CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Universidade Federal
de Campina Grande

LEONARDO ALBUQUERQUE CAMPOS JÚNIOR

Centro de Engenharia
Elétrica e Informática

TRABALHO DE CONCLUSÃO DE CURSO

ESTUDO SOBRE REDES NEURAIS ARTIFICIAIS

Departamento de
Engenharia Elétrica

Campina Grande
2016

LEONARDO ALBUQUERQUE CAMPOS JÚNIOR

ESTUDO SOBRE REDES NEURAIS ARTIFICIAIS

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Telecomunicações

Orientador:
Professor Edmar Candeia Gurjão, D. Sc.

Campina Grande
2016

LEONARDO ALBUQUERQUE CAMPOS JÚNIOR

ESTUDO SOBRE REDES NEURAIS ARTIFICIAIS

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Redes Neurais Artificiais

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professor Edmar Candeia Gurjão, D. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me dar sabedoria para entender que a vida é um eterno aprendizado, e para compreender que nada teria sido possível se não fosse a Sua vontade.

Agradeço à minha família pelo apoio e amparo, se fazendo presente em todos os momentos. Em especial aos meus pais, Leonardo e Marlise, que nunca mediram esforços para me oferecer uma educação de qualidade, fornecendo-me totais condições e apoio para realizar meus sonhos, aos meus irmãos, Patrycia e Gabriel, por sempre acreditarem em meu sucesso e transmissão de confiança e amor incondicional, assim como meus avós, Abel e Esmelinda, por sempre acreditarem nesse sonho e torná-lo possível, não só com carinho e confiança, mas também com seus ensinamentos.

Agradeço à minha namorada, Larissa, por compreensão, apoio nesse período duro e por estar sempre ao meu lado no processo de conclusão da graduação.

Agradeço aos meus bons, velhos e novos amigos, que entenderam todas as minhas deixas e faltas necessárias como requisitos para a conclusão deste objetivo.

Presto sentimento de gratidão ao professor e orientador Edmar Candeia Gurjão, pela contribuição e apoio no desenvolvimento deste trabalho.

Àqueles, que não por menor importância, não foram citados, mas também tiveram grande contribuição na realização do sonho de adquirir o título de Bacharel em Engenharia Elétrica.

RESUMO

Computadores convencionais são eficientes em diversas áreas, porém ainda não apresentam desempenho tão satisfatório se comparados a um sistema real biológico (sistema visual humano, sonar de morcegos, etc). Foi pensando nas características do sistema nervoso, no qual neurônios podem estar conectados uns aos outros, recebendo e enviando estímulos, formando redes, que se desenvolveram as redes neurais artificiais. O grande trunfo desse modelo computacional é a eficiência. Tal diferencial deve-se ao fato desse modelo proporcionar um aprendizado em tempo real, caracterizado pelas as conexões entre os neurônios. Tais fatos culminam com o estímulo por um estudo dessas redes. Sob esse contexto, são abordados dois tipos desses modelos de redes neurais: *perceptron* e *Adaline*, que podem ser empregados em aplicações de diversas áreas. Para isto, utilizou-se o *software* MATLAB como ferramenta de simulação.

Palavras-chave: Redes neurais artificiais, aprendizado, modelo computacional, técnicas de simulação, *perceptron* e *Adaline*.

ABSTRACT

Conventional computers are efficient in several areas. However, they do not yet perform so satisfactorily when compared to real biological system (human visual system, bats' sonar, etc). The artificial neural network was developed by the idea of gathering in common characteristics of the human being nervous system. The fact of the neurons be connected to each other, receiving and sending stimuli, creates a network. The great trump of this computational method is efficiency. Such differential is due to the fact that this model provides real time learning, featured by the neurons' connections. All those facts results in a reason for studying this network. In this context, two artificial neural networking models are discussed, *perceptron* and *Adaline*, which are applied in different areas. Therefore, MATLAB software was used as simulation tool.

Keywords: Artificial neural networking, real time learning, computational method, simulation techniques, *perceptron* and *Adaline*.

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação de um neurônio artificial de acordo com a proposta apresentada por McCulloch e Pitts.. 15
Figura 2 – Exemplos de funções de ativação. 16
Figura 3 – Rede *feedforward* de uma única camada. 17
Figura 4 – Rede *feedforward* de múltiplas camada..... 17
Figura 5 – Rede recorrente..... 17
Figura 6 – Aprendizado supervisionado..... 17
Figura 7 – Topologia de um *perceptron* simples com uma única saída..... 19
Figura 8 – GUI para criação da rede *perceptron*. 22
Figura 9 – Abas para criação de rede *perceptron* e seus dados..... 23
Figura 10 – Aba de treinamento da rede *perceptron*..... 24
Figura 11 – Aba com os resultados do treinamento da rede *perceptron*..... 24
Figura 12 – Aba para gerar o *design* da rede *Adaline*. 25
Figura 13 – Aba para criação da rede *Adaline* a ser treinada. 27
Figura 14 – Treinamento para uma rede *Adaline*. 28
Figura 15 – Rede *Adaline* – projeto. 28
Figura 16 – Rede *Adaline* – treinamento com taxa de aprendizado 0,01. 29
Figura 17 – Rede *Adaline* – treinamento com taxa de aprendizado 0,1..... 29
Figura 18 – Rede *Adaline* – treinamento com taxa de aprendizado 0,5..... 30
Figura 19 – Rede *Adaline* – treinamento com taxa de aprendizado 1,0..... 31
Figura 20 – Rede *perceptron* – saídas desejada e corrente, bem como seu erro..... 32
Figura 21 – Rede *Adaline* (taxa de aprendizado = 0,01) – saídas desejada e corrente, bem como seu erro 32
Figura 22 – Rede *Adaline* (taxa de aprendizado = 0,1) – saídas desejada e corrente, bem como seu erro 32
Figura 23 – Rede *Adaline* (taxa de aprendizado = 0,5) – saídas desejada e corrente, bem como seu erro 32
Figura 24 – Rede *Adaline* (taxa de aprendizado = 1,0) – saídas desejada e corrente, bem como seu erro 32

LISTA DE ABREVIATURAS E SIGLAS

RNAs	Redes Neurais Artificiais
GUI	Graphical User Interface
MCP	McCulloch e Pitts

SUMÁRIO

1 Introdução 10
1.1 Objetivos 10
1.2 Estrutura do Trabalho 11
2 Embasamento Teórico 12
2.1 Redes Neurais Artificiais 12
2.1.1 Neurônios Artificiais: Modelo MCP 12
2.1.2 Funções de Ativação 13
2.1.3 Arquiteturas 14
2.1.4 Aprendizado 16
2.2 Perceptron 18
2.2.1 Porta de Limiar Linear 18
2.2.2 O algoritmo de aprendizado do *perceptron* 19
2.3 Adaline 20
2.3.1 Descrição do Modelo 21
3 Criação de RNAs em MATLAB 22
3.1 Perceptron 22
3.2 Adaline 25
4 Análise dos resultados 31
5 Conclusão 34
Referências 35

1 INTRODUÇÃO

As Redes Neurais Artificiais (RNAs) fundamentam-se nos estudos sobre a estrutura do cérebro humano para tentar emular sua forma inteligente de processar informação. Alguns estudos da neurofisiologia consideram que a riqueza computacional do cérebro humano está associada ao grande número de neurônios, interconectados por uma rede complexa de sinapses [1].

