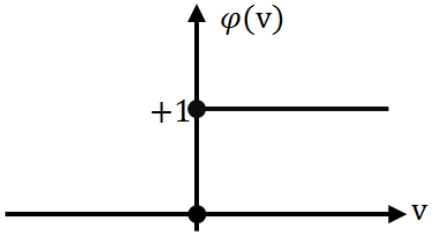
$$\varphi(v)=\begin{cases} 1, & v \geqslant +1/2 \\ v, & +1/2 > v > -1/2 \\ 0, & v \leqslant -1/2 \end{cases} \qquad (4)$$

Conclui-se que o fator de amplificação dentro da região linear é a unidade. Esta forma de função pode ser vista como uma aproximação de um amplificador não-linear (HAYKIN, 2001).

b) *Função de Limiar* ou *Threshold*, onde a saída é alta (1) apenas quando a combinação linear de todas as entradas for maior ou igual a um certo limiar, e baixa (0) no caso contrário. Ou seja, $\boldsymbol{y_k = 1}$ se $\boldsymbol{v_k \geq 0}$ e $\boldsymbol{y_k = 0}$ se $\boldsymbol{v_k} < 0$, conforme a equação 5. A seguir esta função é representada na Figura 13:

$$\varphi(v)=\begin{cases} 1 \quad se \quad v \geqslant 0 \\ 0 \quad se \quad v < 0 \end{cases} \qquad (5)$$

Figura 13 – *Função de limiar*



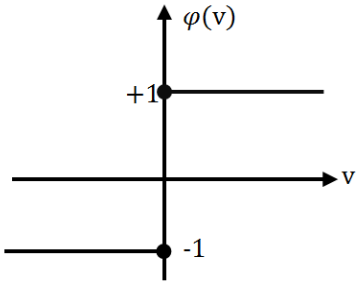
Fonte: elaborado pelo autor

Esta função da Figura 13 é originalmente criada em analogia aos processos nervosos biológicos do cérebro. Entretanto as redes neurais artificiais evoluíram para a utilização de outras funções de ativação.

Pode-se ainda representar a *FunçãoThreshold* com valores negativos conforma a Figura 14:

$$\varphi(\mathrm{v}) = \begin{cases} +1 & \text{se} \quad \mathrm{v} \geqslant 0 \\ -1 & \text{se} \quad \mathrm{v} < 0 \end{cases} \tag{6}$$

Figura 14 – *Função Threshold*



Fonte: elaborado pelo autor

c) *Funções Sigmóide* e *Tangente Hiperbólica*. A função sigmoidal, conhecida também como *S-shape*, é uma função semi-linear. É possível definir várias funções sigmoidais. Uma das funções sigmoidais mais importantes é a função logística definida por:

$$y = \frac{1}{1 + e^{-x/T}} \tag{7}$$

Onde parâmetro T determina a suavidade de curva.

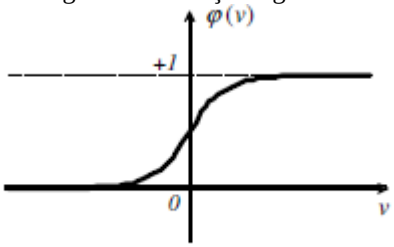
A função *Sigmóide*, que possui gráfico em forma de s, pode ser também utilizada em redes neurais artificiais.

Tem-se nesta função a seguinte operação:

$$\varphi(v) = \frac{1}{1 + e^{(-av)}} \tag{8}$$

Onde *a*, é o parâmetro de inclinação da função sigmóide. A saída desta função varia entre 0 e 1, conforme apresentado no gráfico a seguir da Figura 15:

Figura 15 – Função sigmóide



Fonte: elaborado pelo autor

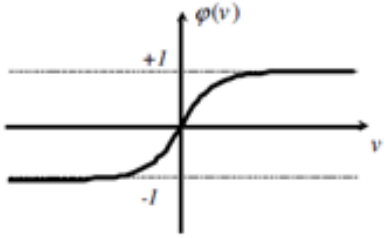
Uma característica importante para redes neurais artificiais é que esta função é diferenciável, o que não ocorre com a função de limiar.

A função *Tangente Hiperbólica* é correspondente à função *Sigmoidal* e permite valores de saída entre -1 e 1. É definida por:

$$\varphi(v) = \tanh(v) = \frac{1 - e^{-v}}{1 + e^{-v}} \tag{9}$$

Seu gráfico é representado pela Figura 16:

Figura 16 – Função tangente hiperbólica



Fonte: elaborado pelo autor

Outra característica importante é que as funções *Sigmóide* e *Tangente Hiperbólica* possuem a curva de saída em formato de S, uma versão suave (com derivada finita) em relação à função de limiar.

2.2.3 Outros tipos de unidade computacional

Podem ser encontrados outros tipos de unidades computacionais, como, por exemplo, a unidade com função de base radial.

Neste caso tem-se:

$$\boldsymbol{v_k} = \frac{1}{\tau_i} ||x - C_i|| \tag{10}$$

Onde *Ci* é um vetor que determina o centro da unidade e $\boldsymbol{\tau_i}$é um escalar que representa o "espalhamento" da função de ativação da unidade.

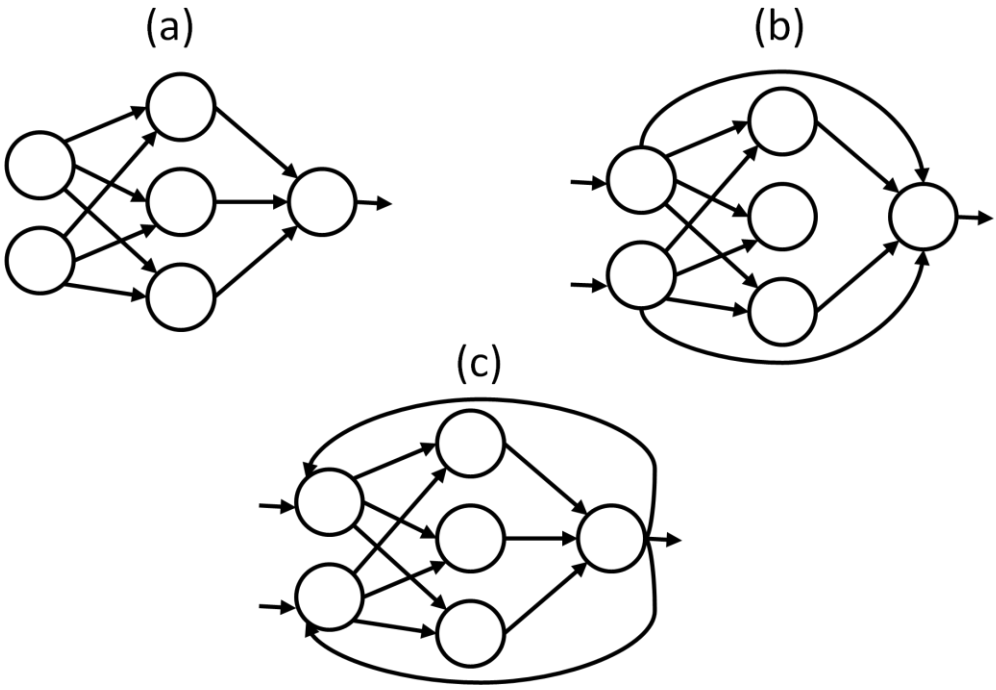
$$y_k \; = \; exp[-(v_{ki})^2] \tag{11}$$

Assim, a saída é máxima quando x = Ci e decai suavemente enquanto xi se afasta do centro Ci.

2.3 Topologia de Redes Neurais Artificiais

São representadas na Figura 17 algumas topologias de redes neurais artificiais:

Figura 17 – Topologias para RNA's: a) camadas isoladas; b) conexões diretas; c) com realimentação



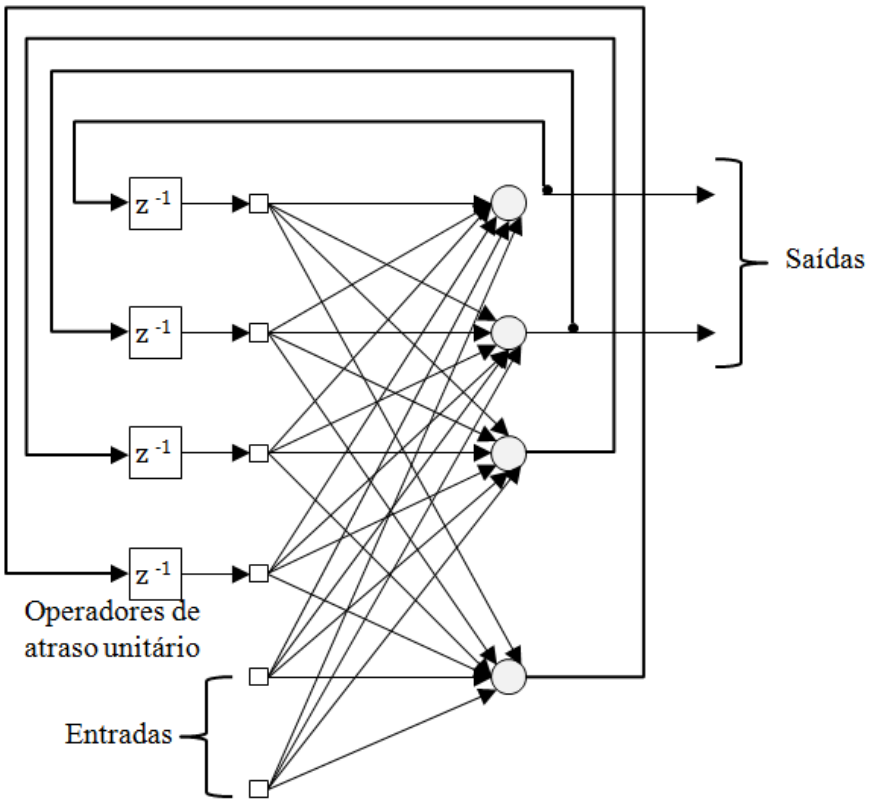
Fonte: elaborado pelo autor

Em (a) temos uma rede sem realimentação e camadas isoladas. Em (b) temos uma rede sem realimentação e com conexões diretas entre camadas de entrada e saída. Em (a) e (b) tem-se redes do tipo *feedforward*, onde uma unidade envia sua saída apenas para unidades das quais ela não recebe nenhum valor direta ou indiretamente (via outras unidades). Podemos denominar a rede representada em (a) como *estritamente feedforward* já que todas as unidades enviam suas saídas para as unidades situadas na próxima camada.

Em (c) tem-se uma rede com realimentação. É bastante utilizada em redes do tipo Hopfield (HOPFIELD, 1982) onde os pesos são fixos e pré-determinados.

A rede que possui laços de realimentação também pode ser chamada de *rede recorrente*. Na Figura 18 a seguir é ilustrada uma rede recorrente com neurônios ocultos e *elementos de atraso unitário*. A utilização destes elementos resulta em um comportamento dinâmico não-linear, admitindo-se que a rede contenha unidades não-lineares (HAYKIN, 2001). Na Figura 18 existem alguns neurônios que recebem o sinal com atraso representado por $\boldsymbol{z^{-1}}$.

Figura 18 – Rede realimentada com sinal de atraso



Fonte: HAYKIN, 2001

2.4 Processo de Aprendizagem

Assim como o cérebro humano, as redes neurais artificiais têm a capacidade de aprender através de exemplos apresentados por meios externos. Uma das características de uma RNA é possuir uma fase de aprendizagem onde a rede extrai características relevantes de padrões de informações apresentados, criando representação própria para o problema. A seguir temos uma definição de uma Rede Neural Artificial proposta em 1970:

> A utilização de uma Rede Neural Artificial na solução de uma tarefa passa inicialmente por uma fase de aprendizagem, onde a rede extrai informações relevantes de padrões de informação apresentados para a mesma, criando assim uma representação própria para o problema. A etapa de aprendizagem consiste em um processo iterativo de ajuste de parâmetros da rede, os pesos das conexões entre as unidades de processamento que guardam, ao final do processo, o conhecimento que a rede adquiriu do ambiente em que está operando. (MENDEL e MCLAREN, 1970)

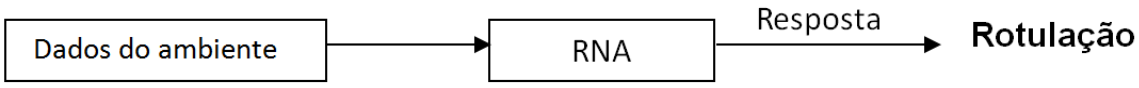
Na fase de aprendizado ou treinamento de uma rede neural artificial, utiliza-se uma regra para alterar os valores da matriz de pesos W (e outros parâmetros que a rede possa possuir). Para isso utilizam-se informações do meio externo disponibilizadas pelo supervisor deste aprendizado. Assim, treinamento, aprendizado ou adaptação se referem às alterações nos parâmetros modificáveis da rede neural (LIPPMAN, 1987).

Diferentes métodos de aprendizado podem ser utilizados e são classificados de acordo com o grau de supervisão no processo. Este grau determina o quanto o supervisor interfere na rede com informações que podem auxiliar no treinamento da rede. A seguir são descritos os tipos de aprendizados em relação ao seu grau de supervisão.

2.4.1 Supervisão Muito Fraca

Também denominado na literatura como *aprendizado não supervisionado*, o algoritmo tenta separar em grupos ou categorias os dados de entrada. O supervisor deve nomear estes agrupamentos. Este tipo de aprendizado ocorre nas Redes de Kohonen (KOHONEN, 1984), por exemplo. O supervisor participa dando os rótulos aos grupos, portanto mesmo que seja uma participação mínima, faz com que o termo *não supervisionado* não seja apropriado (NASCIMENTO e YONEYAMA, 2008). O diagrama que representa este aprendizado está na Figura 19 a seguir:

Figura 19 – Aprendizado com Supervisão Muito Fraca



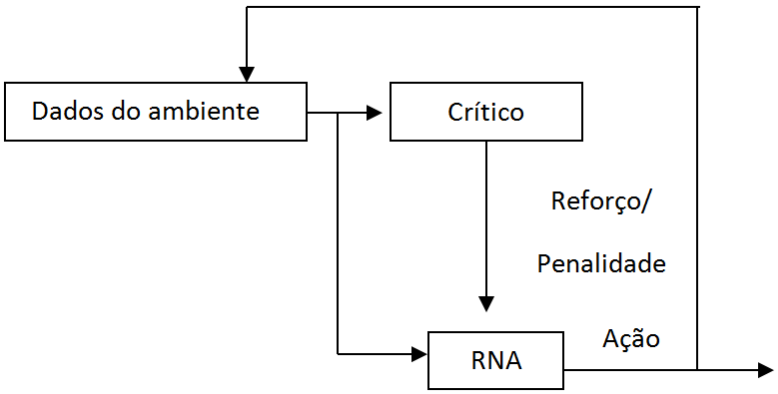
Fonte: elaborado pelo autor

2.4.2 Supervisão fraca

Neste tipo de supervisão, o supervisor fornece uma avaliação crítica abrangente da saída da rede neural artificial. Esta avaliação pode ser, por exemplo, certo ou errado, sucesso, fracasso. São exemplos deste aprendizado os algoritmos de aprendizado por reforço

(*reinforcement learning*) ou de punição/recompensa (*reward/punishment*). É representado pelo diagrama da Figura 20:

Figura 20 – Aprendizado por reforço

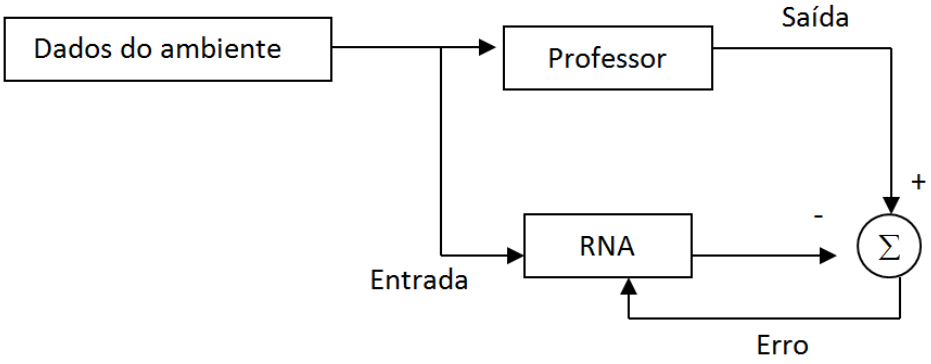


Fonte: elaborado pelo autor

2.4.3 Supervisão Forte

É chamado na literatura de aprendizado supervisionado (HUSH e HORME, 1993) Em um grau de supervisão maior, onde o supervisor fornece para a rede neural artificial um conjunto de entradas e saídas desejadas. A rede deve ajustar os seus pesos para representar a saídas desejadas para cada entrada. Este caso ocorre no algoritmo de *Retropropagação*. Para cada entrada do conjunto de treinamento a saída desejada é comparada à saída e os pesos são alterados para diminuir esta diferença conforme representado na Figura 21.

Figura 21 – Aprendizado com Supervisão Forte



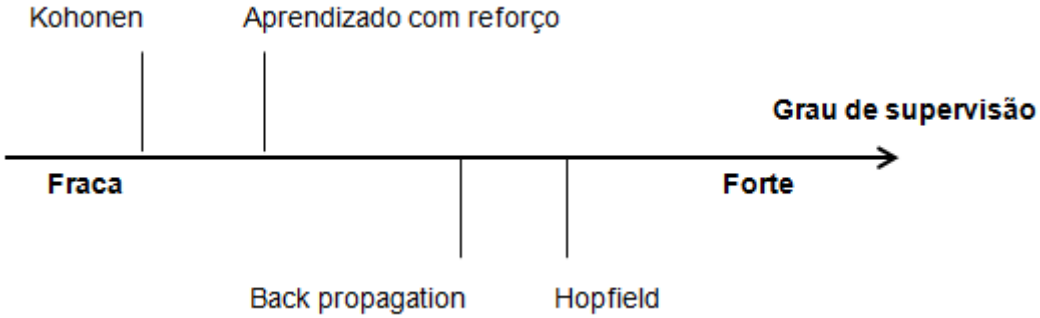
Fonte: elaborado pelo autor

2.4.4 Supervisão Muito Forte

O supervisor fornece para a Rede Neural Artificial os valores dos pesos. São exemplos deste tipo de aprendizado as redes de Hopfield (HOPFIELD, 1982). Na literatura estas redes são denominadas de **pesos fixos**.

Na Figura 22 a seguir estão representados os tipos de aprendizagem em relação ao seu grau de supervisão.

Figura 22 – Tipos de rede em relação ao seu grau de supervisão



Fonte: adaptado de NASCIMENTO e YONEYAMA, 2008

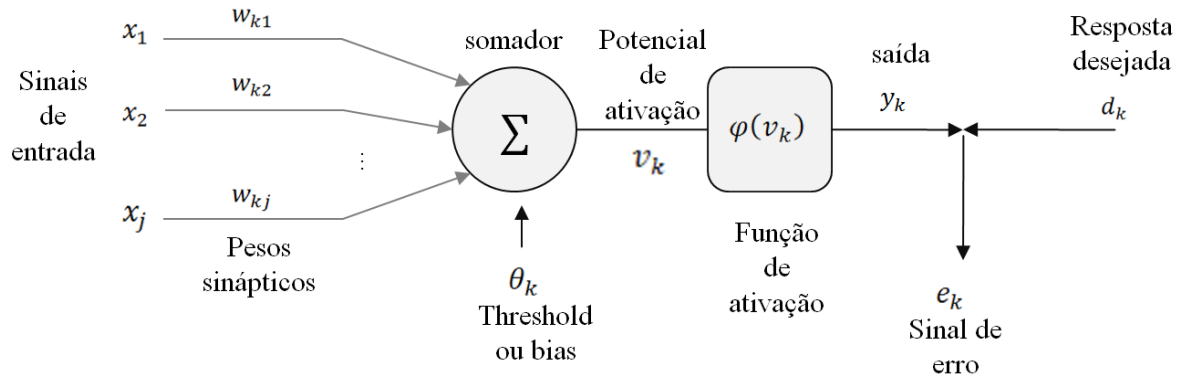
2.5 Regras de Aprendizagem

São apresentadas quatro tipos de regras de aprendizagem: aprendizagem por correção de erro, aprendizagem Hebbiana, aprendizagem competitiva e aprendizagem de Boltzman.

2.5.1 Aprendizagem por correção de erro

Considera-se um neurônio k acionado por um vetor $\boldsymbol{x}(\boldsymbol{n})$, onde n é o passo de tempo de cada iteração que envolve os ajustes dos pesos sinápticos. No tipo de aprendizagem por correção de erro um sinal de saída $\boldsymbol{y_k}(\boldsymbol{n})$ é comparado a uma *saída desejada* $\boldsymbol{d_k}(\boldsymbol{n})$. Gera-se então um erro $\boldsymbol{e_k}(\boldsymbol{n})$ que é a diferença entre o sinal de saída e a saída desejada. Este modelo é representado na Figura 23.

Figura 23 – Modelo do neurônio e sinal de erro



Fonte: elaborado pelo autor

Com o sinal de erro $\boldsymbol{e(k)}$ aplica-se uma sequência de correções nos pesos sinápticos. Com isso deve-se aproximar o sinal de saída $\boldsymbol{y(k)}$ da resposta desejada $\boldsymbol{d_k(n)}$. O objetivo é alcançado quando se minimiza uma *função de custo* que é definida como:

$$\varepsilon(n) = \frac{1}{2e(k)^2(n)} \tag{12}$$

Os ajustes dos pesos sinápticos continuam até que se atinja um estado estável onde os pesos sinápticos estão *estabilizados*. Quando isto ocorrer termina-se o processo de ajuste dos pesos sinápticos.

A regra comumente utilizada para minimizar a função de custo $\boldsymbol{\varepsilon(n)}$ é a regra delta ou Widrow-Hoff(WIDROW e HOFF, 1960). De acordo com esta regra, o ajuste $\boldsymbol{\Delta w_{kj}(n)}$ é definido por:

$$\Delta w_{kj}(n) = \eta\, e_k(n) x_j(n) \tag{13}$$

Onde $\boldsymbol{x_j}$ é um elemento de entrada do vetor $\boldsymbol{x(n)}$, $\boldsymbol{e_k(n)}$ é sinal de erro e $\boldsymbol{\eta}$ é uma constante positiva que determina a *taxa de aprendizado.* Assim, o ajuste no peso sináptico é proporcional ao produto entre o sinal de erro pela entrada $\boldsymbol{x_j(n)}$.

Após o cálculo do ajuste sináptico $\boldsymbol{\Delta w_{kj}(n)}$, o valor do peso poderá ser corrigido da seguinte forma:

$$w_{kj(n+1)} = w_{kj}(n) + \Delta w_{kj}(n) \tag{14}$$

O parâmetro de aprendizado $\boldsymbol{\eta}$ possui influência no algoritmo sendo inversamente proporcional à curva de aprendizagem (WIDROW e STEAMS, 1985).

2.5.2 Aprendizagem Hebbiana

O *aprendizado de Hebb* é assim chamado em homenagem ao neuropsicólogo Hebb. Em seu livro *The Organization of Behavior* (HEBB, 1949), ele propõe que uma modificação a nível celular pode ocorrer quando um axônio de uma célula que está próximo a outra célula realiza um disparo. Esta modificação metabólica pode ocorrer nas duas células e assim a primeira célula aumentará a eficiência de seus disparos após estas modificações.

Expandindo a ideia originalmente do contexto neurobiológico, temos a seguinte regra: A força de uma sinapse é aumentada quando os dois neurônios possuem seus dois lados de sinapse ativados ao mesmo tempo. A sinapse pode ser eliminada se estes lados dos neurônios não forem ativados ao mesmo tempo (CHANGEUX e DANCHIN, 1976).

A formulação matemática da aprendizagem Hebbiana se dá da seguinte forma:

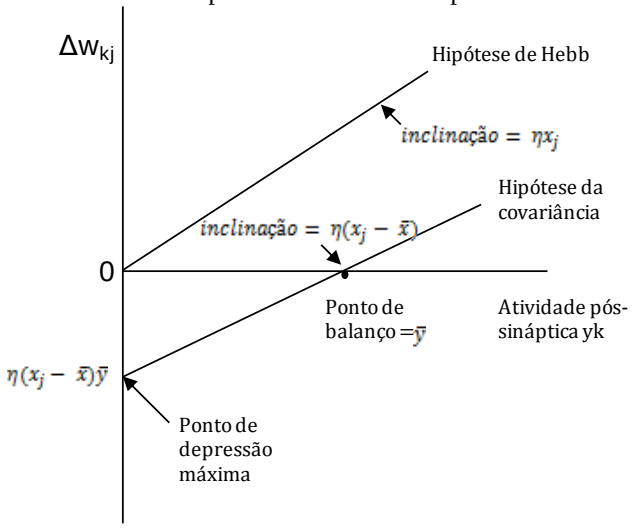
$$\Delta w_{kj(n)} = F\big(y_k\,(n)x_j\,(n)\big) \tag{15}$$

sendo que F(. , .) É uma função tanto do sinal de saída quanto do sinal de entrada, conforme pode-se observar. Uma das formas adaptadas desta última forma de aprendizagem hebbiana é a hipótese hebbiana conforme a fórmula a seguir:

$$\Delta w_{kj(n)} = \eta y_k(n)x_j(n) \tag{16}$$

Sendo $\boldsymbol{\eta}$ a *taxa de aprendizagem*. Na Figura 24 está uma representação gráfica da fórmula anterior.

Figura 24 – Gráfico da hipótese de Hebb e da hipótese da covariância



Fonte: HAYKIN, 2001

Observa-se que o ajuste $\Delta \boldsymbol{w_{kj}}$ a ser aplicado ao peso sináptico aumenta com a repetição do sinal de entrada. Isto leva a conexão sináptica à saturação.