Devido ao elevado número de neurônios (da ordem de 10^{11}) e sinapses (em média 10^{3}) o cérebro humano é capaz de executar rapidamente certas funções, como reconhecer fisionomias e sons, os quais computadores convencionais não conseguem realizar com o mesmo desempenho.

As RNAs são ferramentas de Inteligência Artificial que possuem a capacidade de se adaptar e de aprender a realizar certa tarefa ou comportamento, a partir de um conjunto de exemplos dados. A aplicação das redes neurais nas tarefas de processamento de imagens é bastante atrativa dada às características deste tipo de ferramentas, tais como: robustez, generalização, paralelismo e tolerância ao ruído [2].

As RNAs se aplicam basicamente a problemas em que existam dados, experimentais ou gerados por meio de modelos, por meio dos quais a rede adaptará os seus pesos visando à execução de uma determinada tarefa. As principais tarefas nas quais as redes neurais se aplicam são a classificação, categorização (agrupamento ou *clustering*), aproximação, previsão e otimização. Dentro dessas áreas podemos encontrar aplicações de RNAs nos mais diversos setores como: setor financeiro, setor elétrico, automação e controle, modelagem de sistemas industriais, bioinformática, comércio eletrônico, telecomunicações, etc [3].

1.1 OBJETIVOS

O objetivo principal deste trabalho de conclusão de curso é o estudo de redes neurais artificiais a fim de adquirir conhecimento técnico sobre o tema. A priori, realizar estudo de alguns modelos computacionais com o auxílio da ferramenta MATLAB

R2008b, promover simulações de um exemplo didático para cada modelo, analisar os resultados obtidos, comparando os métodos e sugerindo aplicações.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado em cinco capítulos. O primeiro capítulo destina-se à parte introdutória. No segundo capítulo é feito o embasamento teórico do trabalho, onde são abordados os temas essenciais para o entendimento do tema proposto. O terceiro capítulo destina-se a explicar as etapas das simulações realizadas no MATLAB para cada tipo de modelo. No quarto capítulo, análise dos resultados obtidos de simulação. No quinto capítulo as conclusões e considerações finais são apresentadas.

2 EMBASAMENTO TEÓRICO

No decorrer deste capítulo serão apresentados alguns tópicos necessários para uma melhor compreensão da escolha referente aos métodos utilizados nas simulações das redes, tais como Redes Neurais Artificiais, modelo *perceptron* e modelo *Adaline*.

2.1 REDES NEURAIS ARTIFICIAIS

As Redes Neurais Artificiais são técnicas computacionais que utilizam modelos matemáticos inspirados na estrutura neural de organismos inteligente, como os humanos, capazes de adquirir conhecimento através da experiência. O objetivo do desenvolvimento das RNAs é justificado por uma antiga aspiração do homem: criar máquinas que simulem a inteligência do cérebro humano [4].

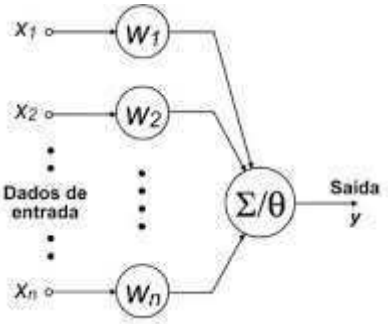
2.1.1 NEURÔNIOS ARTIFICIAIS: MODELO MCP

O modelo de neurônio artificial proposto por Warren McCulloch e Walter Pitts em 1943, ilustrado na Figura 1, é uma simplificação do que se sabia na época a respeito do neurônio biológico. Sua descrição matemática resultou em um modelo com *n* terminais de entrada (dendritos) que recebem os valores $x_1, x_2, \dots, x_n$ (que representam as ativações dos neurônios anteriores) e apenas um terminal de saída *y* (representando o axônio). Para representar o comportamento das sinapses, os terminais de entrada do neurônio têm pesos acoplados $w_1, w_2, \dots, w_n$, cujos valores podem ser positivos ou negativos, dependendo de as sinapses correspondentes serem inibitórias ou excitatórias.

Um neurônio biológico dispara quando a soma dos impulsos que ele recebe ultrapassa o seu limiar de excitação (*threshold*). Esse comportamento do neurônio biológico, por sua vez, é representado no modelo artificial por um mecanismo simples, que faz a soma dos valores $x_i w_i$ recebidos pelo neurônio (soma ponderada) e decide se o neurônio deve ou não disparar (saída igual a 1 ou a 0), comparando a soma obtida ao limiar do neurônio. No modelo MCP, a ativação do neurônio é obtida através da

aplicação de uma “função de ativação”, que ativa ou não a saída, dependendo do valor da soma ponderada das suas entradas [3].

Figura 1: Representação de um neurônio artificial de acordo com a proposta apresentada por McCulloch e Pitts.



Fonte: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1806-11172011000100009>, 2016.