Segundo a *hipótese da covariância* (SEJNOWSKI, 1977), os sinais de entrada $\boldsymbol{x_j(n)}$ e saída $\boldsymbol{y_k(n)}$ são substituídos pela diferença destes sinais em relação a seus valores médios ao longo do tempo. Assim o ajuste ao peso sináptico é definido por:

$$\Delta w_{kj(n)} - \eta \left(x_j - \bar{x}\right)(y_k - \bar{y}) \tag{17}$$

Sendo $\boldsymbol{\eta}$ a taxa de aprendizado e $\overline{\boldsymbol{x}}$e $\overline{\boldsymbol{y}}$ os valores médios dos sinais. A diferença entre a hipótese hebbiana inicial pode ser demonstrada no gráfico da Figura 19. O cruzamento com o eixo $\boldsymbol{y_k}$ na hipótese de hebb se dá na origem. Já na hipótese da covariância ocorre quando $\overline{\boldsymbol{y}} = \boldsymbol{y_k}$. O peso sináptico $\boldsymbol{w_{kj}}$ será reforçado quando as condições $\boldsymbol{x_j} > \overline{\boldsymbol{x}}$ e $\boldsymbol{y_k} > \overline{\boldsymbol{y}}$ forem verdadeiras. Pórém o peso sináptico $\boldsymbol{w_{kj}}$ será diminuído se $\boldsymbol{x_j} > \overline{\boldsymbol{x}}$ quando $\boldsymbol{y_k} < \overline{\boldsymbol{y}}$ ou também quando $\boldsymbol{y_k} > \overline{\boldsymbol{y}}$ e $\boldsymbol{x_j} < \overline{\boldsymbol{x}}$)

2.5.3 Aprendizagem Competitiva

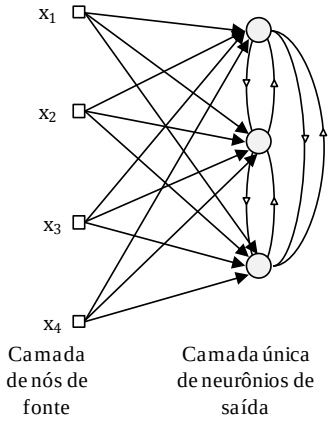
Na *aprendizagem competitiva* os neurônios de saída de uma rede neural artificial competem entre si para se tornar ativos. Desse modo somente um único neurônio de saída estará ativo a cada instante.

Alguns elementos devem ser introduzidos para que se tenha uma regra de *aprendizagem competitiva*. Deve haver um conjunto de neurônios iguais entre si e um mecanismo que permita a competição entre os neurônios para que um deles responda a um determinado conjunto de entradas(RUMELHART e ZISPER, 1985).

Assim, cada neurônio individual aprende a responder à entrada detectando características ou padrões diferentes dos padrões de entrada. Isto será importante para a descoberta de características e que podem ajudar determinar padrões de entrada da rede.

Um exemplo simples seria uma rede de uma única camada com conexões de realimentação entre os neurônios conforme a Figura 25 a seguir:

Figura 25 – Rede com realimentação e apenas uma camada



Fonte: HAYKIN, 2001

Para que um dos neurônios vença a competição, o seu campo local induzido $\boldsymbol{v_k}$ deve ser o maior em relação aos outros. Então o sinal de saída deste neurônio será colocado em 1. Os sinais de saída dos outros neurônios ficarão em 0. Assim tem-se:

$$y_k = \begin{cases} 1 \; se \; v_k \; > \; v_j \; para \; todos \; j.j \; \neq k \\ 0 \qquad caso \; contrário \end{cases} \tag{18}$$

Onde o campo $\boldsymbol{v_k}$ será resultado da ação combinada das entradas e das realimentações do neurônio *k*.

Considerando que a distribuição dos pesos sinápticos $\boldsymbol{w_{kj}}$ atenda a seguinte regra:

$$\sum_j w_{kj} = 1 \textbf{ para todo } k \tag{19}$$

Ou seja, uma quantidade fixa e positiva é distribuída entre os pesos sinápticos. No processo de aprendizagem, um neurônio ativo desloca o peso sináptico dos seus nós inativos

para os nós ativos. Se um neurônio vencer, seus nós terão os pesos atualizados recebendo uma porção dos pesos dos demais nós. Em uma regra padrão de *aprendizagem competitiva* tem-se:

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}) & \textbf{se o neurônio } k \textbf{ vencer a competição} \\ \mathbf{0} & \textbf{se o neurônio } k \textbf{ perder a competição} \end{cases} \quad (20)$$

Onde $\boldsymbol{\eta}$ é a taxa de aprendizagem. Esta regra fará com que o neurônio vencedor k tenha seu vetor de pesos $\boldsymbol{w_k}$ ajustado para um padrão de entrada x.

2.5.4 Aprendizagem de Boltzman

Esta regra tem seu nome em homenagem a Ludwing Boltzman, pois deriva de idéias originadas da mecânica estatística. Em uma *máquina de Boltzman* (denominação da rede que possui aprendizagem pela regra de Boltzman) os neurônios funcionam de forma binária e são estruturados de forma recorrente. É caracterizado por uma função de energia E definida por:

$$\varepsilon = -\frac{1}{2}\sum_j \sum_{\substack{k \\ j \neq k}} w_{kj} x_k y_j \quad (21)$$

Os neurônios podem estar em um estado "ligado" representado por +1 ou "desligado" representado por -1. Na fórmula apresentada, $\boldsymbol{x_j}$ é o estado do neurônio *j* e $\boldsymbol{w_{kj}}$ é o peso conectando o neurônio *j* ao neurônio *k*. A máquina não possui realimentação. A máquina troca o estado de um neurônio k, ao acaso, de $\boldsymbol{x_k}$ para $-\boldsymbol{x_k}$, por exemplo, a uma temperatura *T* com a seguinte probabilidade.

$$P_{(x_k \to -x_k)} = \frac{1}{1 + \exp\left(-\frac{\Delta E_k}{T}\right)} \quad (22)$$

Onde $\Delta \boldsymbol{E_k}$ é a variação de energia resultante da troca de estados. *T* não é uma temperatura física, mas sim uma representação. Pode-se dizer que ao aplicar esta regra repetidamente a máquina atingirá o seu *equilíbrio térmico*.

Considerando duas situações, uma onde os neurônios possuem estados específicos determinados pelo ambiente e outra onde os neurônios estão livres para realizar operações tem-se a as correlações $\boldsymbol{\rho_{kj}^{+}}$ e $\boldsymbol{\rho_{kj}^{-}}$ respectivamente. Estas correspondem às médias de todos os estados possíveis quando ela está em equilíbrio térmico. Então a variação $\Delta \boldsymbol{w_{kj}}$ é definida por

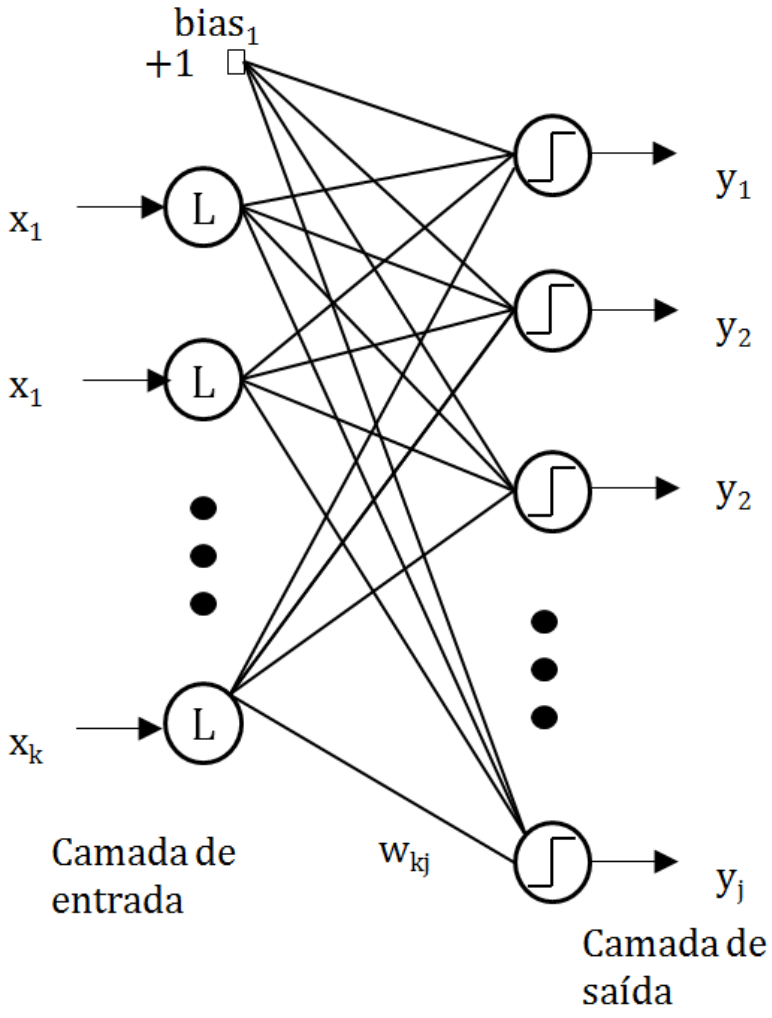
$$\Delta w_{kj} = \eta(\rho_{kj}^{+} - \rho_{kj}^{-}), j \neq k \quad (23)$$

Esta variação é aplicada ao peso sináptico $\boldsymbol{w_{kj}}$ do neurônio j para o neurônio k, onde $\boldsymbol{\eta}$ é a taxa de aprendizagem.

2.6 O *Perceptron* de camada única e regra Delta

Proposto em 1958 por Frank Rosenblatt (ROSEMBLATT, 1958), o modelo do *Perceptron* de camada única de pesos consiste de uma camada de entradas binárias e uma camada de saídas também binárias. Sem camadas escondidas e, portanto, com apenas uma camada de pesos modificáveis. As unidades de saída de saída utilizam a função de limiar ou degrau unitário {0,1}, conforme demonstrado na ilustração da Figura 26:

Figura 26 – *Perceptron* de camada única



Fonte: elaborado pelo autor

Este modelo é uma adaptação do modelo original criado por Rosenblatt (ROSEMBLAT, 1962) para reconhecimento visual de padrões com 3 tipos de unidades representando a retina e as interconexões para as unidades de saída. Pode-se observar que, apesar de ser chamado de "camada única" é comum também denominar o conjunto de entradas de "camada de entrada".

O aprendizado supervisionado do *Perceptron* foi proposto da seguinte forma:

1º) Aplicação de um padrão de entrada e cálculo da saída y;

2º) Ida ao 4º passo caso a saída esteja correta;

3º) Se a saída for incorreta e igual a -1, adição do valor de entrada ao seu respectivo peso $\Delta \boldsymbol{w_{ij}} = \boldsymbol{x_j}$; e se for igual a +1, subtrair o valor, $\Delta \boldsymbol{w_{kj}} = -\boldsymbol{x_j}$;

4º) Seleção de outro padrão de entrada e retorno ao passo 1.

Assim o 3º passo pode se expresso como:

$$\Delta w_{ij} = \frac{1}{2}[D_i - y_i]x_j \tag{24}$$

Segundo Rosenblatt haverá uma solução, após um número finito de iterações, desde que exista uma matriz W que permita a correta classificação dos padrões de treinamento (ROSEMBLAT, 1962). Esta afirmação é também denominada de *Teorema da Convergência do Perceptron*.

Considerando um *Perceptron* de 1 camada com apenas 1 saída, esta unidade dividirá o espaço das entradas em 2 regiões (uma "alta" e outra "baixa", 0 ou 1). Considera-se que estas regiões estão separadas por um *hiperplano* que é uma linha para o caso de duas entradas. A posição deste *hiperplano* é definida pelos pesos e bias que chegam à unidade de saída sendo a equação que representa este *hiperplano* representada por

$$\sum_{j=1}^{p} w_{ij}x_j + bias_i = 0 \tag{25}$$

Foi demonstrado que o *Perceptron* de 1 camada poderá resolver apenas problemas *linearmente separáveis*(MINSKY e PAPERT, 1969). Peretto (PERETTO, 1992), também demonstra que, conforme o número de entradas aumenta, o conjunto de funções linearmente separáveis se reduz a zero.

Esses problemas poderiam ser resolvidos com a rede *Perceptron* de Múltiplas Camadas (abordada no próximo item) onde, com camadas escondidas é possível expandir as superfícies de decisão como uma combinação de vários hiperplanos.

A regra Delta (WIDROW e HOFF, 1960) é um método de aprendizagem baseado no sinal de erro e, portanto, com supervisão forte. Inicialmente foi proposto utilizando a Unidade de Limiar (0 , 1). Esta regra propõe que, para cada par entrada/saída, os pesos sejam ajustados diminuindo-se os quadrados do erro da saída. Este erro é definido como

$$E^{pat} = \sum_{i=1}^{p} E_i^{pat} = \sum_{i=1}^{p} \frac{1}{2}[D_i - y_i]^2 \tag{26}$$

Por isso é também conhecido como método do *Mínimo Quadrado Médio* (LMS - Last Mean Square). Este procedimento usa a direção do gradiente executado em cada iteração. Aplicando esta regra a cada conjunto entrada/saída, o erro médio $\boldsymbol{E^{AV}}$ deve diminuir sendo

$$E^{AV} = \frac{1}{M} \sum_{pat=1}^{M} E^{pat} \tag{27}$$

E consequentemente tem-se:

$$\Delta w_{ij} = -\eta \frac{\partial E^{pat}}{\partial w_{ij}} \tag{28}$$

$$= -\eta \frac{\partial E^{pat}}{\partial y_i} \frac{\partial y_i}{\partial w_{ij}}$$

$$= \eta [D_i - y_i] \frac{\partial y_i}{\partial w_{ij}}$$

Porém, se for utilizada a função de ativação não contínua esta não será diferenciável. Por isso Widrow e Hoff (WIDROW e HOFF, 1960) propuseram que, no treinamento, fosse utilizada uma função linear do tipo Y = E X + bias. Assim o processo de aprendizagem se torna um pouco diferente e as atualizações dos pesos ocorrem mesmo quando a saída está próxima do valor desejado e não apenas quando há um erro grande. Também devem ser utilizadas entradas bipolares (-1, +1) já que para entradas do tipo (0, 1), quando o a entrada é 0, os pesos correspondentes não se alteram.

Assim o processo para o treinamento pode ser dividido nas seguintes etapas:

a) Inicialização da matriz de pesos W com pequenos números aleatórios;

b) Seleção dos conjuntos entrada/saída para o treinamento;

c) Cálculo da saída da rede Y = W X + bias;

d) Ajuste da matriz de pesos e bias com

$$\Delta w_{ij} = \eta[D_i - y_i]x_j \quad (29)$$

$$\Delta bias_i = \eta[D_i - y_i] \quad (30)$$

e) Repetir desde o item b) até que o erro de saída seja suficientemente pequeno;

Se for considerado um caso onde n é suficientemente pequeno, os padrões de treinamento são apresentados com a mesma probabilidade, com *p* padrões de entrada e *p* unidades de entrada e os padrões formarem um conjunto linearmente independente, a regra Delta fará com que a matriz de pesos $\boldsymbol{W}$ convirja para uma solução ótima $\boldsymbol{W}^*$ (NASCIMENTO e YONEYAMA, 2008), onde

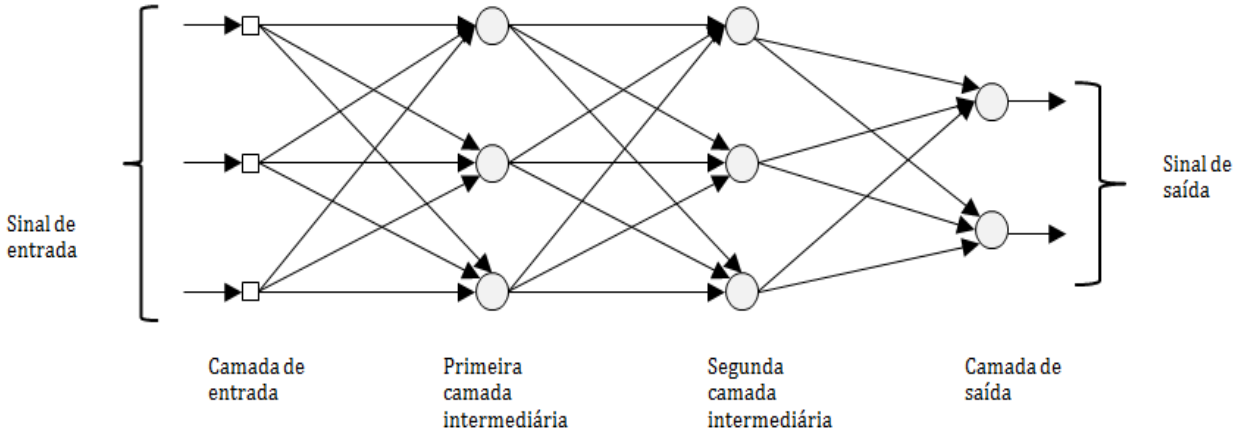
$$W^* = [D^1 D^2 \dots D^P][X^1 X^2 \dots X^P]^{-1} \quad (31)$$

2.7 O *Perceptron* de múltiplas camadas e o algoritmo de *retropropagação*

O *Perceptron* de múltiplas camadas (MLP - Multi Layer Perceptron) possui camadas escondidas que tornam possível a recodificação dos padrões de entrada. O número de unidades escondidas pode auxiliar a encontrar novas representações onde seus vetores de saída são linearmente separáveis, mesmo que as entradas não sejam.

Para o modelo *Perceptron* de múltiplas camadas é necessário pelo menos 1 camada escondida, com função de ativação linear. As unidades de saída podem ter função de ativação Linear ou não-linear (função de limiar, por exemplo). A seguir na Figura 27 um modelo de exemplo para esta rede:

Figura 27 – *Perceptron* de Múltiplas Camadas (MLP)



Fonte: adaptado de HAYKIN, 2001

Na Figura 27 tem-se um modelo de rede com duas camadas escondidas e *totalmente conectadas,* ou seja, qualquer neurônio está conectado a todos os neurônios/sós da camada anterior. O fluxo do sinal se dá da esquerda para a direita, de camada em camada.

Com as camadas escondidas ampliam-se as possibilidades de implementação de superfícies de decisão mais complexas, ou seja, a rede poderá aumentar a sua capacidade de representação. Entretanto, o aprendizado torna-se mais difícil, já que o número de pesos a serem ajustados aumenta e o método de aprendizagem terá que decidir quais aspectos da rede devem ser considerados no treinamento.

O algoritmo de *retropropagação* (*back-propagation*) foi desenvolvido por várias pessoas independente em diversas áreas. Werbos (publicado posteriormente (WERBOS, 1994)) e Parker (PARKER, 1987), por exemplo. Em 1986, Rumelhart, Hinton e Williams (RUMELHART, HINTON e WILLIAMS, 1986) redescobriram e popularizaram o algoritmo.

Seguindo o mesmo princípio da regra Delta, o algoritmo de *retropropagação* minimiza a soma dos quadrados dos erros de saída, através da média em relação ao conjunto do treinamento, baseando-se na direção do gradiente. Também é chamado de *Regra Delta Generalizada*, por se tratar da aplicação da regra Delta, mas considerando todos os neurônios, inclusive das camadas intermediárias.

Sendo o sinal de erro representado por

$$e_j(n) = d_j(n) - y_j(n), \tag{32}$$

Onde $\boldsymbol{e_j(n)}$ é o erro do neurônio *j*, na iteração n.

O valor instantâneo $\boldsymbol{\mathcal{E}(n)}$ de energia total do erro será a somatória do valor de erro instantâneo dos neurônios dado por

$$\mathcal{E}(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \tag{33}$$

Onde *C* representa todos os neurônios de saída da rede. Considerando como *N* sendo o número total de exemplos do conjunto de treinamento, a *energia média* do erro quadrado será

$$\mathcal{E}_{med} = \frac{1}{N}\sum_{n=1}^{N} \mathcal{E}(n) \tag{34}$$

O objetivo é ajustar os parâmetros livres da rede para minimizar o $\boldsymbol{\mathcal{E}_{med}}$.

O sinal de saída do neurônio é dado por

$$y_i(n) = \varphi_j\left(v_j(n)\right) \tag{35}$$

Assim, o algoritmo de *retropropagação*, similarmente ao algoritmo do mínimo quadrado médio (LMS), aplicará na correção ao peso sináptico $\boldsymbol{w_{ji}(n)}$ que será proporcional à derivada parcial $\boldsymbol{\partial\mathcal{E}(n)/\partial w_{ji}(n)}$. O gradiente pode ser expresso por

$$\frac{\partial\mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial\mathcal{E}(n)}{\partial e_j(n)}\frac{\partial e(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}\frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{36}$$

Diferenciando ambos os lados da equação 36 em relação a $e_j(n)$, obtém-se

$$\frac{\partial\mathcal{E}(n)}{\partial e_j(n)} = e_j(n) \tag{37}$$

Diferenciando ambos os lados da equação 37 em relação a $y_j(n)$, obtém-se

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{38}$$

A seguir, diferenciando os lados da equação 38 em relação a $v_j(n)$, obtém-se

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'\left(v_j(n)\right) \tag{39}$$

Onde o uso da apóstrofe (no lado direito) significa a diferenciação em relação ao argumento. Finalmente, diferenciar a equação 39 em relação a $w_{ji}(n)$ produz

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{40}$$

O uso das equações de 37 a 40 em 36 produz

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -e(n)\varphi_j'\left(v_j(n)\right) y_i(n) \tag{41}$$

A correção $\Delta w_{ji}(n)$ aplicada a $w_{ji}(n)$ é definida pela *regra delta*:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} \tag{42}$$

Onde $\boldsymbol{\eta}$ é o *parâmetro da taxa de aprendizagem* do algoritmo de *retroprogagaçã*o. O uso do sinal negativo na equação 42 indica *a descida do gradiente* no espaço de pesos (i.e., busca uma direção para a mudança de peso que reduza o valor de $\varepsilon(n)$). Correspondente, o uso da equação 41 em 42 produz

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{43}$$

Onde o *gradiente local* δ_j(n) é definido por

$$\begin{aligned} \delta_j(n) &= -\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} \\ &= -\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \\ &= e_j(n)\varphi_j'(v_j(n)) \end{aligned} \tag{44}$$

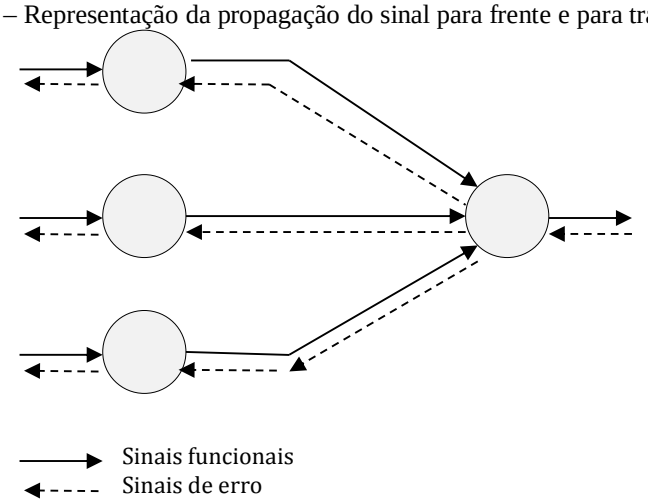
O gradiente local aponta para as modificações necessárias nos pesos sinápticos. De acordo com a equação 44, o gradiente local $\delta_j(n)$ para o neurônio de saída j é igual ao

produto do sinal de erro $\boldsymbol{e_j(n)}$ correspondente para aquele neurônio para derivada $\varphi_j'(v_j(n))$ da função de ativação associada.

Das equações 43 e 44 nota-se a importância no cálculo de ajuste de peso $\Delta w_{ji}(n)$ do sinal de erro $e_j(n)$ na saída do neurônio j.

Entende-se, portanto, que os dados que alimentam o cálculo encontrado se dão da saída para a entrada dos valores. A Figura 28 a seguir apresenta uma parte da rede com dois tipos de sinais envolvidos no processo de aprendizagem, o *sinal funcional* que se propaga para frente e o *sinal de erro* que se propaga para trás (HAYKIN, 2001).

Figura 28 – Representação da propagação do sinal para frente e para trás



Fonte: HAYKIN, 2001

No caso em que o neurônio j é um nó de Saída, tem-se:

$$\delta_j(n) = e_j(n)\varphi_j'\left(v_j(n)\right) \tag{45}$$

Quando o neurônio j está localizado em uma camada oculta da rede, tem-se:

$$\delta_j(n) = \frac{\partial \mathcal{E}(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \tag{46}$$

$$= \frac{\partial \mathcal{E}(n)}{\partial y_j(n)}\varphi_j'(v_j(n)), \text{ o neurônio } j \text{ é oculto}$$

Para calcular a derivada parcial $\boldsymbol{\partial \mathcal{E}(n)/\,\partial y_j(n)}$, pode-se proceder como segue

$$\mathcal{E}(n) = \frac{1}{2}\sum_{k\epsilon C} e_k^2(n), \text{ o neurônio k é um nó de saída} \tag{47}$$

Com o índice k utilizado no lugar do índice j. Diferenciando a equação anterior em relação ao sinal funcional y_j(n), obtém-se

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k^2(n)}{\partial y_j(n)} \tag{48}$$

A seguir utiliza-se a regra da cadeia para a derivada parcial ∂e_k(n)/ ∂y_j(n) e reescreve-se a equação anterior da forma equivalente

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{49}$$

Para o caso em que o neuronio *k* é saída, tem-se:

$$\frac{\partial e_k(n)}{\partial d_k(n)} = -\varphi_k'\big(v_k(n)\big) \tag{50}$$

E seu campo local induzido é

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n)\, y_j(n) \tag{51}$$

Onde *m* é o número total de entradas (excluindo o bias) aplicadas ao neuronio *k.* Diferenciando a equação anterior em relação a y_j(n) tem-se

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{52}$$

Para obter a derivada parcial desejada, tem-se

$$\begin{aligned} \frac{\partial \mathcal{E}(n)}{\partial y_j(n)} &= -\sum_k e_k(n) \varphi_k'\big(v_k(n)\big) w_{kj}(n) \\ &= -\sum_k \delta_k(n)_k w_{kj}(n) \end{aligned} \tag{53}$$

Em seguida obtém-se a fórmula de *retropropagação* para o gradiente local $\boldsymbol{\partial_j(n)}$ como descrito:

$$\delta_j(n) = \varphi_j'\left(v_j(n)\right) \sum_k \delta_k(n)_k w_{kj}(n) \tag{54}$$

Resumindo a fórmula para o algoritmo de *retropropagação*, a correção $\Delta \boldsymbol{w_{ji}}$(n) aplicada ao peso sináptico conectando o neurônio *i* ao neurônio *j* é definida pela regra delta:

$$\begin{pmatrix} correção \\ de\ peso \\ \Delta w_{ji}(\mathrm{n}) \end{pmatrix} = \begin{pmatrix} Parâmetro\ da \\ taxa\ de\ aprendizagem \\ \eta \end{pmatrix} . \begin{pmatrix} Gradiente \\ local \\ \delta_j(n) \end{pmatrix} . \begin{pmatrix} sinal\ de\ entrada \\ do\ neurônio\ j \\ y_i(n) \end{pmatrix}$$

Uma questão importante e uma das modificações em relação à regra Delta é que, para que as funções de ativação sejam diferenciáveis deve-se substituir as funções *threshold* ou *limiar* por funções contínuas e suaves, possibilitando a busca pela descida do gradiente mesmo nas camadas escondidas. Uma função padrão para este fim é a chamada de *S-shaped* ou *sigmoidal*, que é não-linear e continuamente diferenciável. A função pode ser expressa por,

$$\varphi_j\left(v_j(n)\right) = \frac{1}{1 + \exp\left(-a v_j(n)\right)} \; a > 0 \; \mathbf{e} \; -\infty < v_j(n) < \infty \tag{55}$$

Onde $\boldsymbol{v_j(n)}$ é o vetor local induzido do neurônio *j*.

2.7.1 A taxa de aprendizagem

Quanto menor o valor da taxa de aprendizagem $\boldsymbol{\eta}$, menores serão os ajustes aplicados aos pesos sinápticos, o que implicará em uma atualização mais suave ao longo das iterações, o que pode ser uma vantagem em relação à diminuição do erro médio quadrado. Porém o sistema ficará mais lento. Ao aumentar o parâmetro de aprendizagem, a rede pode se tornar instável (oscilatória) e para que isto não ocorra é utilizado um método utilizando um termo de *momento* (RUMELHART, HINTON e WILLIAMS, 1986), que é demonstrado como

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_j(n) \tag{56}$$

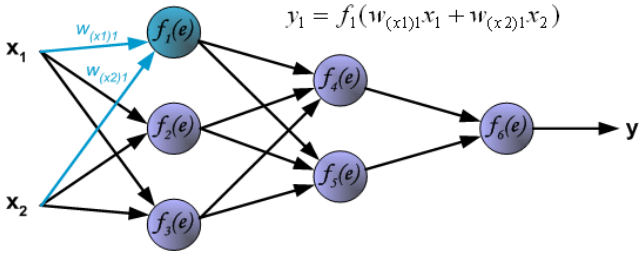
Onde $\boldsymbol{\alpha}$, é denominado *constante de momento* e usualmente um número positivo. Quando $\boldsymbol{\alpha}$ for igual a zero, a equação se tornará a *regra delta generalizada*.

2.8 Etapas da retropropagação

O algoritmo de *retropropagação*, utilizado no *Perceptron* de múltiplas camadas se baseia no método de descida do gradiente em função da diminuição do erro médio $\boldsymbol{\mathcal{E}_{med}}$ e necessita que as suas funções de ativação sejam diferenciáveis. No algoritmo de *retropropagação* identifica-se a existência de três momentos na implementação do algoritmo:

1º) onde propaga-se o sinal para frente calculando os campos locais induzidos $\boldsymbol{v_j(n)}$ e as saídas $\boldsymbol{y_j(n)}$ sem alteração dos pesos sinápticos conforme a Figura 29:

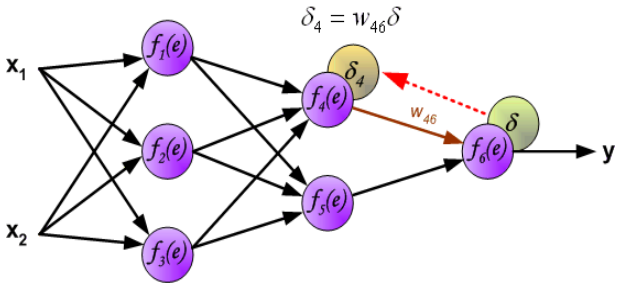
Figura 29 – Representação do cálculo do sinal de saída na propagação para frente



Fonte: Site da Universidade de Ciência e Tecnologia AGH da Polônia [1]

2º) propaga-se o sinal calculando o gradiente local de cada neurônio passando os sinais de erro para trás para conforme a Figura 30:

Figura 30 – Representação do cálculo do erro local na fase de propagação para trás

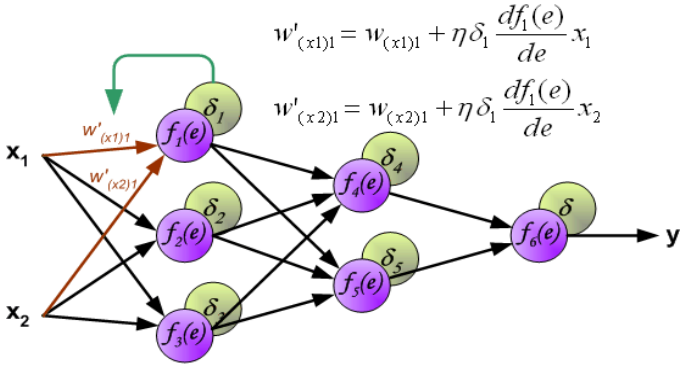


Fonte: Site da Universidade de Ciência e Tecnologia AGH da Polônia [1]

[1] Disponível em http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html. Acesso em jul. 2012

3º) Em seguida os pesos sinápticos devem ser alterados conforme a Figura 31:

Figura 31 – Correção dos pesos sinápticos no algoritmo de retropropagação



Fonte: Site da Universidade de Ciência e Tecnologia AGH da Polônia[2]

Para o ajuste nos pesos dos neurônios de saída, a correção $\Delta \boldsymbol{w_{ji}(n)}$ aplicada ao neurônio utiliza-se a equação 43, sendo que o gradiente $\delta_j(n)$ será a equação 45 para o neurônio de saída e a equação 54 para o neurônio intermediário.

[2] Disponível em http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html. Acesso em jul. 2012

3 FERRAMENTA COMPUTACIONAL PARA MODELOS DE RNA'S

Analisando os diversos tipos de rede considerou-se a implementação do algoritmo as redes *Perceptron* de Múltiplas Camadas (MLP's - *Multi Layer Perceptron*). Estas redes têm sido utilizadas para uma grande variedade de aplicações visto que permitem uma maior capacidade de representação em relação aos resultados desejados, conforme visto no capítulo 2.

As redes MLP's necessitam de um algoritmo capaz de ajustar os pesos das camadas intermediárias e por isso foi utilizado o algoritmo de treinamento de *retropropagação* (*back-propagation*). Este algoritmo, que é uma expansão da regra delta generalizada (WIDROW e HOFF, 1960), utilizará os valores de erro local das camadas intermediárias, e que, conforme visto no capítulo 2 é baseado no método de descida do gradiente do erro.

Portanto o algoritmo utiliza uma regra de aprendizagem por correção e deve possuir funções de ativação diferenciáveis e para o treinamento devem ser fornecidos os valores de entrada e de saída desejados.

3.1 Resultado esperado

O produto esperado ao final da execução do algoritmo é que, após inserir os dados da rede, como seus parâmetros e conjunto entrada/saída, se obtenha um modelo que represente as características desejadas. Este modelo poderá ser utilizado em qualquer sistema para que reproduza das características impostas no treinamento.

A princípio não são testados, na fase de validação, os modelos que, ao serem inseridos em outros sistemas, precisarem de uma realimentação e um ajuste contínuo do pesos sinápticos. Porém existem diversas aplicações em que é necessário não apenas o modelo final obtido, mas sim que todo o algoritmo de aprendizado também faça parte dele. Assim o algoritmo desenvolvido também é aproveitado para ser utilizado em sistemas deste tipo, que necessitam de uma maior interação com os cálculos de aprendizagem e não apenas o modelo final, conforme visto na seção 2.3.

3.2 Etapas do algoritmo

Entre as etapas para a obtenção do modelo tem-se:

3.2.1 A fase de propagação para frente

O cálculo a partir da entrada e propagado até a geração da saída possui os seguintes passos:

a) Multiplicação das entradas pelos pesos sinápticos nas conexões iniciais;
b) Somatório dos resultados das multiplicações;
c) Função de ativação do resultado;
d) Multiplicação pelos pesos sinápticos da camada adiante (se houver), somatório e função de ativação;
d) Repetição destes cálculos até a última camada. Geração do sinal de saída.

3.2.2 Propagação para trás

Cálculo que parte da saída obtida até a primeira camada com os seguintes passos:

a) Cálculo do erro representado pela diferença entre o valor desejado e a saída;
b) Multiplicação do erro pelo peso no sentido em direção à penúltima camada;;
c) Somatório destes novos valores nos neurônios da camada anterior (caso os valores não sejam originados da camada de saída) produzindo novos gradientes;
d) multiplicação destes novos erros pelos pesos em direção à camada anterior e assim por diante até a os neurônios de entrada

3.2.3 Ajuste dos pesos sinápticos

Em seguida serão calculados os novos pesos da rede:

a)Calculam-se os novos pesos das conexões de entrada somando-se o peso da interação inicial ao produto entre a taxa de aprendizagem,o gradiente da conexão

correspondente e a derivada da função de ativação dos neurônios da primeira camada em relação ao potencial de ativação;

b) Repete-se esta operação ajustando-se todos os pesos das conexões adiante.

Estes passos constituem uma iteração ou *época*. São realizadas quantas épocas forem necessárias até que se diminua o erro ε_{med} , apresentado na equação 34, a um valor mínimo desejado.

4 CONSTRUÇÃO DO ALGORITMO PARA RNA'S

Com base nos conceitos estudados neste trabalho foi implementado o sistema para configuração e treinamento da rede neural baseada no algoritmo de retropropagação. De acordo com as etapas desenvolvidas no capítulo 3 construiu-se o código para satisfazer o algoritmo proposto.

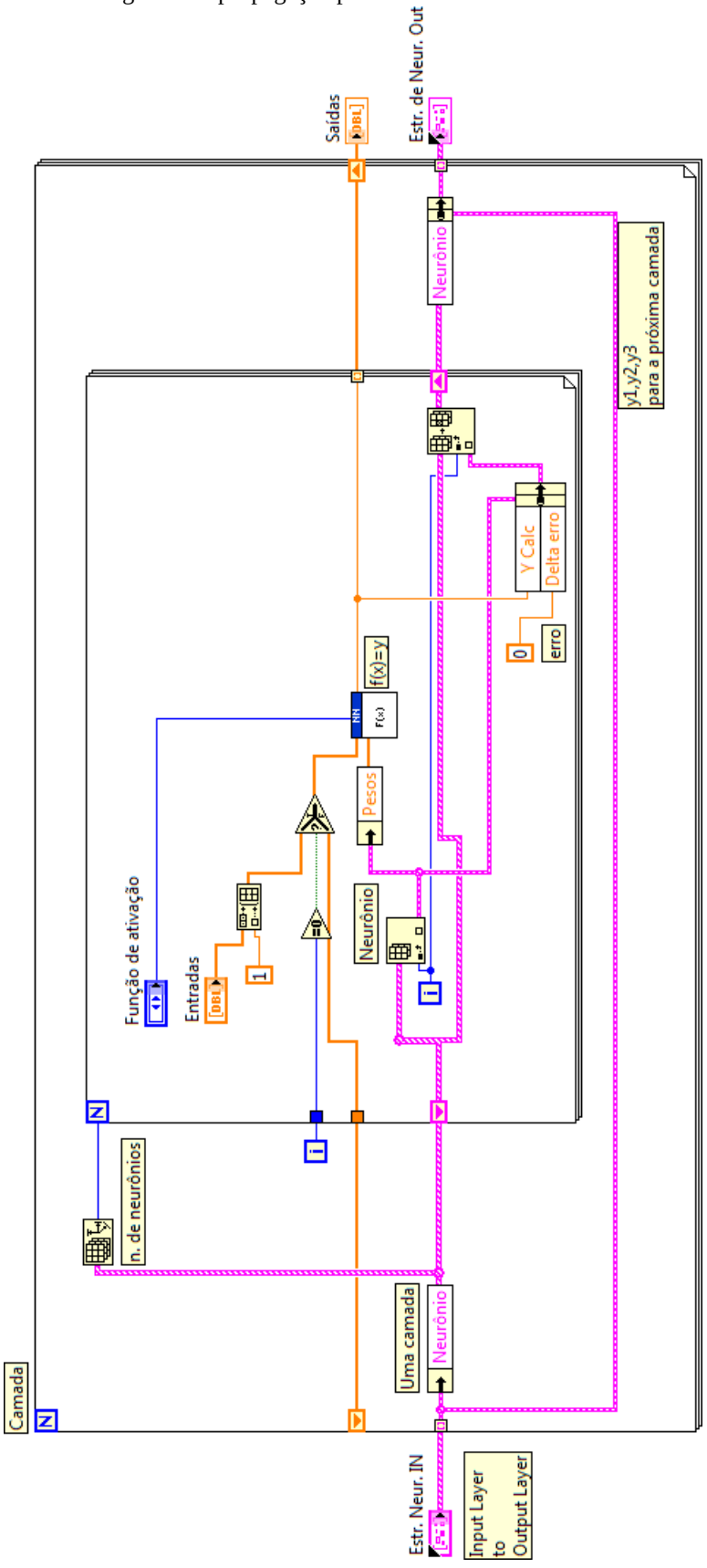
Foi escolhida para a implementação a linguagem gráfica LabVIEW. Esta linguagem se destaca por possuir uma fácil visualização do código como um todo. Possui recursos que facilitam identificar o fluxo dos dados nos algoritmos e corrigir possíveis erros durante o desenvolvimento. É uma linguagem de programação com código basicamente representado por blocos e linhas, constituindo um paradigma de código diferente dos habituais estruturados em texto.

A seguir serão apresentadas as principais etapas para a implementação do algoritmo na linguagem gráfica de programação.

4.1 Propagação adiante e cálculo da saída

A seguir, na Figura 32, é demonstrado o trecho de código que representa a fase de multiplicação dos pesos sinápticos pelas entradas e somatório e função de ativação executados dentro do *subvi*.

Figura 32 – Código com a propagação para frente do sinal em uma rede MLP



Fonte: elaborado pelo autor

Onde o laço *for* representado pelo retângulo maior realiza a repetição dos cálculos para cada camada presente na rede. O laço *for* menor realiza os cálculos em cada neurônio da mesma camada e representa a equação

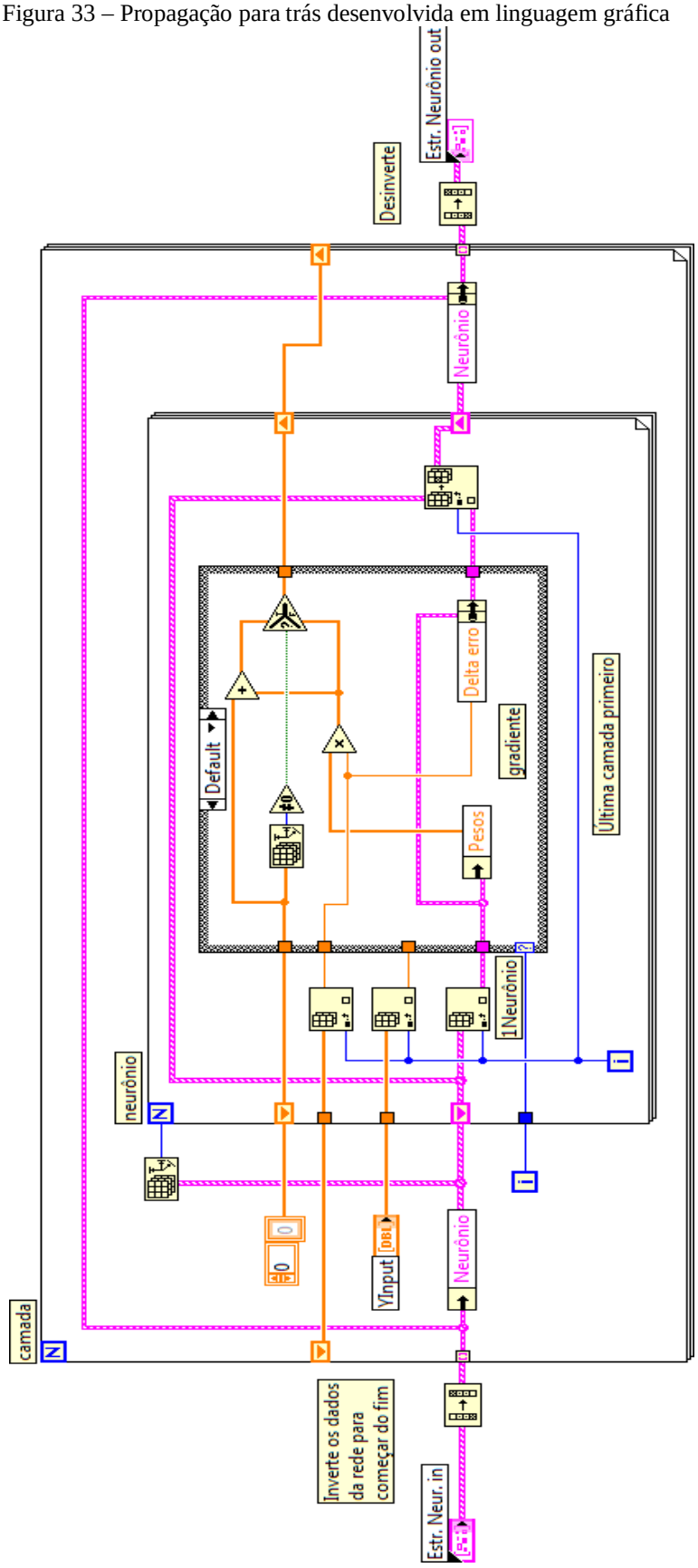
$$y_k = \varphi\left(\theta_k + \sum_{i=1}^{p} w_{ki} x_i\right) \tag{57}$$

Em que $\boldsymbol{y_k}$ é a saída , $\boldsymbol{\theta_k}$ o *bias*, $\boldsymbol{w_{ki}}$ os pesos da camada $\boldsymbol{k}$ em direção à camada $\boldsymbol{i}$ e $\boldsymbol{x_i}$ o valor de entrada do neurônio.

4.2 Propagação para trás

A Figura 33 a seguir apresenta a implementação do cálculo no sentido inverso da rede, começando pelo erro em relação ao valor de saída desejado e em seguida os gradientes locais de erro em cada camada.

Figura 33 – Propagação para trás desenvolvida em linguagem gráfica



Fonte: elaborado pelo autor

Onde estão sendo realizados os cálculos representados por

$$\delta_j(n) = e_j(n)\varphi'_j\left(v_j(n)\right) \tag{58}$$

Quando o neurônio j é um neurônio de saída, $e_j(n)$ é o erro da rede e φ'_j é a derivada da função de ativação sobre $v_j(n)$. Este último é o potencial local induzido, que é a soma ponderada das entradas sinápticas acrescidas do *bias*.

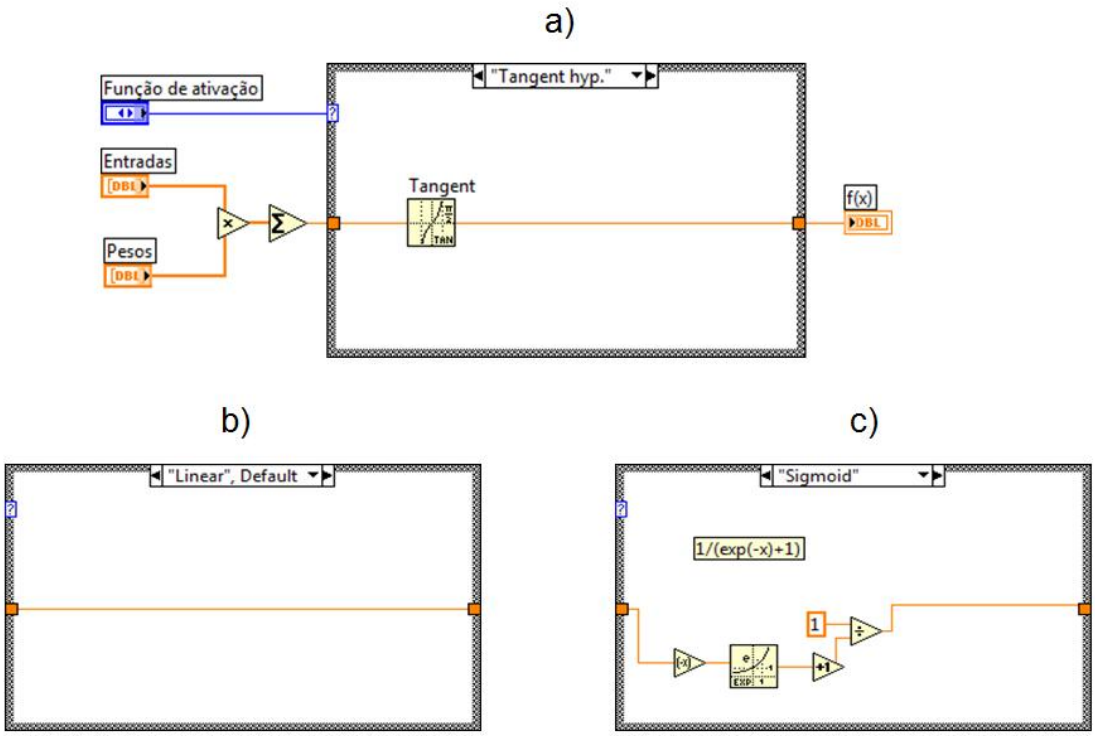
Para os casos em que o neurônio está em uma camada escondida tem-se

$$\delta_j(n) = \varphi'_j\left(v_j(n)\right)\sum_k \delta_k(n)_k w_{kj}(n) \tag{59}$$

4.2.1 Função de ativação

O algoritmo das funções de ativação está representado no código pelo seguinte bloco da Figura 34:

Figura 34 – Código com as funções de ativação: a) função tengente hiperbólica; b) função linear; c) função sigmoidal

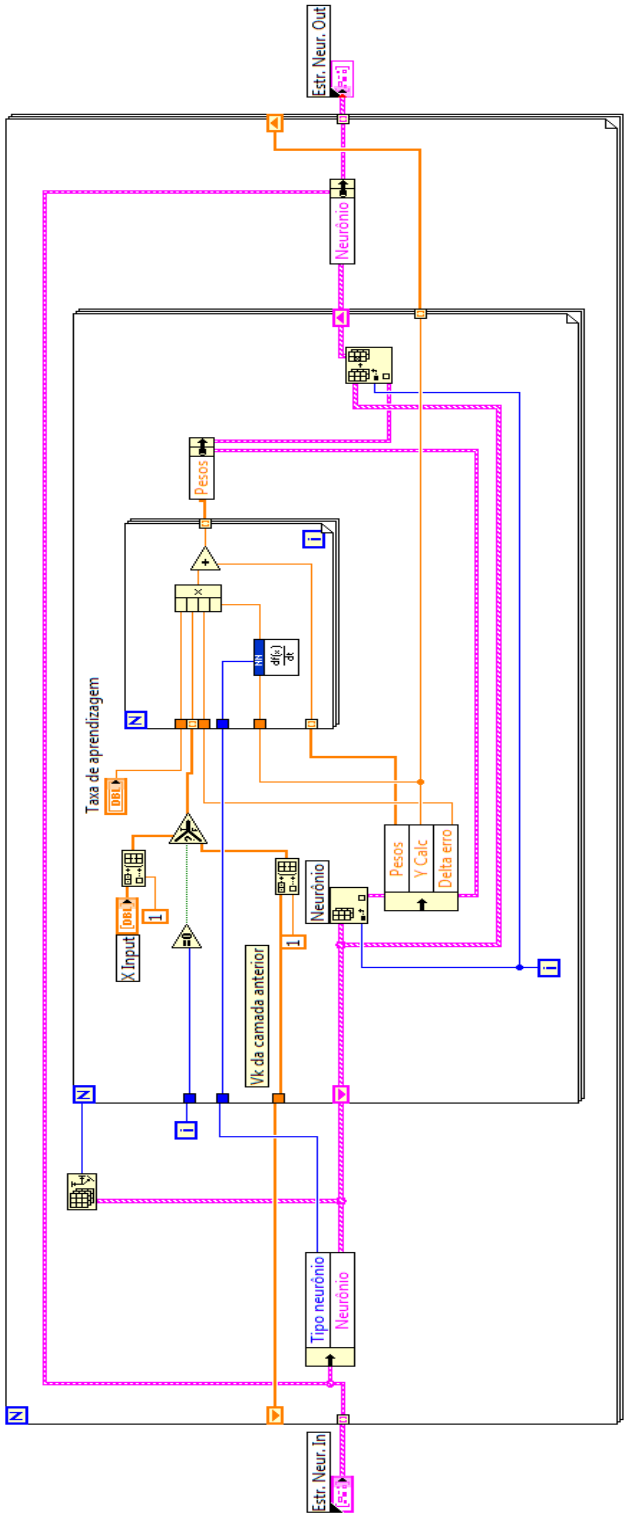


Fonte: elaborado pelo autor

4.3 Ajustes dos pesos sinápticos

Após o cálculo dos gradientes de erro, realiza-se o ajuste representado na linguagem gráfica pela Figura 35:

Figura 35 – Ajuste dos pesos sinápticos em linguagem gráfica



Fonte: elaborado pelo autor

A Figura 35 é responsável pelo cálculo de ajuste representado por

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{60}$$

Ou seja, os novos pesos sinápticos para cada nó serão

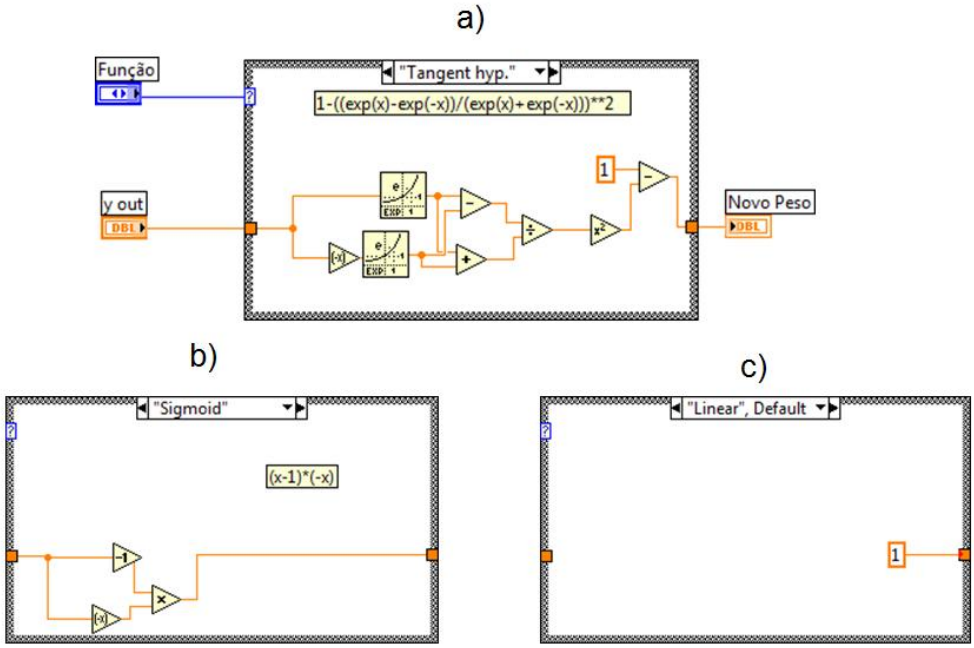
$$w_{ji}(n+1) = w_{ji}(n) \eta \delta_j(n) y_i(n) \tag{61}$$

Em que η é a taxa de aprendizagem, $\delta_j(n)$ é o gradiente local e $y_i(n)$ é o sinal de entrada do neurônio j.