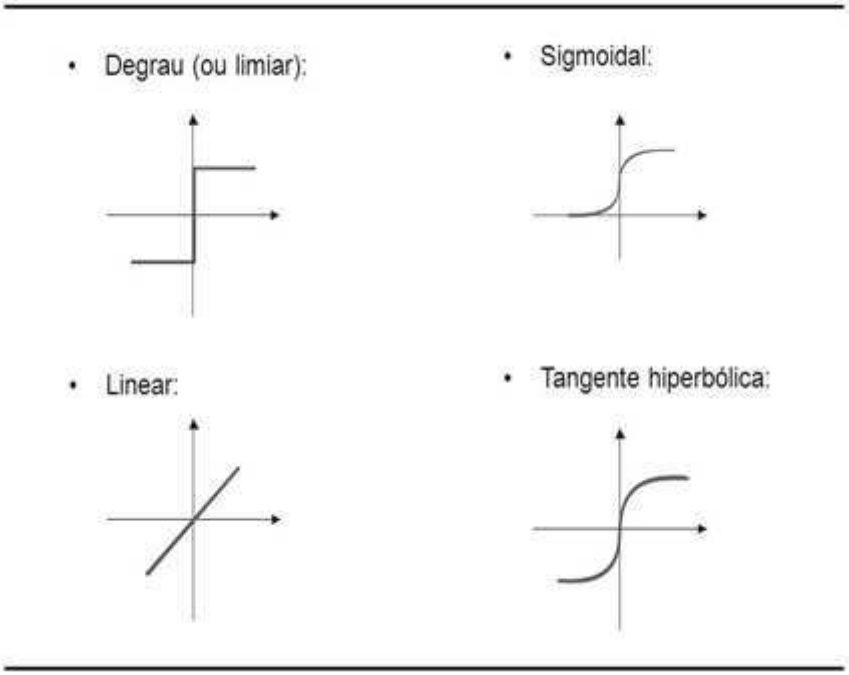
2.1.2 FUNÇÕES DE ATIVAÇÃO

A função de ativação é responsável por gerar a saída *y* do neurônio a partir dos valores dos vetores de peso $\boldsymbol{w} = (w_1, w_2, \dots, w_n)^T$ e de entrada $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T$. Algumas funções de ativação conhecidas são: sigmoidal, tangente hiperbólica, degrau (limiar) e linear, representadas na Figura 2. A abordagem sobre as funções de ativação nesta seção será em cima das funções limiar e linear. A função de ativação de um neurônio MCP, apresentada na Equação 1 é do tipo degrau deslocado do limiar de ativação θ em relação à origem, ou seja, a saída *y* será 1 para $\sum_{i=1}^{n} x_i w_i \geq \theta$ e 0 para $\sum_{i=1}^{n} x_i w_i < \theta$.Dependendo do problema a ser abordado, neurônios com funções de ativação lineares, como a apresentada na Equação 2 podem também ser utilizadas.

$$f(u) = \begin{cases} 1, & \sum_{i=1}^{n} x_i w_i \geq \theta \\ 0, & \sum_{i=1}^{n} x_i w_i < \theta \end{cases} \quad (1)$$

$$f(u) = u \quad (2)$$

Figura 2: Exemplos de funções de ativação.



Fonte: <http://slideplayer.com.br/slide/292589/>, 2016

2.1.3 ARQUITETURAS

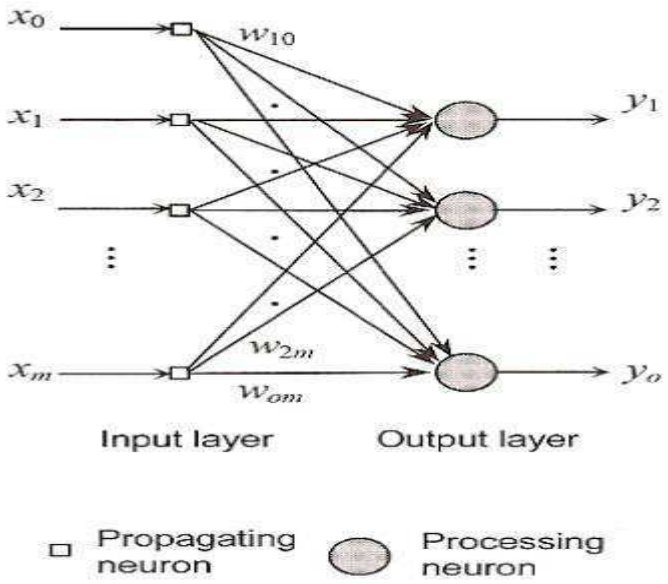
Um neurônio visto de forma isolada não tem função alguma. Contudo se um arranjo de neurônios, com função específica e direção de suas conexões sinápticas definidas caracterizam uma arquitetura de redes neurais. Tais arquiteturas são pensadas no intuito de solução de problemas, ou seja, são definidas para realização de tarefas.

A rede é dividida em três partes: camada de entrada, camadas intermediárias ou ocultas e camada de saída. A camada de entrada recebe o estímulo do ambiente. Nas camadas intermediárias são realizadas a extração das características e interferência. A camada de saída é responsável pela produção e apresentação do resultado final [7].

A arquitetura de redes neurais é dividida em três tipos básicos: redes *feedforward* de uma única camada, redes *feedforward* de múltiplas camadas e redes recorrentes. A terminologia *feedforward* diz respeito ao sinal de entrada que sempre se propaga no sentido da saída, nunca ao contrário.

Redes *feedforward* de uma única camada, representada na Figura 3, são o tipo mais simples, uma camada de neurônios de entrada que alimenta uma camada de neurônios de saída.

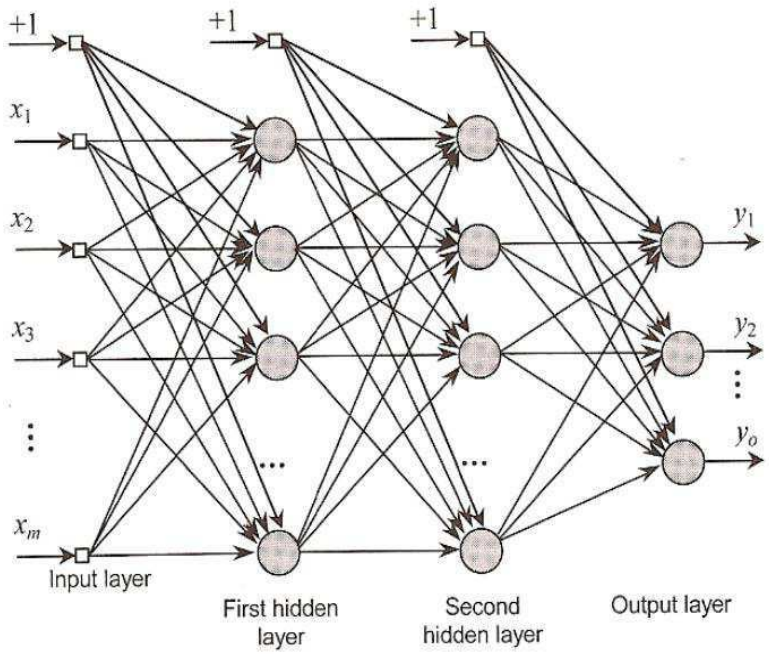
Figura 3: Rede *feedforward* de uma única camada.



Fonte: <http://www.fabriciobreve.com/material/cin/CIN-06-RedesNeuraisArtificiais.pdf>, 20016.

As redes *feedforward* de múltiplas camadas, ilustradas na Figura 4, diferem-se das de uma única camada por possuir uma ou mais camadas intermediárias. A saída de uma camada é usada como entrada para a camada seguinte.