4.3.1 Derivada da função de ativação

O código da derivada da função de ativação desenvolvido e é representado pelo seguinte diagrama de blocos da Figura 36:

Figura 36 – Derivada da função de ativação: a) Derivada da função tangente hiperbólica; b) Derivada da função sigmoidal; c) Derivada da função linear

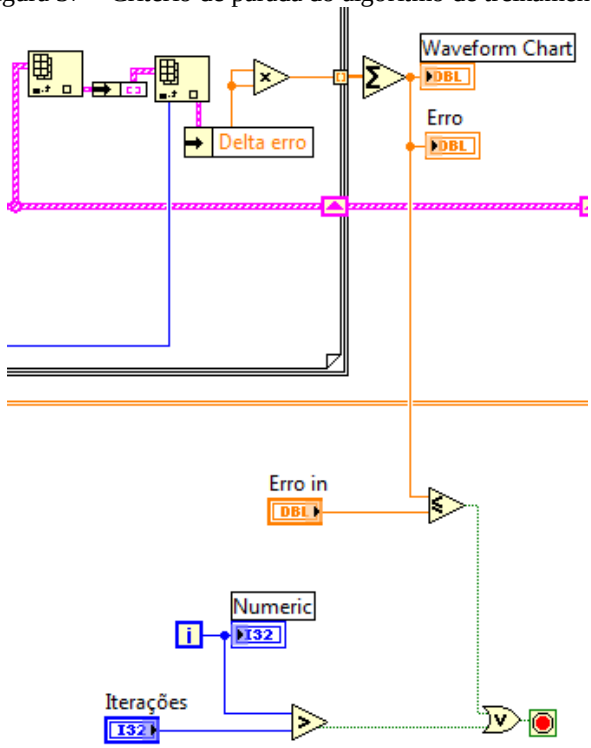


Fonte: elaborado pelo autor

4.4 Critérios de parada

Determinou-se que o algoritmo irá parar de ajustar os pesos pelos seguintes critérios: o erro quadrado médio $\mathcal{E}_{med}$ atingir um valor mínimo desejado ou o código executar um número de épocas (iteração contendo a fase adiante, para trás e ajuste dos pesos) pré-determinado. A seguir, na Figura 37, a representação em código do critério de parada.

Figura 37 – Critério de parada do algoritmo de treinamento



Fonte: elaborado pelo autor

4.5 Observações sobre o algoritmo de treinamento implementado

A divisão do código em sub-códigos durante a construção do algoritmo facilitou a visualização das etapas e, portanto a própria programação, permitindo implementar os cálculos conforme a sequência descrita no capítulo 3. O algoritmo proposto utiliza os cálculos de forma *sequencial*, ou seja, o processo de propagação para trás e para frente e ajuste de pesos é realizado para cada conjunto de entradas de uma vez dentro de uma *época*. Outra forma de realizar o treinamento seria *por lote*, onde todos os pares de entrada são apresentados à rede durante uma *época*. O treinamento sequencial possui algumas desvantagens como a dificuldade de se estabelecer uma condição teórica de convergência.

Entretanto este tipo de algoritmo é bastante utilizado por ser mais simples de se implementar, e fornecer soluções efetivas para diversos problemas. Isso porque, para amostras com redundância de padrões o algoritmo é capaz de acelerar o aprendizado.

Ao construir a ferramenta foram definidos quais parâmetros mínimos uma RNA, no caso uma rede MLP com o algoritmo *retropropagação,*são necessários para aprendizagem.

Estes parâmetros fundamentais refletem as principais características deste trabalho que se propõe a disponibilizar estas ferramentas a qualquer usuário interessado em desenvolver modelos de RNA's, no caso de redes *Perceptron* MLP's.

Os seguintes parâmetros a serem inseridos pelo usuário foram definidos:

a) Os pares de entrada e saídas desejadas;

b) O número de camadas intermediárias;

c) O número de neurônios em cada camada;

d) A função de ativação em cada camada;

e) O algoritmo de treinamento (neste caso utilizaremos apenas a *retropropagação*);

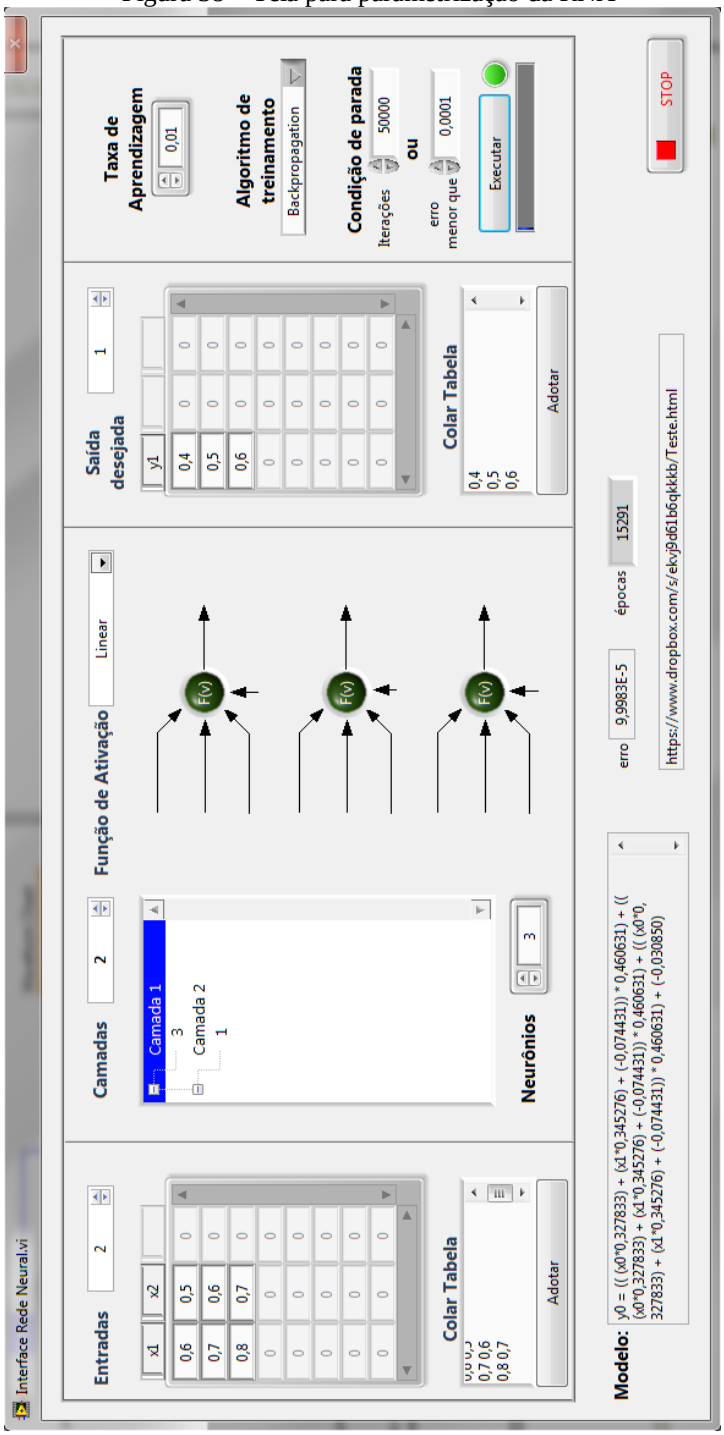
f) A taxa de aprendizagem;

g) O número de iterações (épocas) a serem executadas ou o erro quadrado médio mínimo a ser atingido durante o treinamento.

5 INTERFACE E PARAMETRIZAÇÃO DA REDE

Tendo como base os parâmetros identificados no capítulo 4, foi criada a seguinte interface conforme a Figura 38:

Figura 38 – Tela para parametrização da RNA



Fonte: elaborado pelo autor

A interface possui, em seu lado esquerdo, os campos para a inserção dos valores de entrada da rede. Ao centro é onde se define a configuração de camadas, neurônios e funções de ativação. No lado direito, os dados de saídas desejadas e em seguida a taxa de aprendizagem, número de iterações e erro mínimo. Abaixo está o campo para onde é gerado o modelo após o treinamento.

Considerou-se a criação de um ambiente com poucas "janelas" e que pudesse ser facilmente manipulado. Verificaram-se também as características dos possíveis usuários desta ferramenta. Estas características, tais como os objetivos finais do usuário, definem o tipo de design utilizado na interface. Tratando um pouco sobre design e engenharia de software temos a seguinte citação:

> Muitos designers assumem que criar interfaces fáceis de aprender podem sempre ser o objetivo final para um desenvolvedor. Fácil de aprender é um requisito importante, mas... o objetivo do design da interface depende do contexto -- quem são os usuários, o que eles pretendem fazer, e quais são seu objetivos. (COOPER, REINMAN e CRONIN, 2007)

Assim considerou-se que o usuário interessado será o profissional do ambiente acadêmico ou científico que queira realizar treinamentos com o objetivo de encontrar modelos que atenderão a necessidade de suas aplicações de reconhecimento de padrão ou controle de sistemas.

A interface possui os parâmetros descritos no item 4.5 e também permite que os dados de entrada e saída desejada sejam copiados de outras bases de dados.

Não necessariamente o modelo deve possuir apenas uma variável de saída ou uma de entrada. Deve variar de acordo com os parâmetros da configuração.

Como produto do treinamento o código disponibiliza o modelo final obtido para que o usuário o utilize em outros sistemas e um relatório do treinamento que contém as seguintes informações:

a) Número de camadas;
b) Número de neurônios em cada camada;
c) Funções de ativação utilizadas;
d) Matriz de pesos;
e) Matriz de *bias;*
f) Número de épocas;

g) Erro final atingido;

h) Gráfico do erro médio quadrado ao longo do treinamento;

i) Taxa de aprendizagem;

j) Conjunto de valores de entrada e saída;

k) Modelo final obtido.

6 TESTE DAS FERRAMENTAS

As ferramentas de desenvolvidas foram utilizadas em um sistema de controle em malha fechada. Este sistema é composto por:

a) Um hardware para aquisição e geração de sinais NI MyDAQ da National Instruments;

b) Uma ventoinha;

c) Emissor e receptor infravermelho;

d) Circuito e fios para as conexões;

e) Computador para o processamento dos sinais.

f) Microcontrolador

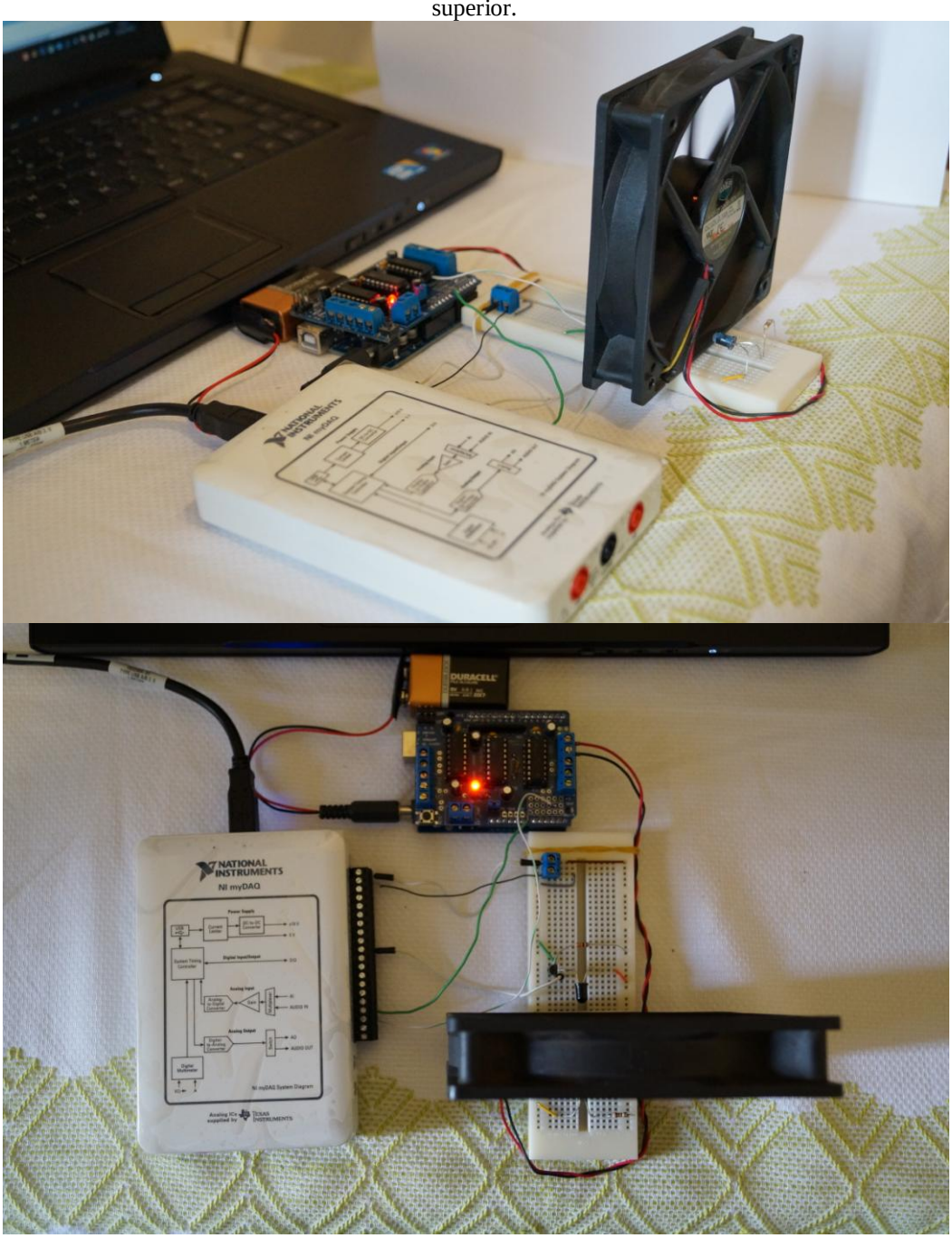
Os dados são adquiridos e gerados pelo computador através da interface USB com o hardware NI MyDAQ. Figura 39 é apresentada a imagem com os dispositivos utilizados. O software realiza a leitura da rotação contando os pulsos gerados pelo receptor infravermelho. Cada haste da ventoinha interrompe o sinal entre o emissor e o receptor infravermelho gerando assim, pulsos a cada interrupção. Estes pulsos são derivados ao longo do tempo para que se obtenha um valor de velocidade da ventoinha dado em rotações por minuto (rpm).

O software também é responsável por gerar uma saída de tensão de 0 a 5 Volts que controla a velocidade da ventoinha. Este sinal passa por um microcontrolador que o converte em um sinal do tipo *Pulse Width Modulation* (PWM), obedecendo à seguinte relação:

Saída PWM
0V -> 0%
5V -> 100%

O microcontrolador é alimentado com uma bateria de 9 volts que também é responsável por fornecer a alimentação à ventoinha.

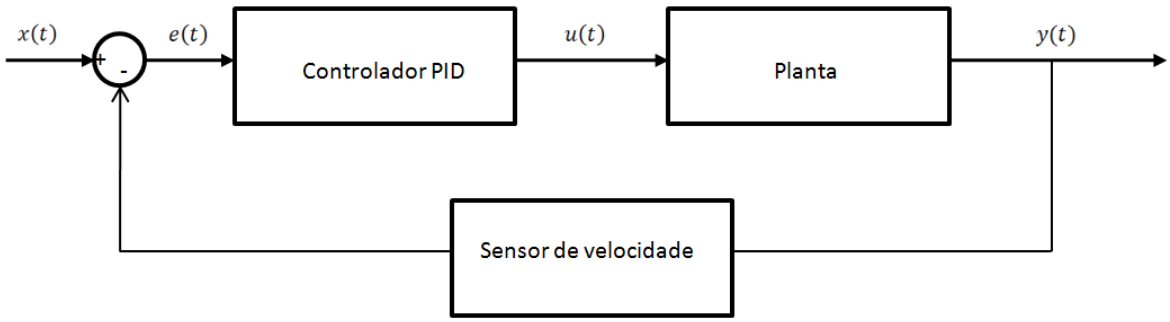
Figura 39 – Hardware e componentes para o teste de controle em malha fechada; vista em ângulo e vista superior.



Fonte: elaborado pelo autor

Inicialmente foi implementado um algoritmo de controle Proporcional, Integral e Derivativo (PID) para o controle de velocidade da ventoinha. O diagrama que representa a malha de controle é apresentado na Figura 40:

Figura 40 – Malha de controle de velocidade

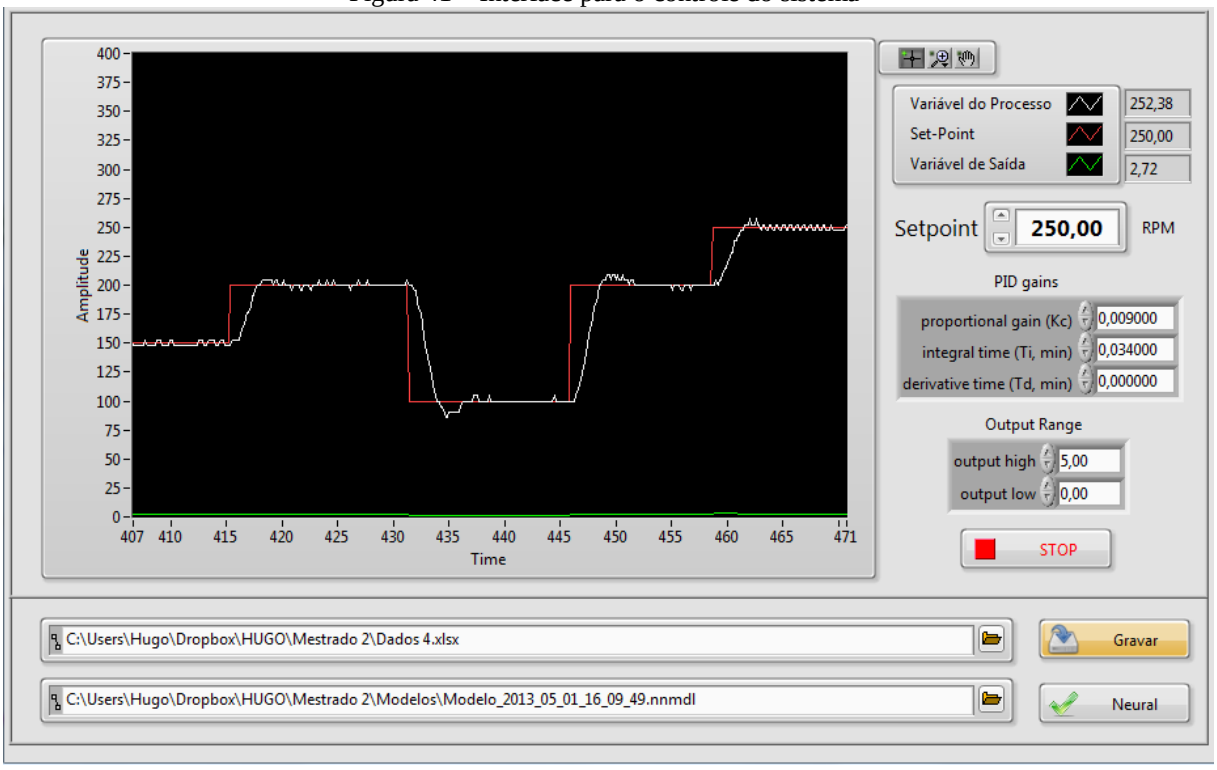


Fonte: elaborado pelo autor

Onde $x(t)$ é o sinal de entrada do sistema que é gerado através do *set-point*, $u(t)$é o sinal de saída do controlador, $y(t)$ é o sinal de resposta da planta e $e(t)$ é o erro em relação ao *set-point*.

A interface de controle, implementada para o teste em LabVIEW, e o gráfico com o comportamento do controlador são apresentados na Figura 41. A interface possui os parâmetros de ajuste Ti (tempo integral), Td (Tempo derivativo) e Kp (Ganho Proporcional). Estes parâmetros foram ajustados de forma que o sinal de resposta do sistema não tivesse sobressinal, ou seja, não ultrapassasse o limite do *set-point* durante o ajuste.

Figura 41 – Interface para o controle do sistema

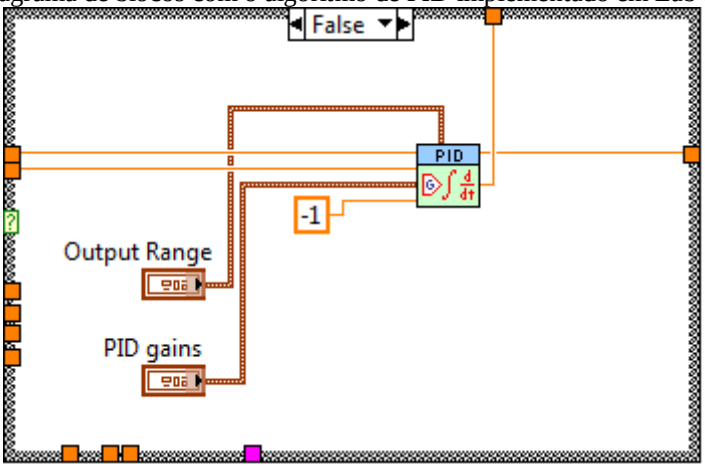


Fonte: elaborado pelo autor

O intervalo de tempo do loop de controle foi configurado para 170 milissegundos.

A seguir, na Figura 42, é apresentado o diagrama de blocos em liguagem gráfica da implementação do algoritmo de controle PID:

Figura 42 – Diagrama de blocos com o algoritmo de PID implementado em LabVIEW



Fonte: elaborado pelo autor

6.1 Gravação dos dados

Os dados de entrada e saída do sistema foram gravados durante o funcionamento no qual se variou o *set-point* (valor desejado de velocidade). Foram inseridos *set-points* de 150, 100 e 200 e 250 rpm, com intervalos de aproximadamente 10 segundos para estabilização da velocidade. O intervalo de tempo para cada gravação foi o mesmo tempo de controle do loop, 170 milissegundos.

Obteve-se uma tabela em formato Excel de aproximadamente 180 linhas com os dados:

a) *Set-point*

b) Variável de processo (rpm)

c) Erro médio quadrado

c) Valores de saída (Volts)

A tabela com os dados foi expandida para que se obtivessem os dados da *nt* com atraso de $z^{-1}, z^{-2}, z^{-3}, z^{-4}$. Ou seja, foram gerados os dados para $i(k-1), i(k-2), i(k-3), i(k-4)$, onde i representa o valor de entrada (*set-point)* e k o valor da iteração.