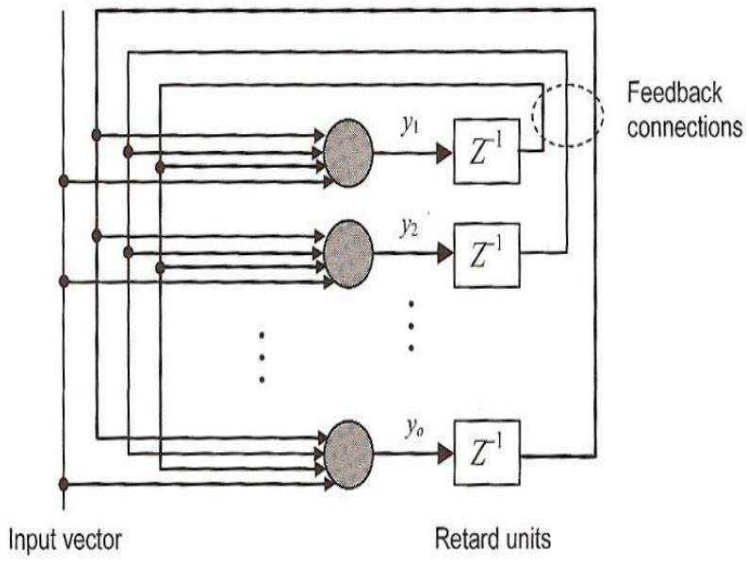
Figura 4: Rede *feedforward* de múltiplas camadas.



Fonte: http://www.fabriciobreve.com/material/cin/CIN-06-RedesNeuraisArtificiais.pdf, 2016.

As redes recorrentes, representadas na Figura 5, podem consistir de uma única camada de neurônios com cada neurônio enviando sua saída de volta para a entrada de outros neurônios. Tem ao menos um *loop* recorrente.

Figura 5: Rede recorrente.



Fonte: <http://www.fabriciobreve.com/material/cin/CIN-06-RedesNeuraisArtificiais.pdf>, 2016.

2.1.4 APRENDIZADO

Uma das características mais importantes das RNAs é a sua capacidade de aprender e generalizar por meio de exemplos. A etapa de aprendizado consiste em um processo iterativo de ajuste de parâmetros da rede, os pesos das conexões, que guardam ao final do processo o conhecimento que a rede adquiriu do ambiente externo [3].

É importante ressaltar que o conceito de aprendizado está relacionado com a melhoria do desempenho da rede segundo algum critério preestabelecido. Um critério que normalmente é levado em conta é o erro. Assim, quando algoritmos de correção de erros são utilizados no treinamento de RNAs, espera-se que o erro diminua à medida que o aprendizado prossiga.

De forma genérica, o valor do vetor de pesos $\boldsymbol{w}(t+1)$ no instante t+1 pode ser escrito conforme a Equação 3,

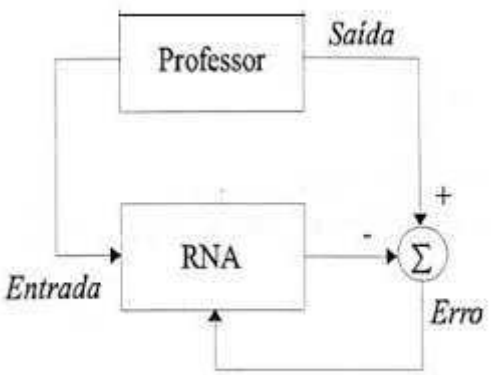
$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \Delta\boldsymbol{w}(t) \qquad (3)$$

onde $\boldsymbol{w}(t)$ e $\boldsymbol{w}(t+1)$ representam os valores dos pesos nos instantes t e t+1, respectivamente, e $\Delta\boldsymbol{w}(t)$ é o ajuste aplicado aos pesos.

Os algoritmos de aprendizagem diferem, basicamente, na forma como sua função limiar é calculada. Há vários algoritmos diferentes para treinamento de redes neurais, podendo os mesmos ser agrupados em dois paradigmas principais: aprendizado supervisionado e aprendizado não-supervisionado.

Aprendizado supervisionado implica necessariamente a existência de um supervisor ou professor externo, o qual é responsável por estimular as entradas da rede por meio de padrões de entrada e observar a saída calculada pela mesma, comparando-a com a saída desejada. Como a resposta da rede é função dos valores atuais do seu conjunto de pesos, estes são ajustados de forma a aproximar a saída da rede da saída desejada. A Figura 6 ilustra uma representação esquemática do aprendizado supervisionado. Para cada padrão de entrada, a rede tem sua saída corrente comparada com a saída desejada pelo supervisor, que fornece informações sobre a direção de ajuste dos pesos [3].

Figura 6: Aprendizado supervisionado.



Fonte: <http://handcorp.blogspot.com.br/2011/02/aprendizagem-artificial.html>, 2016.

No aprendizado não-supervisionado, não há um professor ou supervisor externo para acompanhar o processo de aprendizado. Neste esquema apenas os padrões de entrada estão disponíveis para a rede. Durante o aprendizado os padrões de entrada são apresentados continuamente à rede e a presença de regularidades nesses dados faz com que o aprendizado seja possível.

O aprendizado supervisionado se aplica a problemas em que se deseja obter um mapeamento entre padrões de entrada e saída. Problemas como classificação, envolvem a tarefa de atribuir a um padrão desconhecido uma entre várias classes conhecidas. No caso da classificação, exemplos de padrões são apresentados às entradas e as classes

correspondentes são apresentadas às saídas da rede durante o processo de aprendizado. A rede deverá adaptar os seus pesos de forma a mapear as relações entre padrões de entrada e classes correspondentes de saída, tendo por base os dados do conjunto de treinamento. Assim, a classificação envolve, após o treinamento, atribuir uma das classes conhecidas a um padrão qualquer de entrada.

2.2 PERCEPTRON

O primeiro modelo de rede neural proposto foi o *perceptron* de uma única camada, por Frank Rosenblatt em 1958. Trata-se de uma RNA simples: constituído de uma camada de entrada e uma camada de saída. A cada entrada existe um peso relacionado, sendo que o valor de saída será a soma dos produtos de cada entrada pelo seu respectivo peso. O neurônio possui um comportamento tudo ou nada, logo será necessário estabelecer uma função limiar que defina quando o neurônio estará ativo ou em repouso [5].

2.2.1 PORTA DE LIMIAR LINEAR

A porta de limiar linear possui uma definição semelhante ao neurônio MCP descrito na seção 2.1.1, conforme ilustrado na Equação 4. No entanto, as suas entradas estão restritas a variáveis booleanas, enquanto as entradas de um neurônio MCP podem assumir qualquer valor real. De uma maneira geral, os vetores de entrada $\boldsymbol{x}$ de uma porta limiar são tais que $\boldsymbol{x} \in \{0,1\}^n$ e a função $y = f(x)$ executada pela porta faz o mapeamento *y*: $\{0,1\}^n \rightarrow \{0,1\}$ para $\mathbf{w} \in R^n$.

$$y = f(u) = \begin{cases} 1, & u = \sum_{i=1}^{n} x_i w_i \geq \theta \\ 0, & u = \sum_{i=1}^{n} x_i w_i < \theta \end{cases} \qquad (4)$$

Na forma descrita na Equação 4, as portas de limiar lineares estão restritas à solução de problemas que sejam linearmente separáveis, ou seja, a problemas cuja solução possa ser obtida pela separação de duas regiões por meio de uma reta (ou um hiperplano, para o caso de *n*-dimensional). Em outras palavras, o espaço de entrada da porta de limiar é dividida em vetores $\boldsymbol{x}$ que levam a saída *y* para 1 e outros que levam *y* para 0. Esses vetores ficam em lados opostos de uma superfície de separação, sendo que