A seguir, na Tabela 01, são representadas os valores gravados ao se variar o *set-point* de 150 a 200 rpm onde é possível observar a mudança destes valores até a estabilização do sistema:

Tabela 1 – Valores de entrada e saída para o teste do algoritmo

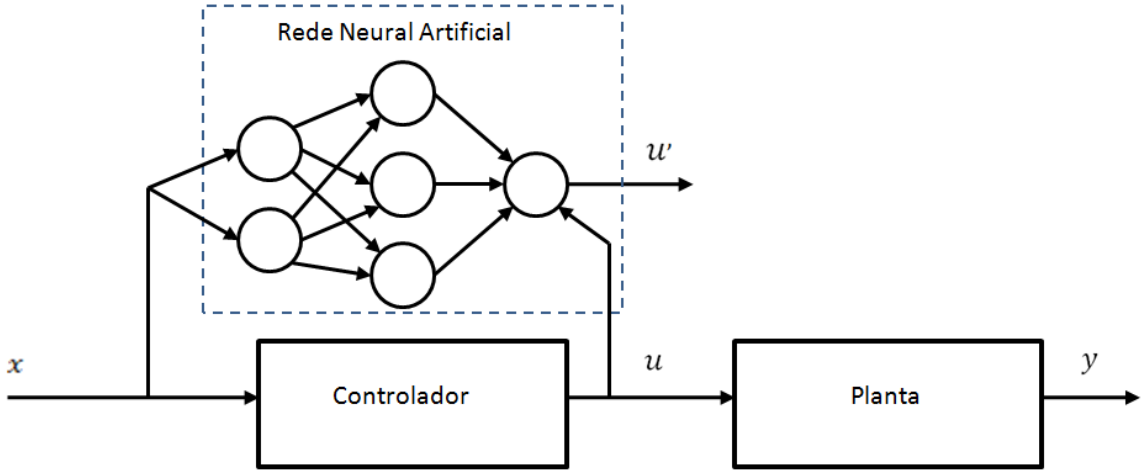
Set-Point	RPM	RPM (k-1)	RPM (k-2)	RPM (k-3)	RPM (k-4)	erro	Saída (V) (k-1)	Saída (V) (k-2)	Saída (V) (k-3)
150	152,381	152,381	147,619	147,619	147,619	-2,381	1,91	1,953	1,951
150	152,381	152,381	152,381	147,619	147,619	-2,381	1,908	1,91	1,953
150	147,619	152,381	152,381	152,381	147,619	2,381	1,907	1,908	1,91
150	147,619	147,619	152,381	152,381	152,381	2,381	1,949	1,907	1,908
150	147,619	147,619	147,619	152,381	152,381	2,381	1,951	1,949	1,907
200	147,619	147,619	147,619	147,619	152,381	52,381	1,953	1,951	1,949
200	152,381	147,619	147,619	147,619	147,619	47,619	2,425	1,953	1,951
200	152,381	152,381	147,619	147,619	147,619	47,619	2,422	2,425	1,953
200	152,381	152,381	152,381	147,619	147,619	47,619	2,46	2,422	2,425
200	152,381	152,381	152,381	152,381	147,619	47,619	2,497	2,46	2,422
200	152,381	152,381	152,381	152,381	152,381	47,619	2,535	2,497	2,46
200	157,143	152,381	152,381	152,381	152,381	42,857	2,573	2,535	2,497
200	161,905	157,143	152,381	152,381	152,381	38,095	2,566	2,573	2,535
200	166,667	161,905	157,143	152,381	152,381	33,333	2,555	2,566	2,573
200	171,429	166,667	161,905	157,143	152,381	28,571	2,541	2,555	2,566
200	176,19	171,429	166,667	161,905	157,143	23,81	2,523	2,541	2,555
200	176,19	176,19	171,429	166,667	161,905	23,81	2,501	2,523	2,541
200	180,952	176,19	176,19	171,429	166,667	19,048	2,52	2,501	2,523
200	190,476	180,952	176,19	176,19	171,429	9,524	2,494	2,52	2,501
200	190,476	190,476	180,952	176,19	176,19	9,524	2,419	2,494	2,52
200	195,238	190,476	190,476	180,952	176,19	4,762	2,427	2,419	2,494
200	200	195,238	190,476	190,476	180,952	0	2,39	2,427	2,419
200	200	200	195,238	190,476	190,476	0	2,349	2,39	2,427
200	200	200	200	195,238	190,476	0	2,349	2,349	2,39

Fonte: elaborado pelo autor

6.2 Treinamento do sistema

Os dados armazenados em arquivo foram inseridos em uma rede para encontrar um modelo que "copiasse" o comportamento do controle PID. O diagrama abaixo da Figura 43 representa o modo como a rede foi utilizada para adquirir as características do controlador:

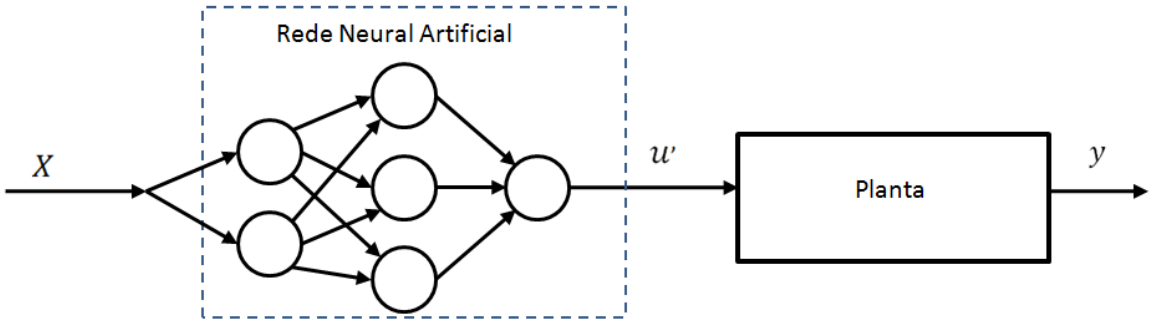
Figura 43 – Identificação do modelo através da utilização de redes neurais artificiais



Fonte: Elaborada pelo autor

Onde x representa o valor de entrada do sistema, u a saída do controlador, y a resposta da planta e u' a saída do modelo neural. Em seguida o controlador PID é substituído pelo modelo gerado pela rede neural artificial conforme a Figura 44 a seguir:

Figura 44 – Diagrama do sistema controlado pelo modelo neural

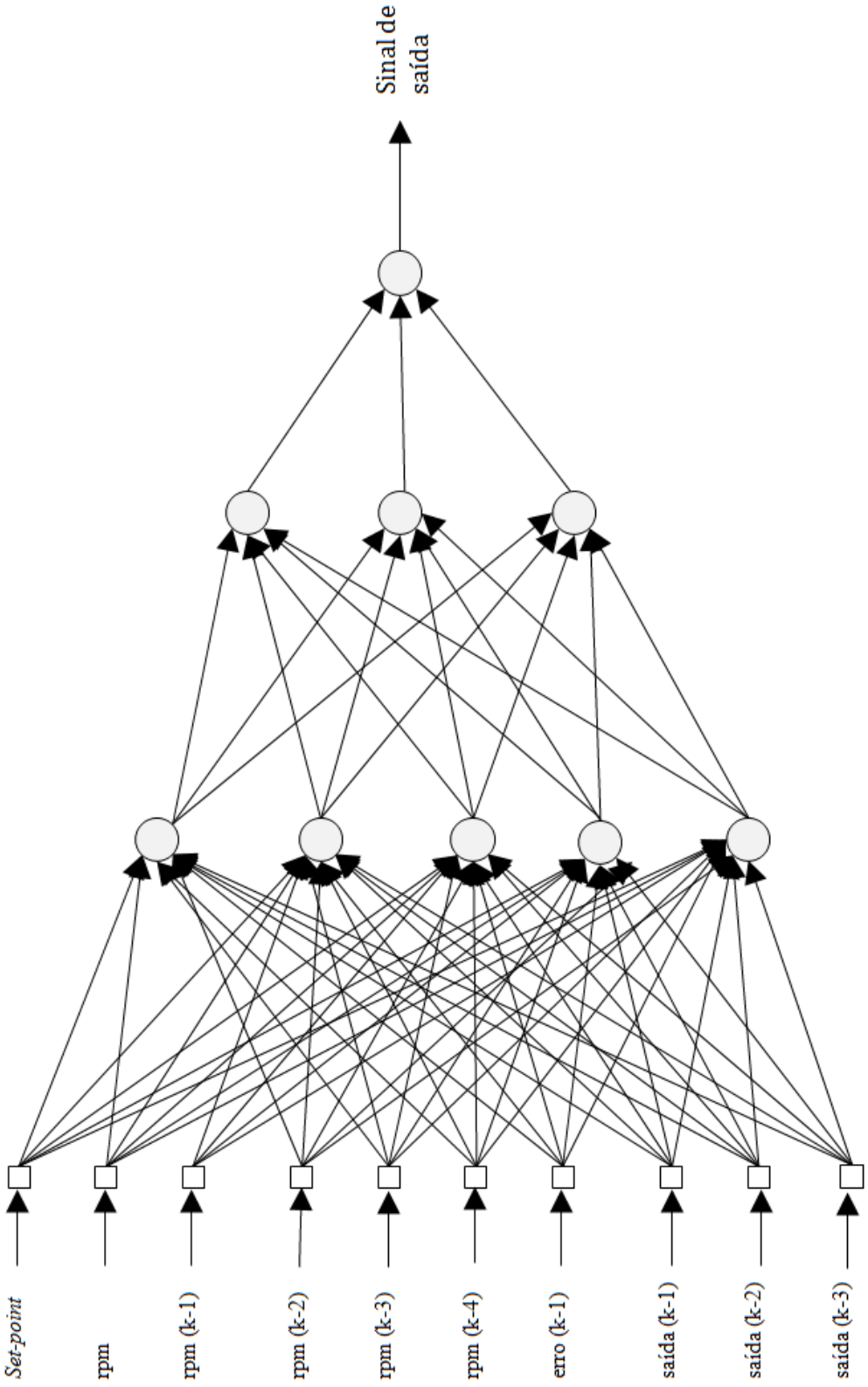


Fonte: elaborado pelo autor

6.3 Parâmetros do treinamento

A rede foi treinada utilizando um modelo com 3(três) camadas possuindo respectivamente 5, 3, 1 neurônios em cada camada. Em todas as camadas foi utilizada a função de ativação do tipo linear. A taxa de aprendizagem e o valor de momento foram fixados em 0,001. O modelo é representado pelo seguinte diagrama da Figura 45 e este tipo de treinamento é adaptado de CUI e SHIN, 1993:

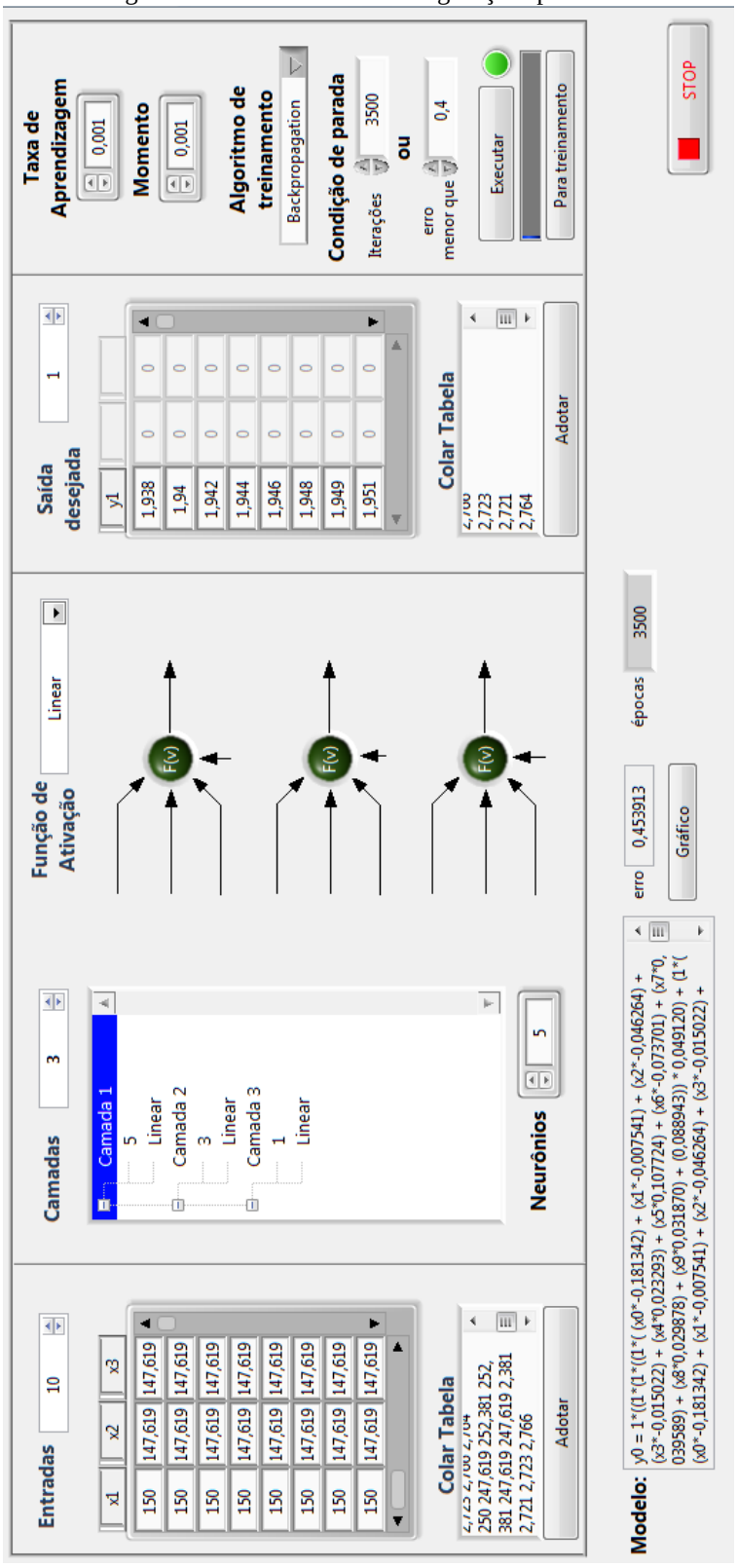
Figura 45 – Diagrama da rede utilizada no treinamento com os dados obtidos no ensaio de controle



Fonte: elaborado pelo autor

A tela do software com as configurações é a apresentada a seguir na Figura 46:

Figura 46 – Imagem do software com as configurações para o treinamento



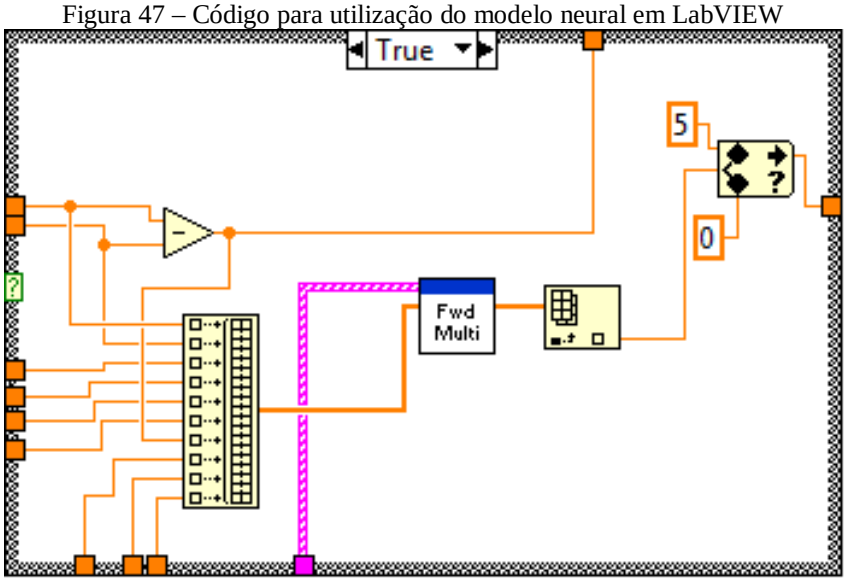
Fonte: elaborado pelo autor

Os dados obtidos após o treinamento são apresentados no apêndice C deste documento.

6.4 Teste do modelo gerado

Após o treinamento utilizou-se o modelo com os pesos e *bias* gerados para substituir o controle existente. O mesmo bloco para gerar a propagação para frente foi utilizado neste sistema, ou seja, foi possível aproveitar os códigos desenvolvidos anteriormente para a interface, desta vez utilizado-o em um sistema a parte. O código em linguagem gráfica LabVIEW processar o modelo obtido é apresentado a seguir na Figura 47:

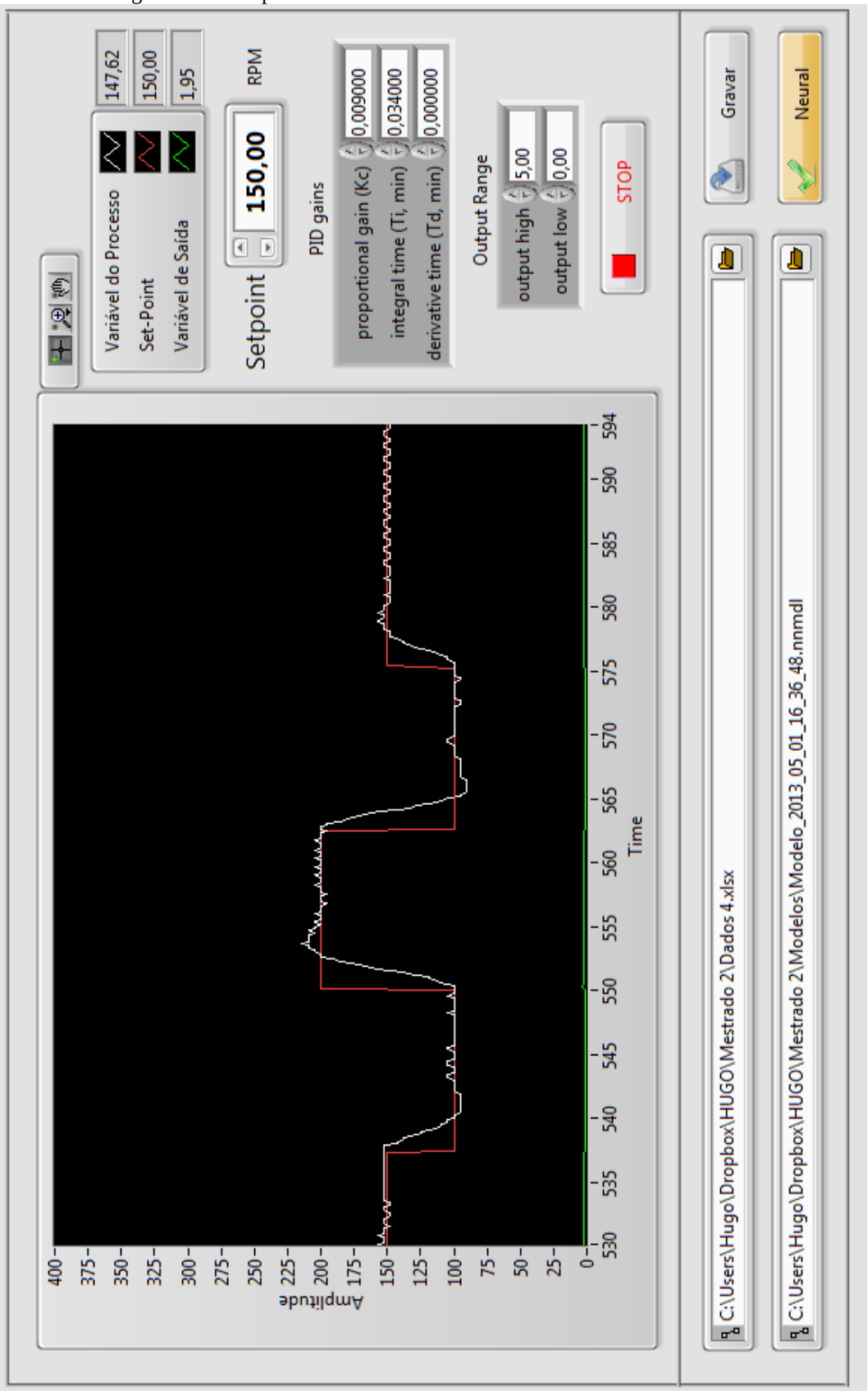
Figura 47 – Código para utilização do modelo neural em LabVIEW



Fonte: elaborado pelo autor

Executando o modelo gerado obteve-se o seguinte comportamento do sistema conforme a Figura 48:

Figura 48 – Comportamento do sistema utilizando o modelo neural

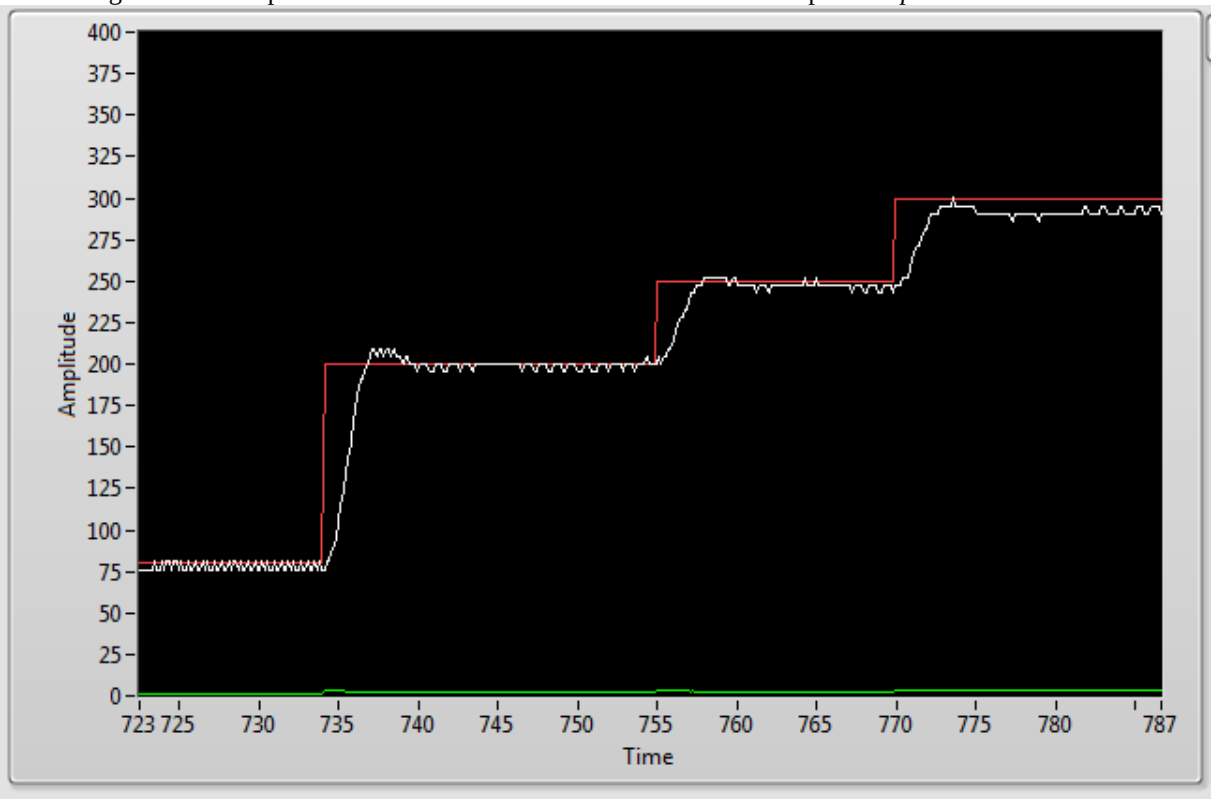


Fonte: elaborado pelo autor

O botão “Neural” em cor amarelada indica que o sistema está sendo executado com o modelo neural escolhido.

Foi realizado também um teste com valores de *set-point* que não haviam sido inseridos no treinamento e obteve-se o seguinte comportamento do sistema conforme a Figura 49:

Figura 49 – Comportamento do sistema utilizando o modelo neural para *set-points* não treinados



Fonte: elaborado pelo autor

Observa-se na Figura 49 que o sistema manteve o comportamento alternando os valores de *set-point* para 80, 200, 250 e 300 rpm. Entretanto é possível observar que para o valor desejado de 300 rpm o sistema possui um erro estacionário. Como esta região de valores não foi inserida na rede neural artificial o sistema não irá necessariamente atingir um resultado esperado.

7 CONSIDERAÇÕES SOBRE O TREINAMENTO DA REDE

Conforme observado, uma das maiores dificuldades em se definir a estrutura de uma Rede Neural Artificial é o dimensionamento de sua topologia. Normalmente, o número de camadas e o número de nós em cada camada é definido em função de uma inspeção prévia nos dados e da complexidade do problema. Uma vez definida a topologia inicial, a estrutura final mais adequada para a modelagem é normalmente obtida através de refinamentos sucessivos, que podem levar a um tempo de dimensionamento alto, já que este tem um grande componente empírico.

O objetivo desta etapa de ajuste é a obtenção de uma topologia de rede que modele com precisão os dados do conjunto de treinamento, mas que também resulte em uma aproximação com boa capacidade de generalização. Como, na maioria dos casos, o conjunto de treinamento de uma Rede Neural Artificial é composto de dados experimentais, estes contêm implicitamente erros inerentes aos processos de amostragem.

Após o término das validações do algoritmo deve ser pensado um modo de auxílio, por parte da ferramenta computacional, quanto a definição da topologia da rede (camadas, neurônios etc.). Os tipos de configuração em uma rede são inúmeros, porém será necessária uma abordagem futura para considerar as melhores topologias para um determinado treinamento.

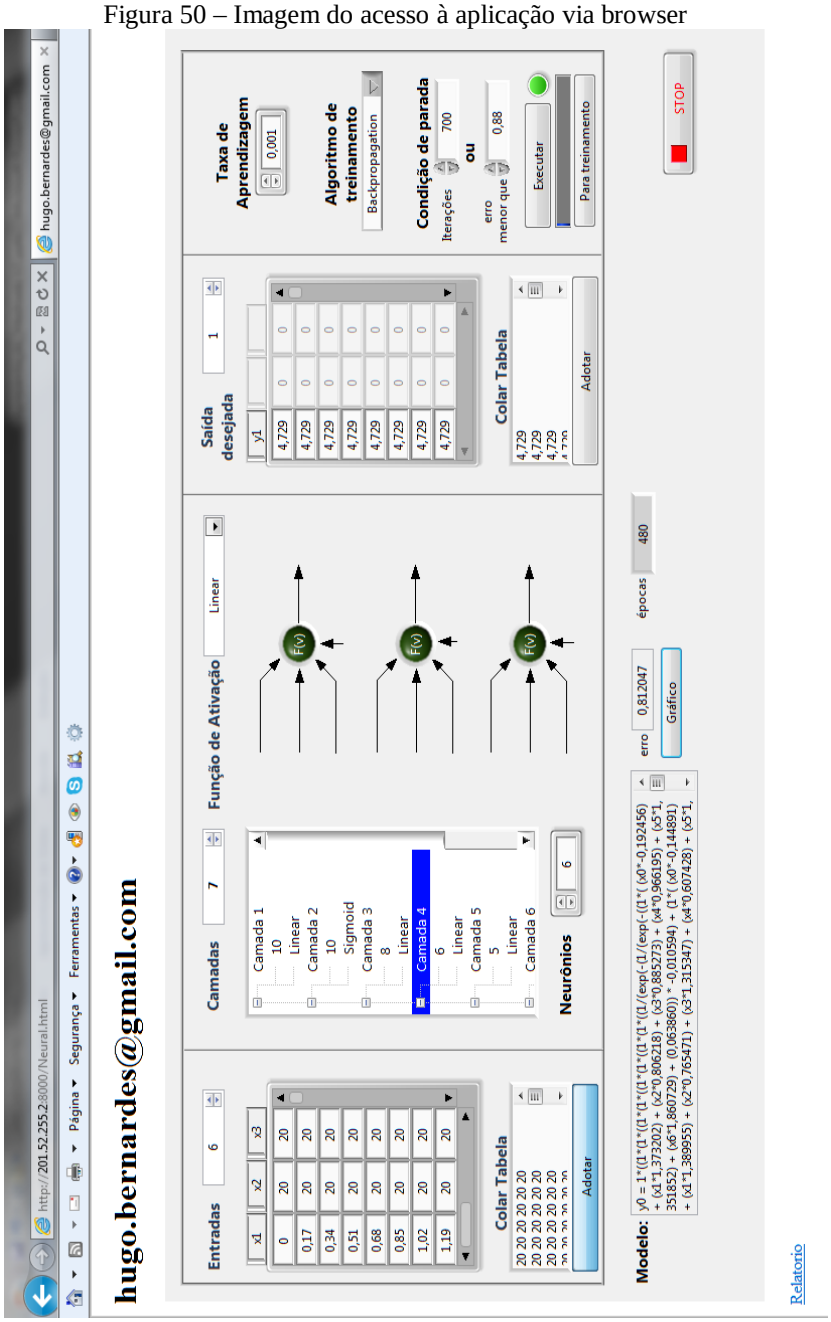
Diversas aplicações são possíveis utilizando os mesmos conceitos abordados neste trabalho. Foi aproveitado no sistema uso da constante de *momento*, que ao ser inserida no algoritmo de ajuste dos pesos, conforme visto no capítulo 2, evitou que o processo de aprendizagem convergisse em um mínimo local da rede (HAYKIN, 2001).

8 PUBLICAÇÃO EM AMBIENTE WEB

Desde o início desta pesquisa considerou-se a utilização da ferramenta em ambiente Web, sendo possível construir uma RNA de qualquer computador conectado à internet e extrair os resultados para qualquer finalidade.

Para colocar esta ideia em prática foi utilizado um recurso oferecido pela mesma plataforma de programação para publicar a página na Web. A seguir, na Figura 50, uma imagem de sua utilização em um navegador padrão:

Figura 50 – Imagem do acesso à aplicação via browser



Fonte: elaborado pelo autor

Como recurso para publicação na internet foi utilizado um serviço gratuito de DNS dinâmico. Este serviço direciona um determinado endereço, pré-definido, para o IP da máquina onde o código está publicado. Entretanto, para que o software esteja sempre disponível na internet é necessário que o computador que execute as ferramentas esteja dentro de um domínio que permita o acesso a esta aplicação de forma irrestrita. Este computador ou servidor deve conter, neste caso, o *Run-time Engine* da National Instrumments instalado. Este programa permite rodar aplicações desenvolvidas em LabVIEW e publicar a interface na Web.

9 CONCLUSÕES

Este trabalho apresentou uma ferramenta para criação de RNA's do tipo *Perceptron* de múltiplas camadas utilizando o algoritmo de *retropropagação*. A ferramenta foi desenvolvida com o ambiente de programação LabVIEW e foi criada uma interface intuitiva para o usuário.

O algoritmo de *retropropagação* para o treinamento da rede foi escolhido por ser amplamente utilizado em identificação de modelos. Para isso construiu-se um código dividido em três etapas: propagação para frente, propagação para trás e ajuste dos pesos. Foram implementadas as funções de ativação do tipo *linear*, *sigmoidal* e *tangente hiperbólica*.

A utilização do ambiente de programação gráfica LabVIEW permitiu a melhor visualização dos algoritmos implementados, possibilitando a rápida verificação dos resultados das cálculos intermediários e a rápida criação de sub-códigos (*subvi's*).

Com o ambiente de desenvolvimento LabVIEW foi possível criar também a interface para a inserção dos dados a serem treinados. Para isso foram identificados os principais parâmetros de entrada para uma rede com algoritmo de *retropropagação*. A interface desenvolvida permite selecionar os parâmetros:

a) Número de camadas;

b) Número de neurônios em cada camada;

c) Função de ativação para cada camada;

d) Taxa de aprendizagem;

e) Número máximo de épocas (iterações);

f) Erro máximo como condição de parada.

Além disso, a ferramenta desenvolvida permite que o usuário insira diretamente as tabelas com os valores de entrada e saída desejadas copiando-as de outra fonte de dados como o Excel, por exemplo. Para a inserção de tabelas com um número maior de valores foi criada a possibilidade de "copiar" e "colar" as tabelas no software. Desta forma não há a necessidade de enviar ou importar um arquivo ao sistema.

Após o treinamento é gerado um relatório com as informações inseridas pelo usuário, o gráfico de erro ao longo do treinamento, a tabela com o teste do modelo, os valores dos pesos e *bias* gerados após o treinamento, conforme apresentados no apêndice C deste documento.

O ambiente de programação em linguagem gráfica permitiu que a interface fosse diretamente publicada em ambiente Web. Assim, permite que o usuário construa modelos de redes com *retropropagação* através da internet.

Foi realizado o teste para verificar o desempenho da ferramenta em identificação de modelos. Chegou-se ao erro desejado após o ajuste empírico dos parâmetros de treinamento. Verificou-se nos testes a capacidade de extrair o modelo após o treinamento e testá-lo em outro ambiente de programação. Foi apresentada a tabela com valores de saída do modelo após o treinamento.

Foi possível também, utilizar as funções desenvolvidas para um teste em que se substituiu um algoritmo de um sistema de malha fechada, demonstrando assim a capacidade de, além de utilizar a interface para o treinamento, aproveitar os sub-códigos criados para desenvolvimento de projetos em LabVIEW. O modelo neural obtido no sistema manteve as características inicias demonstrando a capacidade das ferramentas desenvolvidas de identificar o sistema através dos dados de ensaio.

Verificou-se que, no treinamento de redes mais complexas, ou seja, com grande número de neurônios e camadas o processamento é demorado. Por isso é importante a *otimização* do processamento como, por exemplo, a utilização de GPU's ou de programação distribuída aproveitando os recursos existentes de computação em *nuvem*.

De forma geral pôde-se ao longo do trabalho compreender a evolução dos cálculos baseados em redes neurais artificiais e com isso propor uma ferramenta que difere nas demais em relação à facilidade de uso e capacidade para rápida implementação de modelos com *retropropagação*. As ferramentas já existentes e pesquisadas neste trabalho não apresentaram uma interface padrão para configuração de redes com *retropropagação*, o que torna este trabalho um diferencial, assim como o estudo para criação da interface.

Outros algoritmos de treinamento ainda podem ser incorporados à ferramenta para complementar os tipos de redes como, por exemplo, as redes com função de base radial, e as redes de Kohonen.

Como a plataforma de desenvolvimento em linguagem gráfica possibilita fácil interação com plataformas de hardware embarcado, as ferramentas poderão ainda facilitar os testes de aplicações de RNA's em chips como o FPGA, e a direta integração da linguagem com este tipo de hardware.

REFERÊNCIAS

BERNACKI, M.; WŁODARCZYK, P. Backpropagation. **Site da Universidade de Ciência e Tecnologia da Polônia**, 2005. Disponível em: <http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html>. Acesso em: 20 Julho 2012.

CAMPOS, J. R. et al. **Implementação de Redes Nurais Artificiais Utilizando a linguagem de Programação JAVA**. Serra Negra: 9th Brazilian Conference on Dynamics, Control and their Applications, 2010.

CANETE, J. F.; PEREZ, S. G.; SAZ, O. P. **Software Tools for System Identification and Control using Neural Networks in Process Engineering**. [S.l.]: World Academy of Science, Engineering and Technology, v. 47, 2008.

CHANGEUX, J. P.; DANCHIN, A. **Selective Stabilization of Developing Synapsis as a Mechanism for the Specification of Neural Networks**. [S.l.]: Nature, v. 264, 1976.

COOPER, A.; REINMAN, R.; CRONIN, D. **About Face 3:** The Essential of Interaction Design. Indianapolis: Wiley Publishing, 2007.

CUI, X.; SHIN, K. G. **Direct Control and Coordination using Neural Networks**. 3. ed. [S.l.]: IEEE Trans. Syst., Man, and Cybern., v. 23, 1993.

DIKAIAKOS, M. D. et al. Cloud Computing - Distributed Internet Computing for IT and Scientific Research. **IEEE Internet Computing**, 2009.

HAYKIN, S. **Redes Neurais:** princípios e práticas. Tradução de Paulo Martins Engel. 2ª Edição. ed. Porto Alegre: Bookman, 2001.

HEBB, D. O. **The Organization of Behavior**. [S.l.]: John Wiley & Sons, 1949.

HETCH-NIELSON, R. **Neurocomputing**. [S.l.]: Addison-Wesley, 1990.

HOPFIELD, J. J. **Neural Networks and Physical Systems with Emergent Collective Computational Anilities**. [S.l.]: Proc. Natl. Acad. Sci. USA, v. 79, 1982.

HUANG, Y. Advances in Artificial Neural Networks - Methodoloical Development and Application. **Open Access - Algorithms**, 2009. Disponível em: <www.mdpi.com/journal/algorithms>. Acesso em: Julho 2012.

HUSH, D. R.; HORME, B. G. **Progress in supervised neural networks:** What's new since Lippman? [S.l.]: IEEE Signal Processing Magazine, v. 10, 1993.

KOHONEN, T. **Self-Organization and Associative Memory**. [S.l.]: Springler-Verlag, 1984.

KOWALSKA, T. O.; KAMINSKI, M. FPGA Implementation of the Multilayer Neural Network for the Speed Estimation of the Two-Mass Drive System. **IEEE Transaction on Industrial Informatics**, 2011.

LIPPMAN, R. P. **An Introduction to Computing with Neural Networks**. [S.l.]: IEEE Acoustics, Speech and Signal Processing, v. 4, 1987.

LOPES, N.; RIBEIRO, B. **Hybrid learning multi neural architecture**. [S.l.]: INNS-IEEE International Joint Conference on Neural Networks (IJCNN 2001), v. 4, 2001.

LOPES, N.; RIBEIRO, B. **Stochastic GPU-based multithread implementation of multiple back-propagation**. [S.l.]: Second International Conference on Agents and Artificial Intelligence (ICAART 2010), 2010.

LOPES, N.; RIBEIRO, B. **An evaluation of multiple feed-forward networks on GPUs**. [S.l.]: International Journal of Neural Systems (IJNS), 2011.

MENDEL, M. J.; MCLAREN, W. R. **Reinforcement-Learning Control and Pattern Recognition Systems**. New York: Academic Press, v. 66, 1970.

MINSKY, M. L.; PAPERT, S. A. **Perceptrons**. Cambridge: MIT Press, 1969.

NASCIMENTO, J. C. L.; YONEYAMA, T. **Inteligência Artificial em Controle e Automação**. São Paulo: Blucher, 2008.

PARKER, D. B. **Optimal Algorithms for Adaptative Networks:** Second Order Backpropagation, Second Order Direct Propagation, and Second Order Hebbian Learning. San Diego, CA: IEEE 1st International Conference on Neural Networks, v. 2, 1987.

PERETTO, P. **An Introduction to the Modelling of Neural Networks**. [S.l.]: Cambridge University Press, 1992.

PLAGIANAKOS, V. P.; D, M. G.; VRAHATIS, M. N. Distributed computing methodology for training neural networks in an image-guided diagnostic application. **Birbeck ePrints**, London, 2006.

PONCE, P.; GUTIERREZ, A. M. LabVIEW for Intelligent Control Research and Education. **IEEE**, 2010.

ROSEMBLAT, F. **Principle of Neurodynamics:** Perceptron and Theory of Brain Mechanisms. [S.l.]: Spartan Books, 1962.

ROSEMBLATT, F. **The Perceptron:** A Probabilistic Model for Information Storage and Organization in the Brain. [S.l.]: Psychological Review, v. 65, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning Representations of Back-propgation Errors**. Londres: Nature, v. 323, 1986.

RUMELHART, D. E.; ZISPER, D. **Feature Discovery by Competitive Learning**. [S.l.]: Cognitive Science, v. 9, 1985.

SEJNOWSKI, T. J. **Strong Covariancewith Non-linearity Interacting Neurons**. [S.l.]: Journal of Mathematical Biology, v. 4, 1977.

TISAN, A. et al. **Artificial olfaction system with hardware on-chip learning neural networks**. [S.l.]: 12th International Conference on Optimization os Eletrical and Eletronic Equipment, OPTIM. 2010.

WERBOS, P. J. **The Roots of Back Propagation:** From Ordered Dferivatives to Neural Networks and Political Forecasting (Adaptative and Learning Systems for Signal Processing). [S.l.]: John Wiley & Sons, 1994.

WIDROW, B.; HOFF, J. M. E. **Adaptative Switching Circuits**. [S.l.]: IRE WESCON Convention Record, 1960.

WIDROW, S.; STEAMS, S. D. **Adaptative Signal Processing**. Eglewood Cliffs, NJ: Prentice-Hall, 1985.

APÊNDICE A: LINGUAGEM GRÁFICA LABVIEW

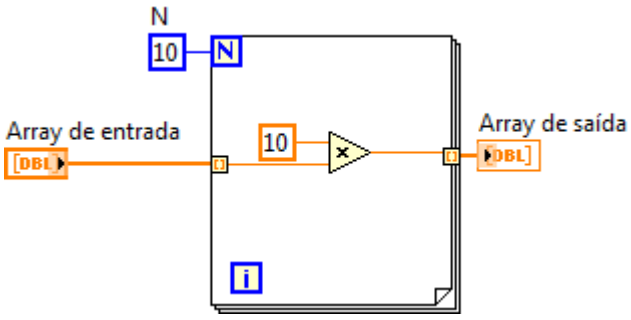
1 LINGUAGEM GRÁFICA PARA CRIAÇÃO DA FERRAMENTA

A linguagem gráfica utilizada possui diversos recursos comuns em outras linguagens como laços de repetição, comandos para tomada de decisão e valores armazenados em memória. Para os já habituados com a linguagem, este capítulo traz apenas uma elucidação básica das estruturas de programação. Os principais recursos serão brevemente apresentados a seguir.

1.1 Laços de repetição em linguagem gráfica

Na linguagem utilizada podemos construir laços do tipo *for* para repetição de operações em um número determinado de vezes. O seu diagrama é apresentado como a seguir na Figura 51:

Figura 51 – laço de repetição *for*

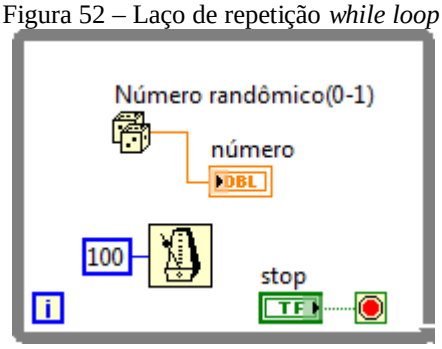


Fonte: elaborado pelo autor

onde N é o número de repetições que o laço irá executar tudo que estiver dentro da região retangular. Neste caso o código irá multiplicar os 10 elementos do **Array de entrada** por 10 e irá gerar um novo **Array de saída** com 10 elementos.

Para laços onde espera-se que uma condição seja atendida para que os comandos parem de se repetir (chamados de *while loops*), tem-se o seguinte diagrama da Figura 52, onde é gerado um número randômico a cada execução do laço até que o controle **stop** mude seu estado para **verdadeiro** e o laço seja interrompido. Também é possível determinar o tempo para cada execução do laço, que neste caso está fixado em 100 milissegundos.

Figura 52 – Laço de repetição *while loop*

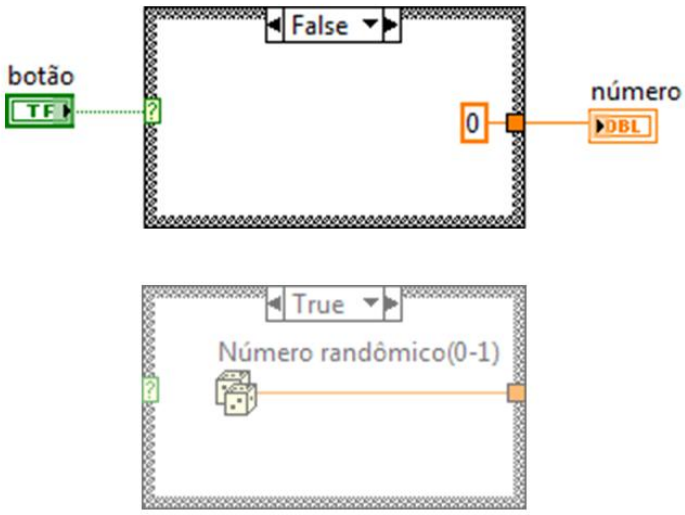


Fonte: elaborado pelo autor

1.2 Tomadas de decisão em linguagem gráfica

Recurso mais comum para as tomadas de decisão na linguagem gráfica LabVIEW é a chamada estrutura *case* que é apresentada na Figura 53:

Figura 53 – Estrutura para tomadas de decisões (*case*)



Fonte: elaborado pelo autor

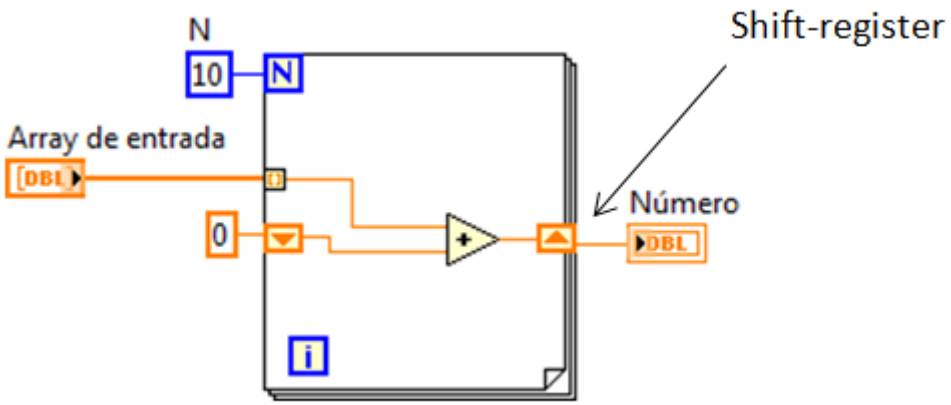
Onde a valor de saída **número** será igual a 0 e se o valor da entrada **botão** for verdadeiro o valor de saída passa e ser um número randômico.

1.3 Valores armazenados em memória

A forma mais rápida de armazenar valores em memória ou utilizar valores de iterações anteriores dentro dos laços de repetição é o chamado *shift-register* na linguagem LabVIEW.

Pode ser utilizado tanto no laço *for* quanto no laço *while*, conforme demonstrado a seguir na Figura 54.

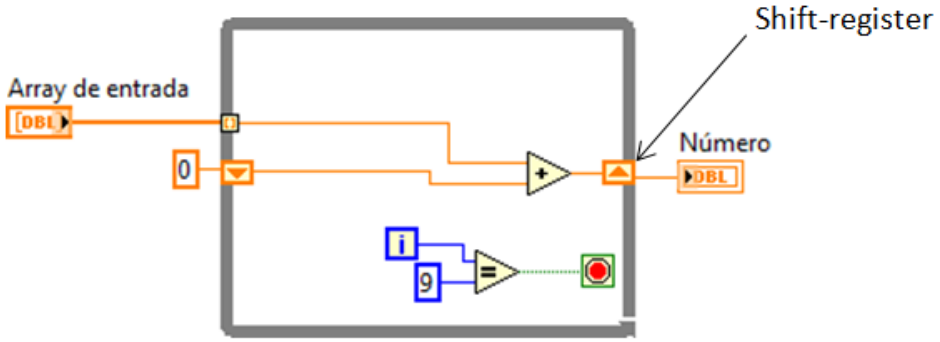
Figura 54 – *Shift-register* para armazenamento de valores em memória



Fonte: elaborado pelo autor

Onde o resultado apresentado em **número** após a execução será igual ao somatório dos valores do **Array de entrada.** O mesmo código pode ser representado utilizando o *while loop* conforme a Figura 55 a seguir:

Figura 55 - w*hile loop* com *shift-register*

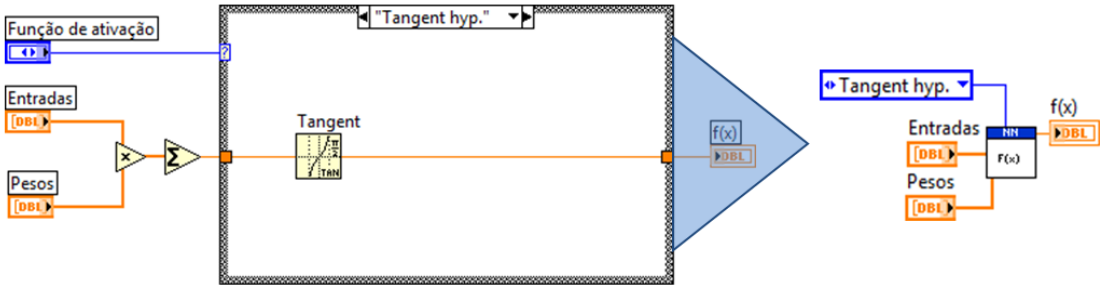


Fonte: elaborado pelo autor

1.4 Sub-códigos (*subvi's*)

Os códigos podem ser "encapsulados" e utilizados dentro de outros códigos permitindo que o código seja dividido em níveis facilitando a sua construção e manutenção. A seguir um exemplo de código transformado em um sub-código para ser inserido posteriormente em um código maior. Este sub-código é chamado de **subvi,** já que a extensão do arquivo salvo em LabVIEW é ".vi". Figura 56 tem-se um a conversão de um código em um *subvi*.

Figura 56 - Código transformado em um *subvi*



Fonte: elaborado pelo autor

APÊNDICE B: TESTES COMPARATIVOS E VALIDAÇÃO DAS FERRAMENTAS

Foram realizados testes com intuito de comparar os resultados gerados pela ferramenta computacional desenvolvida e os resultados de outra ferramenta existente que propõe fornecer ferramentas para configuração e treinamento de redes neurais artificiais. Para os pesos e *bias* encontrados foi mantido um número grande de algarismos decimais (15 algarismos) para uma melhor avaliação dos resultados.

1 TESTE 1

Foi utilizado o *toolbox* do MATLAB e realizou-se o treinamento com as seguintes características:

a) Conjunto de entradas e saídas:

Tabela 2 – Valores de entradas e saídas desejadas utilizados no teste de comparação

Entradas		Saídas
0,6	0,5	0,4
0,7	0,6	0,5
0,8	0,7	0,6

Fonte: elaborado pelo autor

b)Número de camadas intermediárias: 2.

c)Número de neurônios em cada camada:
3 neurônios na camada de entrada e 1 neurônio na camada de saída

d) Função de ativação:
Linear para todos os neurônios

e) Taxa de aprendizagem: 0,01

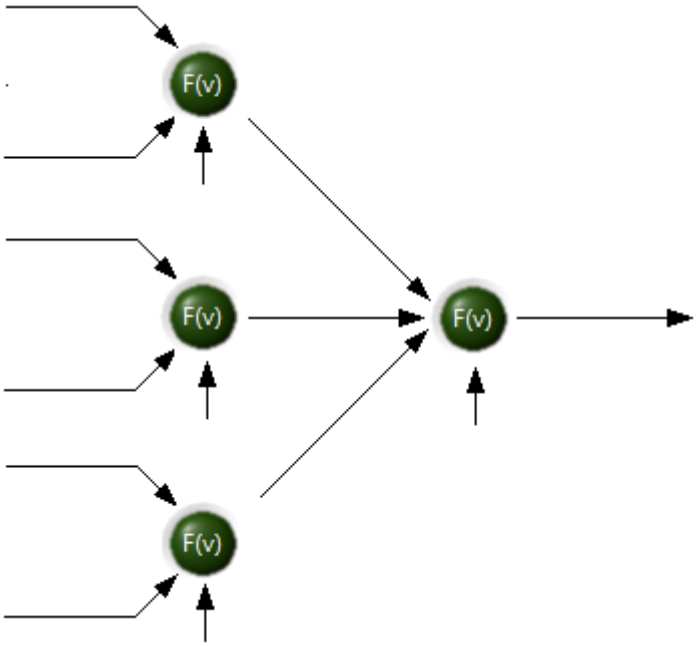
f) Algoritmo de aprendizagem: Retropropagação (*back-propagation*)

g) Critério de parada: 50000 iterações ou erro médio quadrado menor que 0,0001

h) Todos os pesos e *bias* inicializados em 0,1

Diagrama de representação da rede é apresentado na Figura 57:

Figura 57 – Diagrama da rede neural artificial usada para o treinamento



Fonte: elaborado pelo autor

1.1 Comandos em MATLAB para o treinamento proposto

Foi inserido o seguinte código na janela de comando do MATLAB:

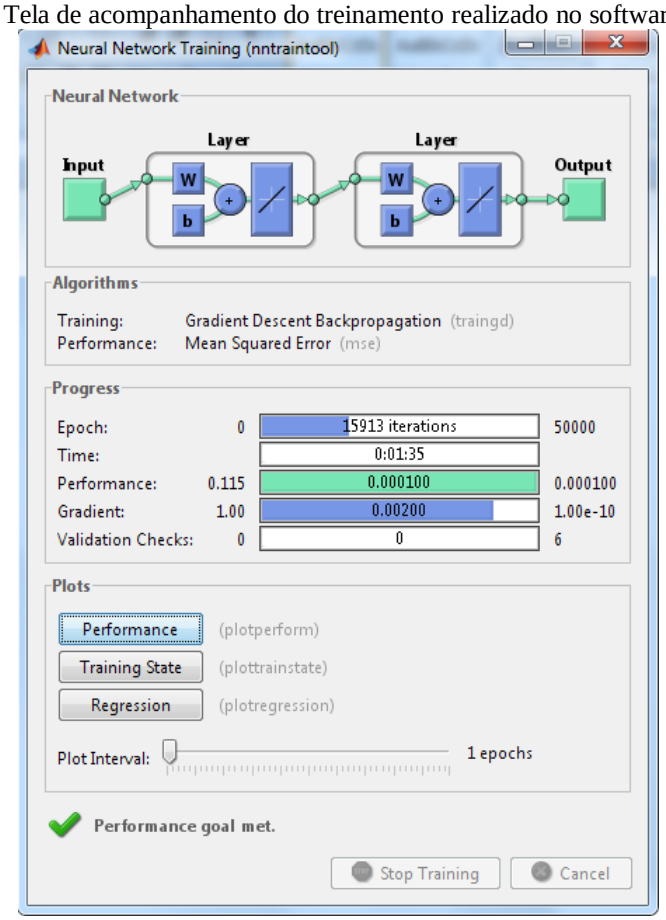
```
clear all
clc
%Definição de entradas e saídas
inputs=[0.6 0.7 0.8;0.5 0.6 0.7];
targets=[0.4 0.5 0.6];
%Criando a rede
net=newff(minmax(inputs),[3 1], {'purelin','purelin'},'traingd');
%Definindo parâmetros
net.trainParam.epochs = 50000;
net.trainParam.goal = 1e-4;
%Inicializando os pesos e bias
net.LW{2,1}=[0.1,0.1,0.1];  %dim 1x3
```

```
net.IW{1,1}=[0.1,0.1;0.1,0.1;0.1,0.1];
net.b{1,1}=[0.1;0.1;0.1];
net.b{2,1}=[0.1];
%Treinando a rede
[net,tr]=train(net,inputs,targets)
```

1.1.a Resultados

O *toolkit* mostrou a seguinte tela como resultado do treinamento conforme a Figura 58:

Figura 58 – Tela de acompanhamento do treinamento realizado no software MATLAB



Fonte: elaborado pelo autor

O treinamento atingiu um erro médio quadrado menor que 0,0001 em 15913 iterações. Para obter os novos pesos ajustados foram inseridos na janela de comando os seguintes comandos:

```
net.LW{2,1}
net.IW{1,1}
```

net.IW{2,1}

net.b{1,1}

net.b{2,1}

Utilizou-se também o comando "format long" para que o MATLAB mostrasse um maior número de casas decimais.