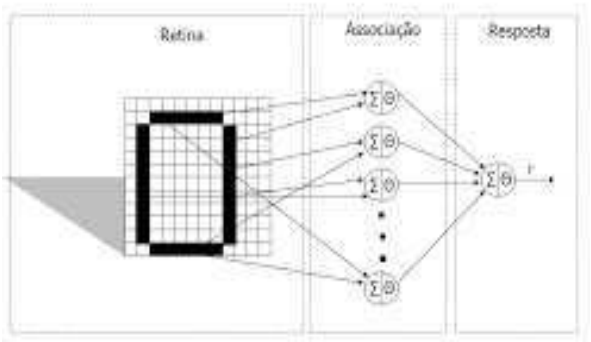
para $\theta \geq 0$, os pontos que correspondem a $y = 1$ ficam acima e aqueles que correspondem a $y = 0$ ficam abaixo da superfície.

Existem também portas de limiar quadráticas e portas de limiar polinomiais, porém essas não fazem parte do escopo desse trabalho [3].

2.2.2 O ALGORITMO DE APRENDIZADO DO *PERCEPTRON*

Uma RNA é composta por um conjunto de neurônios com capacidade de processamento local, uma topologia de conexão que define a forma como estes neurônios estão conectados e uma regra de aprendizado. O *perceptron* é formado por neurônios MCP e pela topologia de rede descrita na Figura 7. Nesta seção será descrita a regra de aprendizado do *perceptron*, que permite a adaptação dos seus pesos de forma que a rede execute uma determinada tarefa.

Figura 7: Topologia de um *perceptron* simples com uma única saída.



Fonte: <http://slideplayer.com.br/slide/295667/>, 2016.

De uma maneira geral, durante o processo de adaptação ou aprendizado, o que se deseja obter é o valor do incremento $\Delta \boldsymbol{w}(n)$ a ser aplicado ao vetor de pesos $\boldsymbol{w}(n)$ de tal forma que o seu valor atualizado

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \Delta \mathbf{w}(n) \quad (5)$$

esteja mais próximo da solução desejada do que $\boldsymbol{w}(n)$. Portanto, os algoritmos de aprendizado em RNAs visam ao desenvolvimento de técnicas para a obtenção do valor de $\Delta \boldsymbol{w}(n)$ mais apropriado para a obtenção da solução do problema em questão [3].

De acordo com o Teorema da Convergência, a atualização dos pesos pela Equação 6 leva sempre a uma solução, caso as classes em questão sejam linearmente separáveis.

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \eta e \boldsymbol{x}, \quad (6)$$

em que:

$\boldsymbol{w}(n+1)$ é o vetor peso seguinte;

$\boldsymbol{w}(n)$ é o vetor peso atual;

η é a taxa de aprendizado;

e é o erro calculado entre saída desejada e saída atual;

$\boldsymbol{x}$ é o vetor entrada.

O algoritmo de treinamento do *perceptron* sempre chega, em um tempo finito, a uma solução para o problema de separação de duas classes linearmente separáveis [3]. A seguir é apresentado um exemplo de aplicação deste algoritmo a um problema de reconhecimento de padrões. De um modo geral, o algoritmo de treinamento para um neurônio simples de um *perceptron* simples pode ser descrito da seguinte forma:

i. Inicializar η;
ii. Inicializar o vetor de pesos $\boldsymbol{w}$ com valores aleatórios;
iii. Aplicar a regra de atualização dos pesos (Equação 6) para todos os p pares (x^i, y_d^i) do conjunto de treinamento $\Gamma = \{(x^i, y_d^i)\}_{i=1}^{p}$;
iv. Repetir o passo anterior até que $e = 0$ para todos os p elementos de Γ.

2.3 ADALINE

O modelo *Adaline* (*ADAptive LINEar*) surgiu na literatura quase que simultaneamente com o *perceptron* ao final da década de 1950. Ambos os modelos são baseados em elementos de processamento que executam operações sobre a soma ponderada de suas entradas. Estas operações podem ser não-lineares do tipo degrau para o *perceptron* ou puramente lineares para o *Adaline*. O modelo surgiu no trabalho de Bernard Widrow e Marcian Hoff, que enfocou a descrição do *Adaline* na construção de filtros lineares.

O algoritmo de treinamento do *Adaline* se baseia na magnitude e no sinal do gradiente do erro para obter a direção e o valor do ajuste $\Delta \boldsymbol{w}$ a ser aplicado ao vetor de pesos. Este algoritmo, conhecido como Regra Delta, deu origem anos mais tarde ao primeiro algoritmo para treinamento de redes *perceptron* de múltiplas camadas: o *back-propagation* [3].

2.3.1 DESCRIÇÃO DO MODELO

O modelo *Adaline* é caracterizado pela utilização de uma função de ativação linear

$$f(u) = u \qquad (7)$$

onde $u = \sum_{i=0}^{n} x_i w_i$ é a soma das entradas x_i ponderadas pelos pesos correspondentes w_i.

Embora não exista limiar de ativação para o modelo *Adaline*, o termo de polarização θ é também considerado ao calcular a saída *y*. Nesse caso, o parâmetro θ, corresponde a um grau de liberdade a mais para o neurônio, que resulta em deslocamento da função de ativação em relação à origem do sistema de coordenadas. A Equação 7 pode também ser reescrita conforme a Equação 8, descrevendo-se a saída *y* como uma função do vetor de entrada $\boldsymbol{x}$, tendo o vetor de pesos $\boldsymbol{w}$ como vetor de parâmetros (incluindo-se $\theta = w_0$) para o cálculo de saída.

$$y = f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}.\boldsymbol{x} \quad (8)$$

De maneira geral, a saída *y* corresponde a uma combinação linear das entradas x_i, em que os pesos da combinação linear são obtidos através do treinamento, podendo estes assumir quaisquer valores reais. Em outras palavras, para um *Adaline* de *n* entradas, a saída *y* é calculada através da expressão $y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$, em que os parâmetros w_i são resultado do treinamento [3].

A principal diferença entre os modelos *perceptron* e *Adaline* é que o primeiro se caracteriza como separador linear e o segundo como aproximador linear de funções. Esses modelos de uma única camada se aplicam a problemas de natureza distinta, sendo o *perceptron* utilizado na classificação de padrões e o *Adaline* na aproximação de funções.