Obtiveram-se assim os novos pesos e *bias*:

a) Pesos na camada 1

Tabela 3 – Pesos da camada 1 (teste 1.1)

Neurônio 1	Neurônio 2	Neurônio 3
0,324537969712158	0,324537969712158	0,324537969712158
0,334356477031032	0,334356477031032	0,334356477031032

b) *Bias* na camada 1

Tabela 4 – *Bias* da camada 1 (teste 1.1)

Neurônio 1	Neurônio 2	Neurônio 3
0,444018562636704	0,444018562636704	0,444018562636704

c) Pesos na camada de saída

Tabela 5 – Pesos da camada de saída (teste 1.1)

0,001814926811246	0,001814926811246	0,001814926811246

d) *Bias* na camada de saída

-0,071758205832244

1.2 Testes com a ferramenta desenvolvida em ambiente LabVIEW

A tela ficou com os seguintes parâmetros conforme a Figura 59:

Figura 59 – Tela com a configuração realizada no programa desenvolvido em LabVIEW

Fonte: elaborado pelo autor

O treinamento atingiu um erro médio quadrado menor que 0,0001 em 15921 iterações.

Os novos pesos obtidos foram:

1.2.a Resultados

a) Pesos na camada 1

Tabela 6 – Pesos da camada 1 (teste 1.2)

Neurônio 1	Neurônio 2	Neurônio 3
0,333310327706074	0,333310327706074	0,333310327706074
0,344161677753542	0,344161677753542	0,344161677753542

b) *Bias* na camada 1

Tabela 7 – *Bias* da camada 1 (teste 1.2)

Neurônio 1	Neurônio 2	Neurônio 3
-0,0085135004746664	-0,0085135004746664	-0,0085135004746664

c) Pesos na camada de saída

Tabela 8 – Pesos da camada de saída (teste 1.2)

0,457799973450425	0,457799973450425	0,457799973450425

d) *Bias* na camada de saída

-0,0919225777815564

2 TESTE 2

Com o intuito de verificar as diferenças entre as duas ferramentas efetuou-se um treinamento simples com apenas 1 neurônio e 1 camada.

Este treinamento possui as seguintes características:

a) Conjunto de entradas e saídas:

Tabela 9 – conjunto de entradas e saídas para o teste com um neurônio

Entradas		Saídas
0,6	0,5	0,4

b) Número de camadas intermediárias: 0.

c) Número de neurônios em cada camada:

1 neurônios na camada de entrada

d) Função de ativação:Linear

e) Taxa de aprendizagem: 0,01

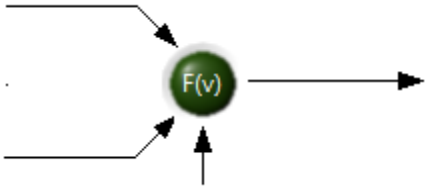
f) Algoritmo de aprendizagem: *Retropropagação* (*back-propagation*)

g) Critério de parada: 1 iteração

h) Todos os pesos e *bias* inicializados em 0,1

Diagrama de representação da rede conforme a Figura 60:

Figura 60 – Diagrama com apenas um neurônio



Fonte: elaborado pelo autor

2.1 Comandos no MATLAB

Foram inseridos os seguintes comandos:

```
clear all
clc
%Definição de entradas e saídas
inputs=[0.6;0.5];
targets=[0.4];
%Criando a rede
net=newff(minmax(inputs),[1], {'purelin'},'traingd');
%Definindo parâmetros
net.trainParam.epochs = 1;
net.trainParam.goal = 1e-4;
net.trainParam.lr=0.01;
net.IW{1,1}=[0.1,0.1];
net.b{1,1}=[0.1];
[net,tr]=train(net,inputs,targets)
```

2.1.1 Resultados no MATLAB

Obtiveram-se os pesos e *bias*:

Tabela 10 – Pesos obtidos no teste 2.1

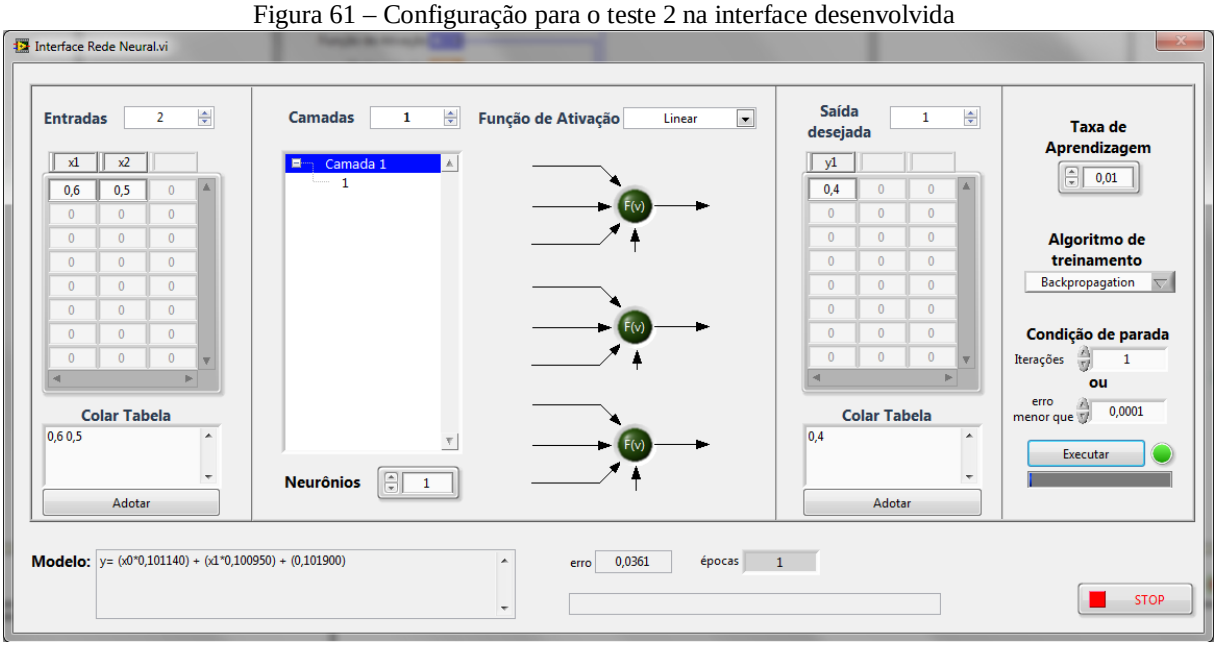
0,102280000000000	0,101900000000000

Bias camada de saída:

0,103800000000000

2.2 Configurações no ambiente desenvolvido em linguagem gráfica

Figura 61 – Configuração para o teste 2 na interface desenvolvida



Fonte: elaborado pelo autor

2.2.1 Resultados

Obtiveram-se os pesos e *bias*:

Tabela 11 – Pesos obtidos no teste 2.2

0,101140000000000	0,100950000000000

Bias camada de saída:

0,101900000000000

2.2.3 Resultado com 2 iterações

Executando o programa com 2 iterações obteve-se o resultado mais próximo do item 2.1.1 conforme a seguir:

Tabela 12 – Pesos obtidos no teste 2.2 com duas iterações

0,102261646000000	0,101884705000000

Bias camada de saída:

0,103769410000000

Não é objetivo desta comparação avaliar a performance de cada ferramenta mas apenas avaliar os resultados. Ambas apresentaram resultados semelhantes no Teste 1, tanto para o número de épocas quanto para os valores de pesos e de *bias.*

No Teste 2, verificou-se que os resultados foram diferentes para um algoritmo simples com apenas 1 neurônio. A ferramenta desenvolvida em LabVIEW apresentou os valores de peso e *bias,* conforme os cálculos apresentadas no trabalho. Os comandos inseridos no MATLAB apresentaram resultados similares.

Para o teste com 2 iterações (épocas) na ferramenta desenvolvida obteve-se um resultado mais próximo do obtido no MALAB, entretanto observa-se que pode haver também uma diferença em relação ao arredondamentos utilizados que podem estar relacionados às configurações numéricas do ambiente MATLAB. No caso da ferramenta desenvolvida em LabVIEW utilizou-se, em todo o código, de variáveis do tipo *double* de 64 bits com aproximadamente 15 dígitos decimais.[3]

Assim, a ferramenta proposta apresentou os resultados esperados e foi capaz de realizar os treinamentos sugeridos gerando os pesos e *bias* satisfatórios, ou seja, que atendam os critérios de erro mínimo quadrado ou de iterações (épocas) estabelecidos.

[3] Conforme visto no link http://zone.ni.com/reference/en-XX/help/371361G-01/lvhowto/numeric_data_types_table/, Acessado em julho de 2012

APÊNDICE C: RELATÓRIO GERADO APÓS O TREINAMENTO

É apresentado a seguir o relatório do ensaio gerado automaticamente pela ferramenta desenvolvida. Este relatório está no formato *html* e pode ser acessado através do link "Relatório" encontrado na parte inferior da interface acessada através da internet.

Camada 1

	pesos	pesos	pesos	pesos	pesos	pesos	pesos	pesos	pesos	pesos	bias
Neurônio 1	-0,181342	-0,007541	-0,046264	-0,015022	0,023293	0,107724	-0,073701	0,039589	0,029878	0,031870	0,088943
Neurônio 2	-0,181342	-0,007541	-0,046264	-0,015022	0,023293	0,107724	-0,073701	0,039589	0,029878	0,031870	0,088943
Neurônio 3	-0,181342	-0,007541	-0,046264	-0,015022	0,023293	0,107724	-0,073701	0,039589	0,029878	0,031870	0,088943
Neurônio 4	-0,181342	-0,007541	-0,046264	-0,015022	0,023293	0,107724	-0,073701	0,039589	0,029878	0,031870	0,088943
Neurônio 5	0,100100	0,100100	0,100100	0,100100	0,100100	0,100100	0,100100	0,100100	0,100100	0,100100	0,100100

Camada 2

	pesos	pesos	pesos	bias
Neurônio 1	0,049120	0,049120	0,049120	0,049120
Neurônio 2	0,058428	0,058428	0,058428	0,058428
Neurônio 3	0,058428	0,058428	0,058428	0,058428

Camada 3

	pesos	pesos	pesos	bias
Neurônio 1	0,044757	-0,117342	-0,117342	0,813377

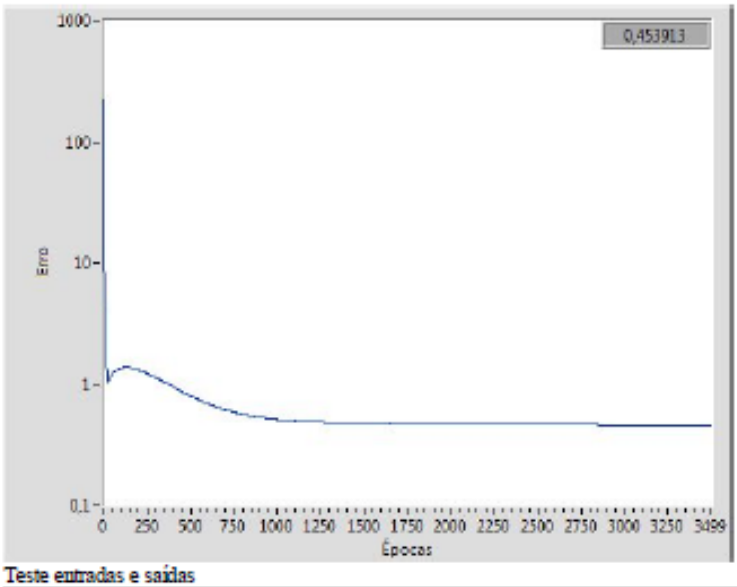
	Camada 1	Camada 2	Camada 3
Função de ativação	Linear	Linear	Linear

Modelo:
y0 = 1*((1*(1*((1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,049120) + (1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,049120) + (1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,049120) + (1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,049120) + (1*((x0*0,100100) + (x1*0,100100) + (x2*0,100100) + (x3*0,100100) + (x4*0,100100) + (x5*0,100100) + (x6*0,100100) + (x7*0,100100) + (x8*0,100100) + (x9*0,100100) + (0,100100)) * 0,083476)) + (0,127701)) * 0,044757) + (1*(1*((1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,058428) + (1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,058428) + (1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,058428) + (1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,058428) + (1*((x0*0,100100) + (x1*0,100100) + (x2*0,100100) + (x3*0,100100) + (x4*0,100100) + (x5*0,100100) + (x6*0,100100) + (x7*0,100100) + (x8*0,100100) + (x9*0,100100) + (0,100100)) * 0,000000)) + (0,058428)) * -0,117342) + (1*(1*((1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,058428) + (1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,058428) + (1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,058428) + (1*((x0*-0,181342) + (x1*-0,007541) + (x2*-0,046264) + (x3*-0,015022) + (x4*0,023293) + (x5*0,107724) + (x6*-0,073701) + (x7*0,039589) + (x8*0,029878) + (x9*0,031870) + (0,088943)) * 0,058428) + (1*((x0*0,100100) + (x1*0,100100) + (x2*0,100100) + (x3*0,100100) + (x4*0,100100) + (x5*0,100100) + (x6*0,100100) + (x7*0,100100) + (x8*0,100100) + (x9*0,100100) + (0,100100)) * 0,000000)) + (0,058428)) * -0,117342)) + (0,813377)

Taxa de Aprendizagem: 0,001000

Momento: 0,001000

Gráfico do erro quadrado médio



Teste entradas e saídas

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	saida desejada 1	y1
150,000	147,619	147,619	147,619	147,619	147,619	2,381	1,938	1,938	1,938	1,938	1,980
150,000	147,619	147,619	147,619	147,619	147,619	2,381	1,938	1,938	1,938	1,940	1,980
150,000	147,619	147,619	147,619	147,619	147,619	2,381	1,940	1,938	1,938	1,942	1,980
150,000	147,619	147,619	147,619	147,619	147,619	2,381	1,942	1,940	1,938	1,944	1,980
150,000	147,619	147,619	147,619	147,619	147,619	2,381	1,944	1,942	1,940	1,946	1,980
150,000	147,619	147,619	147,619	147,619	147,619	2,381	1,946	1,944	1,942	1,948	1,980
150,000	147,619	147,619	147,619	147,619	147,619	2,381	1,948	1,946	1,944	1,949	1,980
150,000	147,619	147,619	147,619	147,619	147,619	2,381	1,949	1,948	1,946	1,951	1,980
150,000	147,619	147,619	147,619	147,619	147,619	2,381	1,951	1,949	1,948	1,953	1,980
150,000	147,619	147,619	147,619	147,619	147,619	2,381	1,953	1,951	1,949	1,955	1,980
150,000	152,381	147,619	147,619	147,619	147,619	-2,381	1,955	1,953	1,951	1,912	1,965
150,000	152,381	152,381	147,619	147,619	147,619	-2,381	1,912	1,955	1,953	1,910	1,977
150,000	152,381	152,381	152,381	147,619	147,619	-2,381	1,910	1,912	1,955	1,908	1,982
150,000	147,619	152,381	152,381	152,381	147,619	2,381	1,908	1,910	1,912	1,951	1,993
150,000	147,619	147,619	152,381	152,381	152,381	2,381	1,951	1,908	1,910	1,953	1,960
150,000	147,619	147,619	147,619	152,381	152,381	2,381	1,953	1,951	1,908	1,955	1,954
150,000	152,381	147,619	147,619	147,619	152,381	-2,381	1,955	1,953	1,951	1,912	1,943
150,000	152,381	152,381	147,619	147,619	147,619	-2,381	1,912	1,955	1,953	1,910	1,977
150,000	152,381	152,381	152,381	147,619	147,619	-2,381	1,910	1,912	1,955	1,908	1,982
150,000	152,381	152,381	152,381	152,381	147,619	-2,381	1,908	1,910	1,912	1,907	1,979
150,000	147,619	152,381	152,381	152,381	152,381	2,381	1,907	1,908	1,910	1,949	1,971
150,000	147,619	147,619	152,381	152,381	152,381	2,381	1,949	1,907	1,908	1,951	1,960
150,000	147,619	147,619	147,619	152,381	152,381	2,381	1,951	1,949	1,907	1,953	1,954
150,000	152,381	147,619	147,619	147,619	152,381	-2,381	1,953	1,951	1,949	1,910	1,943
150,000	152,381	152,381	147,619	147,619	147,619	-2,381	1,910	1,953	1,951	1,908	1,977
150,000	152,381	152,381	152,381	147,619	147,619	-2,381	1,908	1,910	1,953	1,907	1,982
150,000	147,619	152,381	152,381	152,381	147,619	2,381	1,907	1,908	1,910	1,949	1,993
150,000	147,619	147,619	152,381	152,381	152,381	2,381	1,949	1,907	1,908	1,951	1,960
150,000	147,619	147,619	147,619	152,381	152,381	2,381	1,951	1,949	1,907	1,953	1,954
200,000	147,619	147,619	147,619	147,619	152,381	52,381	1,953	1,951	1,949	2,425	2,582
200,000	152,381	147,619	147,619	147,619	147,619	47,619	2,425	1,953	1,951	2,422	2,589
200,000	152,381	152,381	147,619	147,619	147,619	47,619	2,422	2,425	1,953	2,460	2,600

200,000	152,381	152,381	152,381	147,619	147,619	47,619	2,460	2,422	2,425	2,497	2,605
200,000	152,381	152,381	152,381	152,381	147,619	47,619	2,497	2,460	2,422	2,535	2,602
200,000	152,381	152,381	152,381	152,381	152,381	47,619	2,535	2,497	2,460	2,573	2,580
200,000	157,143	152,381	152,381	152,381	152,381	42,857	2,573	2,535	2,497	2,566	2,565
200,000	161,905	157,143	152,381	152,381	152,381	38,095	2,566	2,573	2,535	2,555	2,562
200,000	166,667	161,905	157,143	152,381	152,381	33,333	2,555	2,566	2,573	2,541	2,565
200,000	171,429	166,667	161,905	157,143	152,381	28,571	2,541	2,555	2,566	2,523	2,564
200,000	176,190	171,429	166,667	161,905	157,143	23,810	2,523	2,541	2,555	2,501	2,541
200,000	176,190	176,190	171,429	166,667	161,905	23,810	2,501	2,523	2,541	2,520	2,533
200,000	180,952	176,190	176,190	171,429	166,667	19,048	2,520	2,501	2,523	2,494	2,499
200,000	190,476	180,952	176,190	176,190	171,429	9,524	2,494	2,520	2,501	2,419	2,456
200,000	190,476	190,476	180,952	176,190	176,190	9,524	2,419	2,494	2,520	2,427	2,464
200,000	195,238	190,476	190,476	180,952	176,190	4,762	2,427	2,419	2,494	2,390	2,456
200,000	200,000	195,238	190,476	190,476	180,952	0,000	2,390	2,427	2,419	2,349	2,425
200,000	200,000	200,000	195,238	190,476	190,476	0,000	2,349	2,390	2,427	2,349	2,398
200,000	200,000	200,000	200,000	195,238	190,476	0,000	2,349	2,349	2,390	2,349	2,400
200,000	204,762	200,000	200,000	200,000	195,238	-4,762	2,349	2,349	2,349	2,304	2,361
200,000	204,762	204,762	200,000	200,000	200,000	-4,762	2,304	2,349	2,349	2,300	2,351
200,000	204,762	204,762	204,762	200,000	200,000	-4,762	2,300	2,304	2,349	2,296	2,356
200,000	204,762	204,762	204,762	204,762	200,000	-4,762	2,296	2,300	2,304	2,293	2,353
200,000	204,762	204,762	204,762	204,762	204,762	-4,762	2,293	2,296	2,300	2,289	2,331
200,000	204,762	204,762	204,762	204,762	204,762	-4,762	2,289	2,293	2,296	2,285	2,331
200,000	204,762	204,762	204,762	204,762	204,762	-4,762	2,285	2,289	2,293	2,281	2,331
200,000	200,000	204,762	204,762	204,762	204,762	0,000	2,281	2,285	2,289	2,322	2,345
200,000	204,762	200,000	204,762	204,762	204,762	-4,762	2,322	2,281	2,285	2,277	2,319
200,000	204,762	204,762	200,000	204,762	204,762	-4,762	2,277	2,322	2,281	2,274	2,326
200,000	200,000	204,762	204,762	200,000	204,762	0,000	2,274	2,277	2,322	2,315	2,349
200,000	200,000	200,000	204,762	204,762	200,000	0,000	2,315	2,274	2,277	2,315	2,355
200,000	200,000	200,000	200,000	204,762	204,762	0,000	2,315	2,315	2,274	2,315	2,328
200,000	204,762	200,000	200,000	200,000	204,762	-4,762	2,315	2,315	2,315	2,270	2,317
200,000	200,000	204,762	200,000	200,000	200,000	0,000	2,270	2,315	2,315	2,311	2,365
200,000	200,000	200,000	204,762	200,000	200,000	0,000	2,311	2,270	2,315	2,311	2,359
200,000	200,000	200,000	200,000	204,762	200,000	0,000	2,311	2,311	2,270	2,311	2,350
200,000	200,000	200,000	200,000	200,000	204,762	0,000	2,311	2,311	2,311	2,311	2,332
200,000	195,238	200,000	200,000	200,000	200,000	4,762	2,311	2,311	2,311	2,356	2,368
200,000	200,000	195,238	200,000	200,000	200,000	0,000	2,356	2,311	2,311	2,315	2,341
200,000	200,000	200,000	195,238	200,000	200,000	0,000	2,315	2,356	2,311	2,315	2,348
200,000	200,000	200,000	200,000	195,238	200,000	0,000	2,315	2,315	2,356	2,315	2,357
200,000	200,000	200,000	200,000	200,000	195,238	0,000	2,315	2,315	2,315	2,315	2,375
200,000	195,238	200,000	200,000	200,000	200,000	4,762	2,315	2,315	2,315	2,359	2,368
200,000	195,238	195,238	200,000	200,000	200,000	4,762	2,359	2,315	2,315	2,363	2,356
200,000	200,000	195,238	195,238	200,000	200,000	0,000	2,363	2,359	2,315	2,322	2,336
200,000	200,000	200,000	195,238	195,238	200,000	0,000	2,322	2,363	2,359	2,322	2,352
200,000	200,000	200,000	200,000	195,238	195,238	0,000	2,322	2,322	2,363	2,322	2,379
200,000	200,000	200,000	200,000	200,000	195,238	0,000	2,322	2,322	2,322	2,322	2,375
200,000	195,238	200,000	200,000	200,000	200,000	4,762	2,322	2,322	2,322	2,367	2,368
200,000	200,000	195,238	200,000	200,000	200,000	0,000	2,367	2,322	2,322	2,326	2,341
200,000	200,000	200,000	195,238	200,000	200,000	0,000	2,326	2,367	2,322	2,326	2,348
200,000	200,000	200,000	200,000	195,238	200,000	0,000	2,326	2,326	2,367	2,326	2,357
200,000	200,000	200,000	200,000	200,000	195,238	0,000	2,326	2,326	2,326	2,326	2,375
200,000	204,762	200,000	200,000	200,000	200,000	-4,762	2,326	2,326	2,326	2,281	2,339