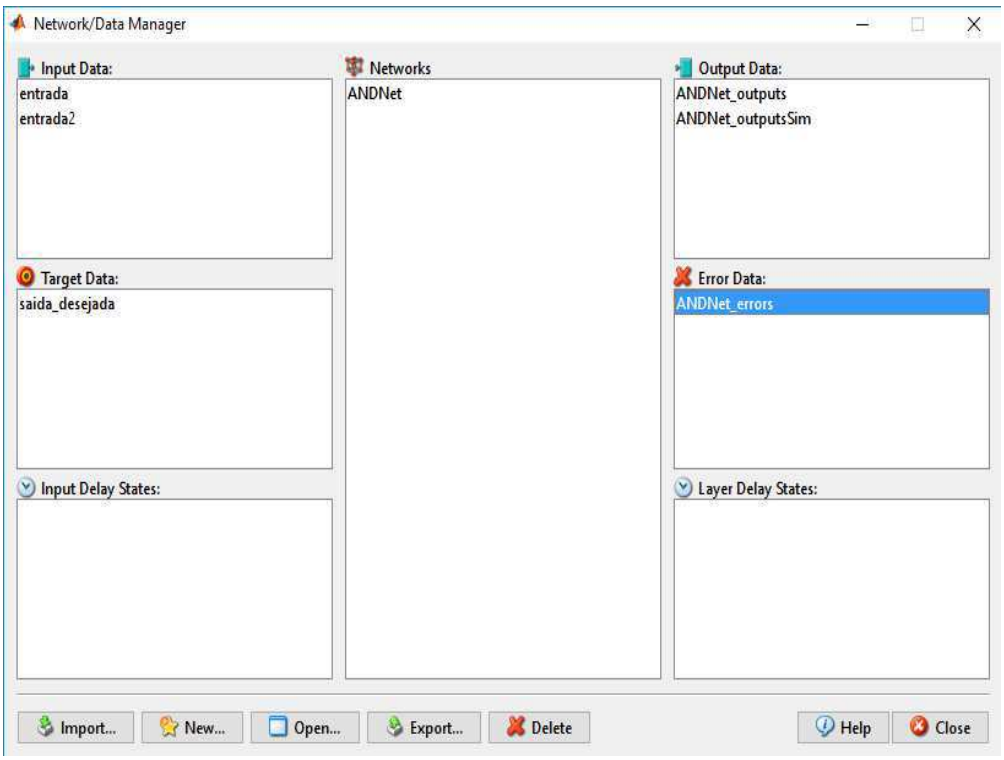
3 CRIAÇÃO DE RNAS EM MATLAB

Nesta seção alguns modelos de redes neurais são simulados a partir do uso do *software* MATLAB R2008b. Dessa forma, um roteiro para a criação dessas RNAs foi elaborado a fim de descrever cada etapa do processo.

3.1 PERCEPTRON

A partir de uma interface gráfica do usuário (*graphical user interface* – GUI), é possível criar uma rede *perceptron*, visualizá-la, treiná-la, simulá-la, etc. Através do comando *nntool*, a janela *Network/Data Manager* é iniciada como representado pela Figura 8.

Figura 8: GUI para criação da rede *perceptron*.

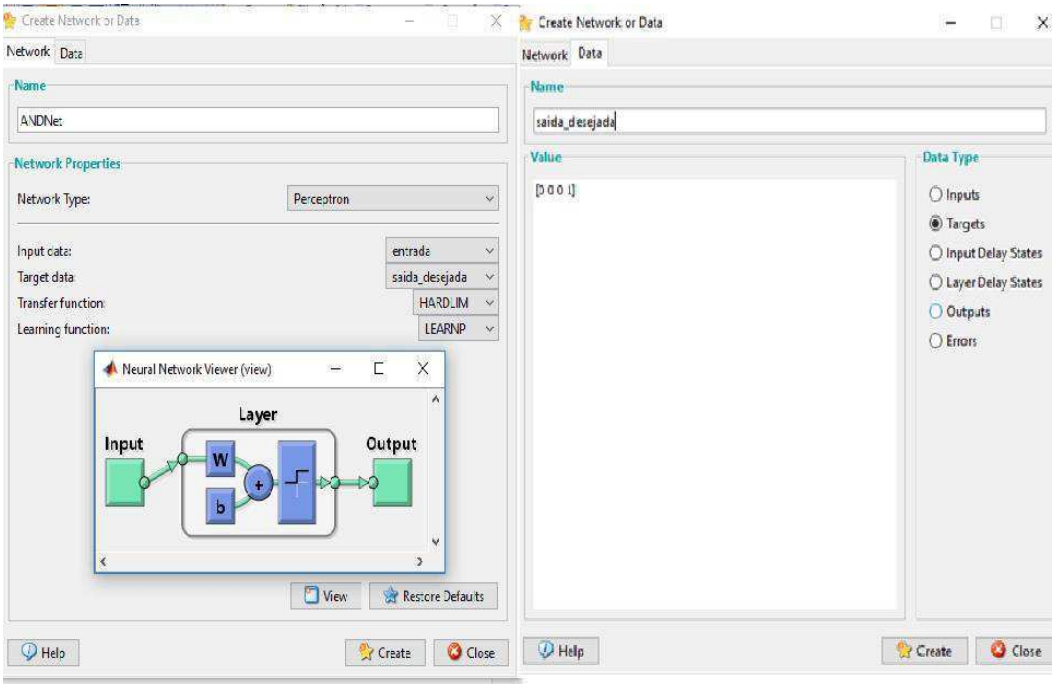


Fonte: O próprio autor, 2016.

Seguem abaixo algumas etapas importantes para a criação da RNA:

i. Definir os vetores de entrada da rede, bem como sua saída desejada como ilustrado na Figura 9. Os vetores de entrada a serem usados são sugeridos pelo próprio software na seção *HELP*. Assim, $\boldsymbol{x_0} = [0\ 0\ 1\ 1]$ e $\boldsymbol{x_1} = [0\ 1\ 0\ 1]$ e a saída desejada $y_d = [0\ 0\ 0\ 1]$. É importante ressaltar que existe uma gama de opções para a escolha da rede, assim como o números de camadas e o número de neurônios por camada. Como tratamos da rede *perceptron*, só pode-se usar as entradas previamente inseridas e saída desejada, não podendo variar a quantidade de neurônios (definidos ao se inserir os dados de entrada) e camadas, por se tratar de uma camada simples;

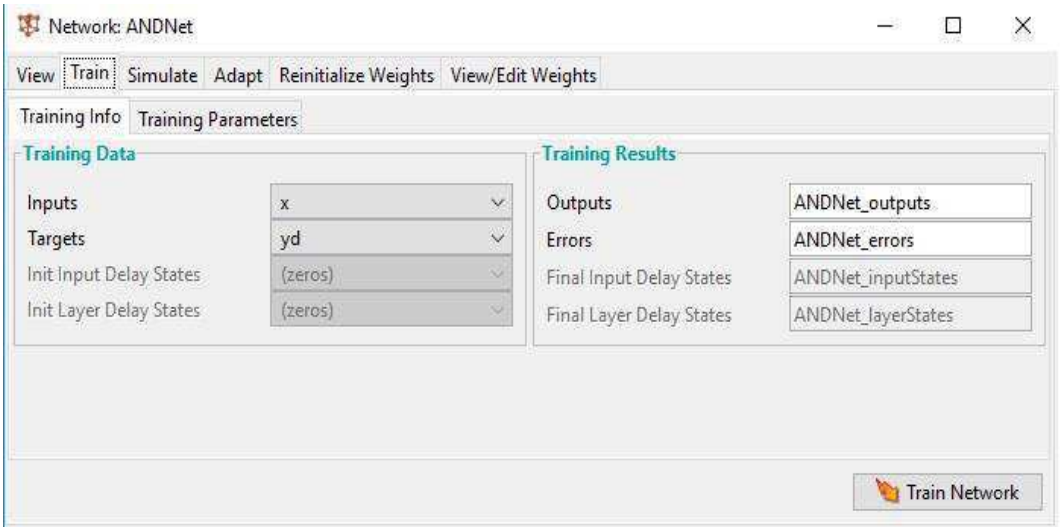
Figura 9: Abas para criação de rede *perceptron* e seus dados.



Fonte: O próprio autor, 2016.

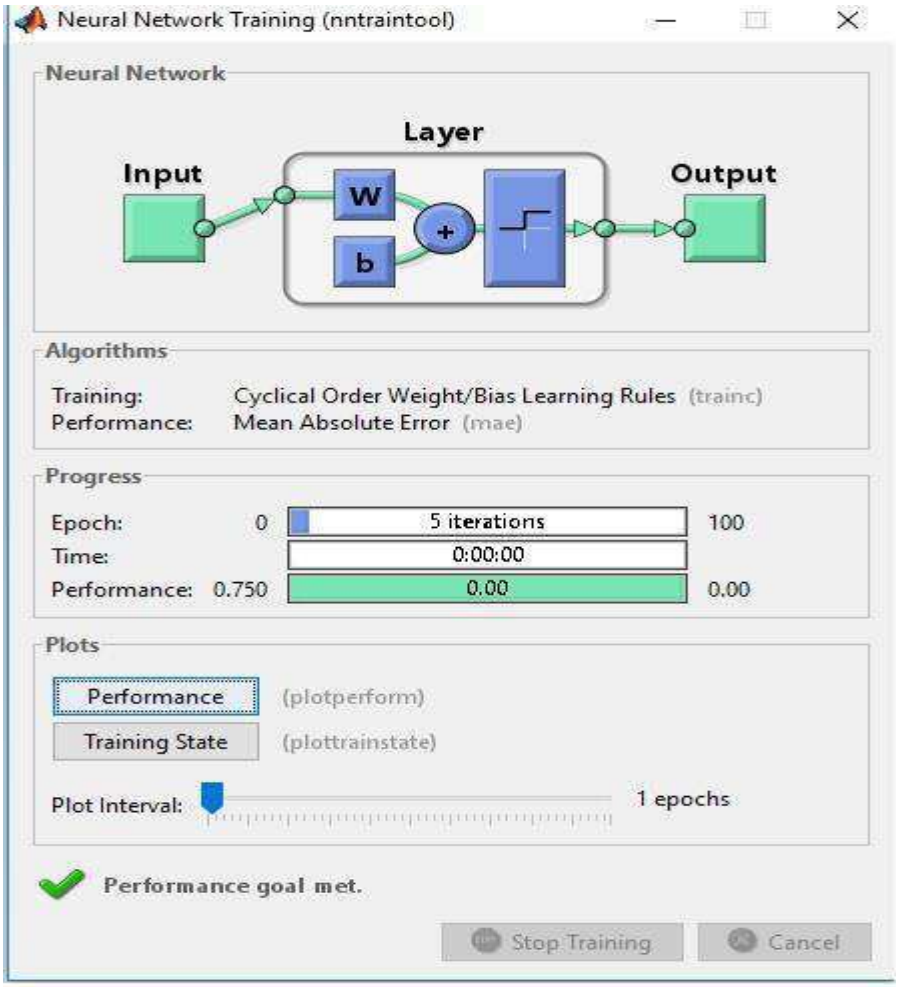
ii. Para treinar a rede, deve-se informar os dados de entrada e saída desejada, como apresentado na Figura 10. Ao realizar esse passo, obtém-se uma breve análise da rede, informando o número de iterações que a rede precisa para se atingir erro nulo, tempo de treinamento, dentre outros, ilustrado na Figura 11.

Figura 10: Aba de treinamento da rede *perceptron*.



Fonte: O próprio autor, 2016.

Figura 11: Aba com os resultados do treinamento da rede *perceptron*.



Fonte: O próprio autor, 2016.

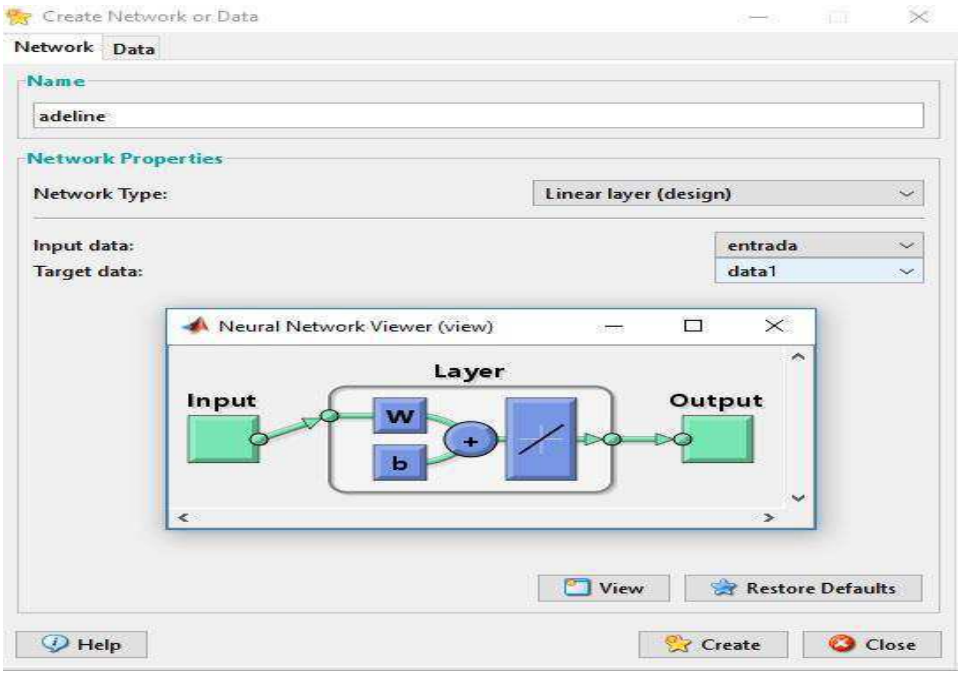
3.2 ADALINE

A partir de uma interface gráfica do usuário (*graphical user interface* – GUI), é possível criar uma rede *Adaline*, visualizá-la, treiná-la, simulá-la, etc. Através do comando *nntool*, a janela *Network/Data Manager* é iniciada como representado pela Figura 5.