200,000	200,000	204,762	200,000	200,000	200,000	0,000	2,281	2,326	2,326	2,322	2,365
200,000	200,000	200,000	204,762	200,000	200,000	0,000	2,322	2,281	2,326	2,322	2,359
200,000	200,000	200,000	200,000	204,762	200,000	0,000	2,322	2,322	2,281	2,322	2,350
200,000	200,000	200,000	200,000	200,000	204,762	0,000	2,322	2,322	2,322	2,322	2,332
200,000	204,762	200,000	200,000	200,000	200,000	-4,762	2,322	2,322	2,322	2,277	2,339
200,000	200,000	204,762	200,000	200,000	200,000	0,000	2,277	2,322	2,322	2,318	2,365
200,000	200,000	200,000	204,762	200,000	200,000	0,000	2,318	2,277	2,322	2,318	2,359
200,000	200,000	200,000	200,000	204,762	200,000	0,000	2,318	2,318	2,277	2,318	2,350
200,000	204,762	200,000	200,000	200,000	204,762	-4,762	2,318	2,318	2,318	2,274	2,317
200,000	200,000	204,762	200,000	200,000	200,000	0,000	2,274	2,318	2,318	2,315	2,365
200,000	200,000	200,000	204,762	200,000	200,000	0,000	2,315	2,274	2,318	2,315	2,359
200,000	200,000	200,000	200,000	204,762	200,000	0,000	2,315	2,315	2,274	2,315	2,350
200,000	200,000	200,000	200,000	200,000	204,762	0,000	2,315	2,315	2,315	2,315	2,332
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,315	2,315	2,315	2,315	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,315	2,315	2,315	2,315	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,315	2,315	2,315	2,315	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,315	2,315	2,315	2,315	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,315	2,315	2,315	2,315	2,353
200,000	195,238	200,000	200,000	200,000	200,000	4,762	2,315	2,315	2,315	2,359	2,368
200,000	200,000	195,238	200,000	200,000	200,000	0,000	2,359	2,315	2,315	2,318	2,341
200,000	200,000	200,000	195,238	200,000	200,000	0,000	2,318	2,359	2,315	2,318	2,348
200,000	200,000	200,000	200,000	195,238	200,000	0,000	2,318	2,318	2,359	2,318	2,357
200,000	200,000	200,000	200,000	200,000	195,238	0,000	2,318	2,318	2,318	2,318	2,375
200,000	204,762	200,000	200,000	200,000	200,000	-4,762	2,318	2,318	2,318	2,274	2,339
200,000	200,000	204,762	200,000	200,000	200,000	0,000	2,274	2,318	2,318	2,315	2,365
200,000	200,000	200,000	204,762	200,000	200,000	0,000	2,315	2,274	2,318	2,315	2,359
200,000	200,000	200,000	200,000	204,762	200,000	0,000	2,315	2,315	2,274	2,315	2,350
200,000	200,000	200,000	200,000	200,000	204,762	0,000	2,315	2,315	2,315	2,315	2,332
200,000	204,762	200,000	200,000	200,000	200,000	-4,762	2,315	2,315	2,315	2,270	2,339
200,000	200,000	204,762	200,000	200,000	200,000	0,000	2,270	2,315	2,315	2,311	2,365
200,000	200,000	200,000	204,762	200,000	200,000	0,000	2,311	2,270	2,315	2,311	2,359
200,000	200,000	200,000	200,000	204,762	200,000	0,000	2,311	2,311	2,270	2,311	2,350
200,000	200,000	200,000	200,000	200,000	204,762	0,000	2,311	2,311	2,311	2,311	2,332
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353

200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,311	2,311	2,311	2,311	2,353
200,000	204,762	200,000	200,000	200,000	200,000	-4,762	2,311	2,311	2,311	2,266	2,339
100,000	200,000	204,762	200,000	200,000	200,000	-100,000	2,266	2,311	2,311	1,367	1,116
100,000	200,000	200,000	204,762	200,000	200,000	-100,000	1,367	2,266	2,311	1,288	1,111
100,000	200,000	200,000	200,000	204,762	200,000	-100,000	1,288	1,367	2,266	1,209	1,103
100,000	195,238	200,000	200,000	200,000	204,762	-95,238	1,209	1,288	1,367	1,174	1,100
100,000	195,238	195,238	200,000	200,000	200,000	-95,238	1,174	1,209	1,288	1,098	1,111
100,000	185,714	195,238	195,238	200,000	200,000	-85,714	1,098	1,174	1,209	1,112	1,135
100,000	180,952	185,714	195,238	195,238	200,000	-80,952	1,112	1,098	1,174	1,089	1,129
100,000	176,190	180,952	185,714	195,238	195,238	-76,190	1,089	1,112	1,098	1,069	1,143
100,000	166,667	176,190	180,952	185,714	195,238	-66,667	1,069	1,089	1,112	1,098	1,162
100,000	157,143	166,667	176,190	180,952	185,714	-57,143	1,098	1,069	1,089	1,135	1,209
100,000	147,619	157,143	166,667	176,190	180,952	-47,619	1,135	1,098	1,069	1,179	1,229
100,000	138,095	147,619	157,143	166,667	176,190	-38,095	1,179	1,135	1,098	1,231	1,252
100,000	133,333	138,095	147,619	157,143	166,667	-33,333	1,231	1,179	1,135	1,245	1,283
100,000	128,571	133,333	138,095	147,619	157,143	-28,571	1,245	1,231	1,179	1,263	1,326
100,000	114,286	128,571	133,333	138,095	147,619	-14,286	1,263	1,245	1,231	1,375	1,403
100,000	109,524	114,286	128,571	133,333	138,095	-9,524	1,375	1,263	1,245	1,408	1,423
100,000	104,762	109,524	114,286	128,571	133,333	-4,762	1,408	1,375	1,263	1,445	1,435
100,000	100,000	104,762	109,524	114,286	128,571	0,000	1,445	1,408	1,375	1,486	1,465
100,000	100,000	100,000	104,762	109,524	114,286	0,000	1,486	1,445	1,408	1,486	1,516
100,000	95,238	100,000	100,000	104,762	109,524	4,762	1,486	1,486	1,445	1,531	1,551
100,000	95,238	95,238	100,000	100,000	104,762	4,762	1,531	1,486	1,486	1,535	1,564
100,000	90,476	95,238	95,238	100,000	100,000	9,524	1,535	1,531	1,486	1,584	1,595
100,000	85,714	90,476	95,238	95,238	100,000	14,286	1,584	1,535	1,531	1,636	1,601
100,000	85,714	85,714	90,476	95,238	95,238	14,286	1,636	1,584	1,535	1,647	1,606
100,000	90,476	85,714	85,714	90,476	95,238	9,524	1,647	1,636	1,584	1,614	1,589
100,000	90,476	90,476	85,714	85,714	90,476	9,524	1,614	1,647	1,636	1,621	1,626
100,000	90,476	90,476	90,476	85,714	85,714	9,524	1,621	1,614	1,647	1,629	1,653
100,000	90,476	90,476	90,476	90,476	85,714	9,524	1,629	1,621	1,614	1,636	1,650
100,000	90,476	90,476	90,476	90,476	90,476	9,524	1,636	1,629	1,621	1,644	1,628
100,000	90,476	90,476	90,476	90,476	90,476	9,524	1,644	1,636	1,629	1,652	1,628
100,000	90,476	90,476	90,476	90,476	90,476	9,524	1,652	1,644	1,636	1,659	1,628
100,000	95,238	90,476	90,476	90,476	90,476	4,762	1,659	1,652	1,644	1,622	1,614
100,000	100,000	95,238	90,476	90,476	90,476	0,000	1,622	1,659	1,652	1,581	1,611
100,000	100,000	100,000	95,238	90,476	90,476	0,000	1,581	1,622	1,659	1,581	1,628
100,000	100,000	100,000	100,000	95,238	90,476	0,000	1,581	1,581	1,622	1,581	1,630
100,000	100,000	100,000	100,000	100,000	95,238	0,000	1,581	1,581	1,581	1,581	1,605
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,581	1,581	1,581	1,581	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,581	1,581	1,581	1,581	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,581	1,581	1,581	1,581	1,583
100,000	104,762	100,000	100,000	100,000	100,000	-4,762	1,581	1,581	1,581	1,536	1,568
100,000	104,762	104,762	100,000	100,000	100,000	-4,762	1,536	1,581	1,581	1,532	1,580
100,000	104,762	104,762	104,762	100,000	100,000	-4,762	1,532	1,536	1,581	1,529	1,586
100,000	100,000	104,762	104,762	104,762	100,000	0,000	1,529	1,532	1,536	1,570	1,597
100,000	100,000	100,000	104,762	104,762	104,762	0,000	1,570	1,529	1,532	1,570	1,563
100,000	100,000	100,000	100,000	104,762	104,762	0,000	1,570	1,570	1,529	1,570	1,558
100,000	100,000	100,000	100,000	100,000	104,762	0,000	1,570	1,570	1,570	1,570	1,561
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,570	1,570	1,570	1,570	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,570	1,570	1,570	1,570	1,583
100,000	104,762	100,000	100,000	100,000	100,000	-4,762	1,570	1,570	1,570	1,525	1,568

100,000	100,000	104,762	100,000	100,000	100,000	0,000	1,525	1,570	1,570	1,566	1,595
100,000	100,000	100,000	104,762	100,000	100,000	0,000	1,566	1,525	1,570	1,566	1,588
100,000	100,000	100,000	100,000	104,762	100,000	0,000	1,566	1,566	1,525	1,566	1,580
100,000	100,000	100,000	100,000	100,000	104,762	0,000	1,566	1,566	1,566	1,566	1,561
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,566	1,566	1,566	1,566	1,583
100,000	104,762	100,000	100,000	100,000	100,000	-4,762	1,566	1,566	1,566	1,521	1,568
100,000	100,000	104,762	100,000	100,000	100,000	0,000	1,521	1,566	1,566	1,562	1,595
100,000	100,000	100,000	104,762	100,000	100,000	0,000	1,562	1,521	1,566	1,562	1,588
100,000	100,000	100,000	100,000	104,762	100,000	0,000	1,562	1,562	1,521	1,562	1,580
100,000	100,000	100,000	100,000	100,000	104,762	0,000	1,562	1,562	1,562	1,562	1,561
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,562	1,562	1,562	1,562	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,562	1,562	1,562	1,562	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,562	1,562	1,562	1,562	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,562	1,562	1,562	1,562	1,583
100,000	100,000	100,000	100,000	100,000	100,000	0,000	1,562	1,562	1,562	1,562	1,583
200,000	100,000	100,000	100,000	100,000	100,000	100,000	1,562	1,562	1,562	2,502	2,832
200,000	100,000	100,000	100,000	100,000	100,000	100,000	2,502	1,562	1,562	2,581	2,831
200,000	100,000	100,000	100,000	100,000	100,000	100,000	2,581	2,502	1,562	2,661	2,830

200,000	104,762	100,000	100,000	100,000	100,000	95,238	2,661	2,581	2,502	2,695	2,814
200,000	109,524	104,762	100,000	100,000	100,000	90,476	2,695	2,661	2,581	2,726	2,811
200,000	114,286	109,524	104,762	100,000	100,000	85,714	2,726	2,695	2,661	2,753	2,814
200,000	119,048	114,286	109,524	104,762	100,000	80,952	2,753	2,726	2,695	2,777	2,813
200,000	123,810	119,048	114,286	109,524	104,762	76,190	2,777	2,753	2,726	2,796	2,790
200,000	133,333	123,810	119,048	114,286	109,524	66,667	2,796	2,777	2,753	2,767	2,753
200,000	142,857	133,333	123,810	119,048	114,286	57,143	2,767	2,796	2,777	2,731	2,728
200,000	147,619	142,857	133,333	123,810	119,048	52,381	2,731	2,767	2,796	2,731	2,722
200,000	157,143	147,619	142,857	133,333	123,810	42,857	2,731	2,731	2,767	2,683	2,687
200,000	166,667	157,143	147,619	142,857	133,333	33,333	2,683	2,731	2,731	2,628	2,636
200,000	171,429	166,667	157,143	147,619	142,857	28,571	2,628	2,683	2,731	2,610	2,609
200,000	180,952	171,429	166,667	157,143	147,619	19,048	2,610	2,628	2,683	2,543	2,573
200,000	185,714	180,952	171,429	166,667	157,143	14,286	2,543	2,610	2,628	2,513	2,538
200,000	190,476	185,714	180,952	171,429	166,667	9,524	2,513	2,543	2,610	2,480	2,498
200,000	200,000	190,476	185,714	180,952	171,429	0,000	2,480	2,513	2,543	2,398	2,458
200,000	200,000	200,000	190,476	185,714	180,952	0,000	2,398	2,480	2,513	2,398	2,440
200,000	204,762	200,000	200,000	190,476	185,714	-4,762	2,398	2,398	2,480	2,353	2,411
200,000	204,762	204,762	200,000	200,000	190,476	-4,762	2,353	2,398	2,398	2,349	2,394
200,000	204,762	204,762	204,762	200,000	200,000	-4,762	2,349	2,353	2,398	2,345	2,356
200,000	209,524	204,762	204,762	204,762	200,000	-9,524	2,345	2,349	2,353	2,297	2,338
200,000	209,524	209,524	204,762	204,762	204,762	-9,524	2,297	2,345	2,349	2,289	2,328
200,000	204,762	209,524	209,524	204,762	204,762	-4,762	2,289	2,297	2,345	2,327	2,348
200,000	209,524	204,762	209,524	209,524	204,762	-9,524	2,327	2,289	2,297	2,278	2,318
200,000	209,524	209,524	204,762	209,524	209,524	-9,524	2,278	2,327	2,289	2,270	2,303
200,000	204,762	209,524	209,524	204,762	209,524	-4,762	2,270	2,278	2,327	2,308	2,326
200,000	209,524	204,762	209,524	209,524	204,762	-9,524	2,308	2,270	2,278	2,259	2,318
200,000	204,762	209,524	204,762	209,524	209,524	-4,762	2,259	2,308	2,270	2,296	2,318
200,000	204,762	204,762	209,524	204,762	209,524	-4,762	2,296	2,259	2,308	2,293	2,314
200,000	209,524	204,762	204,762	209,524	204,762	-9,524	2,293	2,296	2,259	2,244	2,313
200,000	204,762	209,524	204,762	204,762	209,524	-4,762	2,244	2,293	2,296	2,281	2,321
200,000	204,762	204,762	209,524	204,762	204,762	-4,762	2,281	2,244	2,293	2,277	2,336
200,000	204,762	204,762	204,762	209,524	204,762	-4,762	2,277	2,281	2,244	2,274	2,328
200,000	200,000	204,762	204,762	204,762	209,524	0,000	2,274	2,277	2,281	2,315	2,324
200,000	204,762	200,000	204,762	204,762	204,762	-4,762	2,315	2,274	2,277	2,270	2,319
200,000	204,762	204,762	200,000	204,762	204,762	-4,762	2,270	2,315	2,274	2,266	2,326
200,000	200,000	204,762	204,762	200,000	204,762	0,000	2,266	2,270	2,315	2,307	2,349
200,000	200,000	200,000	204,762	204,762	200,000	0,000	2,307	2,266	2,270	2,307	2,355
200,000	200,000	200,000	200,000	204,762	204,762	0,000	2,307	2,307	2,266	2,307	2,328
200,000	200,000	200,000	200,000	200,000	204,762	0,000	2,307	2,307	2,307	2,307	2,332
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353

200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,307	2,307	2,307	2,307	2,353
200,000	195,238	200,000	200,000	200,000	200,000	4,762	2,307	2,307	2,307	2,352	2,368
200,000	200,000	195,238	200,000	200,000	200,000	0,000	2,352	2,307	2,307	2,311	2,341
200,000	200,000	200,000	195,238	200,000	200,000	0,000	2,311	2,352	2,307	2,311	2,348
200,000	200,000	200,000	200,000	195,238	200,000	0,000	2,311	2,311	2,352	2,311	2,357
200,000	200,000	200,000	200,000	200,000	195,238	0,000	2,311	2,311	2,311	2,311	2,375
200,000	195,238	200,000	200,000	200,000	200,000	4,762	2,311	2,311	2,311	2,356	2,368
200,000	195,238	195,238	200,000	200,000	200,000	4,762	2,356	2,311	2,311	2,359	2,356
200,000	200,000	195,238	195,238	200,000	200,000	0,000	2,359	2,356	2,311	2,318	2,336
200,000	200,000	200,000	195,238	195,238	200,000	0,000	2,318	2,359	2,356	2,318	2,352
200,000	200,000	200,000	200,000	195,238	195,238	0,000	2,318	2,318	2,359	2,318	2,379
200,000	200,000	200,000	200,000	200,000	195,238	0,000	2,318	2,318	2,318	2,318	2,375
200,000	195,238	200,000	200,000	200,000	200,000	4,762	2,318	2,318	2,318	2,363	2,368
200,000	200,000	195,238	200,000	200,000	200,000	0,000	2,363	2,318	2,318	2,322	2,341
200,000	200,000	200,000	195,238	200,000	200,000	0,000	2,322	2,363	2,318	2,322	2,348
200,000	200,000	200,000	200,000	195,238	200,000	0,000	2,322	2,322	2,363	2,322	2,357
200,000	200,000	200,000	200,000	200,000	195,238	0,000	2,322	2,322	2,322	2,322	2,375
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,322	2,322	2,322	2,322	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,322	2,322	2,322	2,322	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,322	2,322	2,322	2,322	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,322	2,322	2,322	2,322	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,322	2,322	2,322	2,322	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,322	2,322	2,322	2,322	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,322	2,322	2,322	2,322	2,353
200,000	200,000	200,000	200,000	200,000	200,000	0,000	2,322	2,322	2,322	2,322	2,353
250,000	200,000	200,000	200,000	200,000	200,000	50,000	2,322	2,322	2,322	2,792	2,978
250,000	200,000	200,000	200,000	200,000	200,000	50,000	2,792	2,322	2,322	2,832	2,977
250,000	204,762	200,000	200,000	200,000	200,000	45,238	2,832	2,792	2,322	2,827	2,962
250,000	200,000	204,762	200,000	200,000	200,000	50,000	2,827	2,832	2,792	2,907	2,988
250,000	204,762	200,000	204,762	200,000	200,000	45,238	2,907	2,827	2,832	2,902	2,967
250,000	204,762	204,762	200,000	204,762	200,000	45,238	2,902	2,907	2,827	2,938	2,970
250,000	209,524	204,762	204,762	200,000	204,762	40,476	2,938	2,902	2,907	2,929	2,942
250,000	214,286	209,524	204,762	204,762	200,000	35,714	2,929	2,938	2,902	2,917	2,958
250,000	214,286	214,286	209,524	204,762	204,762	35,714	2,917	2,929	2,938	2,945	2,953
250,000	219,048	214,286	214,286	209,524	204,762	30,952	2,945	2,917	2,929	2,929	2,940
250,000	223,810	219,048	214,286	214,286	209,524	26,190	2,929	2,945	2,917	2,909	2,913
250,000	228,571	223,810	219,048	214,286	214,286	21,429	2,909	2,929	2,945	2,885	2,893
250,000	228,571	228,571	223,810	219,048	214,286	21,429	2,885	2,909	2,929	2,902	2,907
250,000	238,095	228,571	228,571	223,810	219,048	11,905	2,902	2,885	2,909	2,829	2,858
250,000	238,095	238,095	228,571	228,571	223,810	11,905	2,829	2,902	2,885	2,839	2,857
250,000	242,857	238,095	238,095	228,571	228,571	7,143	2,839	2,829	2,902	2,803	2,831
250,000	242,857	242,857	238,095	238,095	228,571	7,143	2,803	2,839	2,829	2,809	2,836
250,000	247,619	242,857	242,857	238,095	238,095	2,381	2,809	2,803	2,839	2,770	2,783
250,000	247,619	247,619	242,857	242,857	238,095	2,381	2,770	2,809	2,803	2,772	2,792
250,000	252,381	247,619	247,619	242,857	242,857	-2,381	2,772	2,770	2,809	2,729	2,760

250,000	252,381	252,381	247,619	247,619	242,857	-2,381	2,729	2,772	2,770	2,727	2,769
250,000	252,381	252,381	252,381	247,619	247,619	-2,381	2,727	2,729	2,772	2,725	2,752
250,000	257,143	252,381	252,381	252,381	247,619	-7,143	2,725	2,727	2,729	2,679	2,735
250,000	252,381	257,143	252,381	252,381	252,381	-2,381	2,679	2,725	2,727	2,718	2,739
250,000	252,381	252,381	257,143	252,381	252,381	-2,381	2,718	2,679	2,725	2,716	2,732
250,000	252,381	252,381	252,381	257,143	252,381	-2,381	2,716	2,718	2,679	2,714	2,724
250,000	257,143	252,381	252,381	252,381	257,143	-7,143	2,714	2,716	2,718	2,667	2,691
250,000	252,381	257,143	252,381	252,381	252,381	-2,381	2,667	2,714	2,716	2,706	2,739
250,000	252,381	252,381	257,143	252,381	252,381	-2,381	2,706	2,667	2,714	2,704	2,732
250,000	247,619	252,381	252,381	257,143	252,381	2,381	2,704	2,706	2,667	2,747	2,739
250,000	252,381	247,619	252,381	252,381	257,143	-2,381	2,747	2,704	2,706	2,704	2,693
250,000	252,381	252,381	247,619	252,381	252,381	-2,381	2,704	2,747	2,704	2,703	2,722
250,000	247,619	252,381	252,381	247,619	252,381	2,381	2,703	2,704	2,747	2,745	2,745
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,745	2,703	2,704	2,747	2,752
250,000	252,381	247,619	247,619	252,381	252,381	-2,381	2,747	2,745	2,703	2,704	2,710
250,000	252,381	252,381	247,619	247,619	252,381	-2,381	2,704	2,747	2,745	2,703	2,725
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,703	2,704	2,747	2,745	2,767
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,745	2,703	2,704	2,747	2,752
250,000	252,381	247,619	247,619	252,381	252,381	-2,381	2,747	2,745	2,703	2,704	2,710
250,000	252,381	252,381	247,619	247,619	252,381	-2,381	2,704	2,747	2,745	2,703	2,725
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,703	2,704	2,747	2,745	2,767
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,745	2,703	2,704	2,747	2,752
250,000	252,381	247,619	247,619	252,381	252,381	-2,381	2,747	2,745	2,703	2,704	2,710
250,000	252,381	252,381	247,619	247,619	252,381	-2,381	2,704	2,747	2,745	2,703	2,725
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,703	2,704	2,747	2,745	2,767
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,745	2,703	2,704	2,747	2,752
250,000	252,381	247,619	247,619	252,381	252,381	-2,381	2,747	2,745	2,703	2,704	2,710
250,000	252,381	252,381	247,619	247,619	252,381	-2,381	2,704	2,747	2,745	2,703	2,725
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,703	2,704	2,747	2,745	2,767
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,745	2,703	2,704	2,747	2,752
250,000	252,381	247,619	247,619	252,381	252,381	-2,381	2,747	2,745	2,703	2,704	2,710
250,000	252,381	252,381	247,619	247,619	252,381	-2,381	2,704	2,747	2,745	2,703	2,725
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,703	2,704	2,747	2,745	2,767
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,745	2,703	2,704	2,747	2,752
250,000	252,381	247,619	247,619	252,381	252,381	-2,381	2,747	2,745	2,703	2,704	2,710
250,000	252,381	252,381	247,619	247,619	252,381	-2,381	2,704	2,747	2,745	2,703	2,725
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,703	2,704	2,747	2,745	2,767
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,745	2,703	2,704	2,747	2,752
250,000	252,381	247,619	247,619	252,381	252,381	-2,381	2,747	2,745	2,703	2,704	2,710
250,000	252,381	252,381	247,619	247,619	252,381	-2,381	2,704	2,747	2,745	2,703	2,725
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,703	2,704	2,747	2,745	2,767
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,745	2,703	2,704	2,747	2,752
250,000	252,381	247,619	247,619	252,381	252,381	-2,381	2,747	2,745	2,703	2,704	2,710
250,000	252,381	252,381	247,619	247,619	252,381	-2,381	2,704	2,747	2,745	2,703	2,725
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,703	2,704	2,747	2,745	2,767
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,745	2,703	2,704	2,747	2,752
250,000	252,381	247,619	247,619	252,381	252,381	-2,381	2,747	2,745	2,703	2,704	2,710
250,000	252,381	252,381	247,619	247,619	252,381	-2,381	2,704	2,747	2,745	2,703	2,725
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,703	2,704	2,747	2,745	2,767
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,745	2,703	2,704	2,747	2,752
250,000	247,619	247,619	247,619	252,381	252,381	2,381	2,747	2,745	2,703	2,749	2,725

250,000	252,381	247,619	247,619	247,619	252,381	-2,381	2,749	2,747	2,745	2,706	2,713
250,000	247,619	252,381	247,619	247,619	247,619	2,381	2,706	2,749	2,747	2,749	2,762
250,000	247,619	247,619	252,381	247,619	247,619	2,381	2,749	2,706	2,749	2,751	2,755
250,000	247,619	247,619	247,619	252,381	247,619	2,381	2,751	2,749	2,706	2,753	2,747
250,000	247,619	247,619	247,619	247,619	252,381	2,381	2,753	2,751	2,749	2,755	2,728
250,000	247,619	247,619	247,619	247,619	247,619	2,381	2,755	2,753	2,751	2,757	2,750
250,000	247,619	247,619	247,619	247,619	247,619	2,381	2,757	2,755	2,753	2,759	2,750
250,000	247,619	247,619	247,619	247,619	247,619	2,381	2,759	2,757	2,755	2,761	2,750
250,000	247,619	247,619	247,619	247,619	247,619	2,381	2,761	2,759	2,757	2,762	2,750
250,000	252,381	247,619	247,619	247,619	247,619	-2,381	2,762	2,761	2,759	2,720	2,735
250,000	247,619	252,381	247,619	247,619	247,619	2,381	2,720	2,762	2,761	2,762	2,762
250,000	247,619	247,619	252,381	247,619	247,619	2,381	2,762	2,720	2,762	2,764	2,755
250,000	247,619	247,619	247,619	252,381	247,619	2,381	2,764	2,762	2,720	2,766	2,747
250,000	252,381	247,619	247,619	247,619	252,381	-2,381	2,766	2,764	2,762	2,723	2,713
250,000	252,381	252,381	247,619	247,619	247,619	-2,381	2,723	2,766	2,764	2,721	2,747
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,721	2,723	2,766	2,764	2,767
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,764	2,721	2,723	2,766	2,752
250,000	252,381	247,619	247,619	252,381	252,381	-2,381	2,766	2,764	2,721	2,723	2,710
250,000	252,381	252,381	247,619	247,619	252,381	-2,381	2,723	2,766	2,764	2,721	2,725
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,721	2,723	2,766	2,764	2,767
250,000	247,619	247,619	252,381	252,381	247,619	2,381	2,764	2,721	2,723	2,766	2,752
250,000	252,381	247,619	247,619	252,381	252,381	-2,381	2,766	2,764	2,721	2,723	2,710
250,000	252,381	252,381	247,619	247,619	252,381	-2,381	2,723	2,766	2,764	2,721	2,725
250,000	247,619	252,381	252,381	247,619	247,619	2,381	2,721	2,723	2,766	2,764	2,767