Seguem abaixo algumas etapas importantes para a criação das redes neurais:

a) Definir os vetores de entrada da rede, bem como sua saída desejada como ilustrado na Figura 12. Os vetores de entrada a serem usados são sugeridos pelo próprio software na seção *HELP newlin*, o qual é o comando para ativar esse tipo de rede. Assim, $\boldsymbol{x_0} = [0\ 0\ 1\ 1]$ e $\boldsymbol{x_1} = [0\ 1\ 0\ 1]$ e a saída desejada $y_d = [0\ 0\ 0\ 1]$. É importante ressaltar que existe uma gama de opções para a escolha da rede, assim como o números de camadas e o número de neurônios por camada. Como tratamos da rede *Adaline*, só pode-se usar as entradas previamente inseridas e saída desejada, não podendo variar a quantidade de neurônios (definidos ao se inserir os dados de entrada) e camadas, por se tratar de uma camada simples;

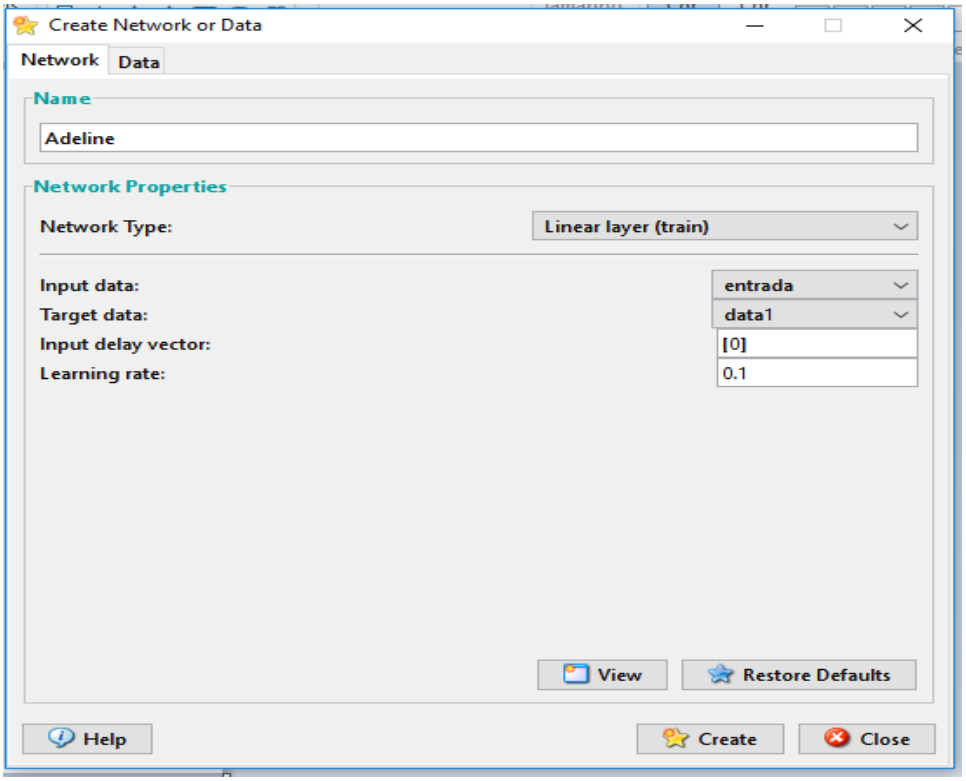
Figura 12: Aba para gerar o *design* da rede *Adaline*.



Fonte: O próprio autor, 2016.

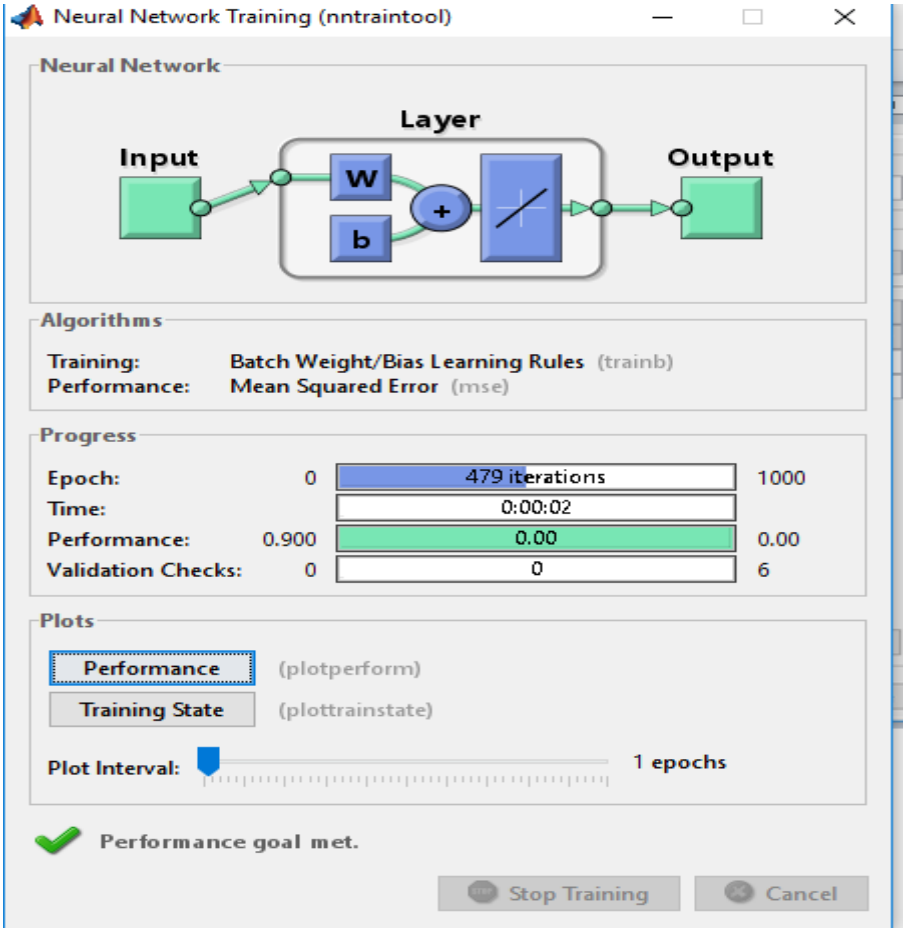
b) Para treinar a rede, inicialmente deve-se criar outra rede denominada *Linear layer (train)*. Em seguida, informar os dados de entrada e saída desejada, como apresentado na Figura 13. Ao realizar esse passo, obtemos uma breve análise da rede, informando o número de iterações(finitas) que a rede precisa, tempo de treinamento, dentre outros, ilustrado na Figura 14.

Figura 13: Aba para criação da rede *Adaline* a ser treinada.



Fonte: O próprio autor, 2016.

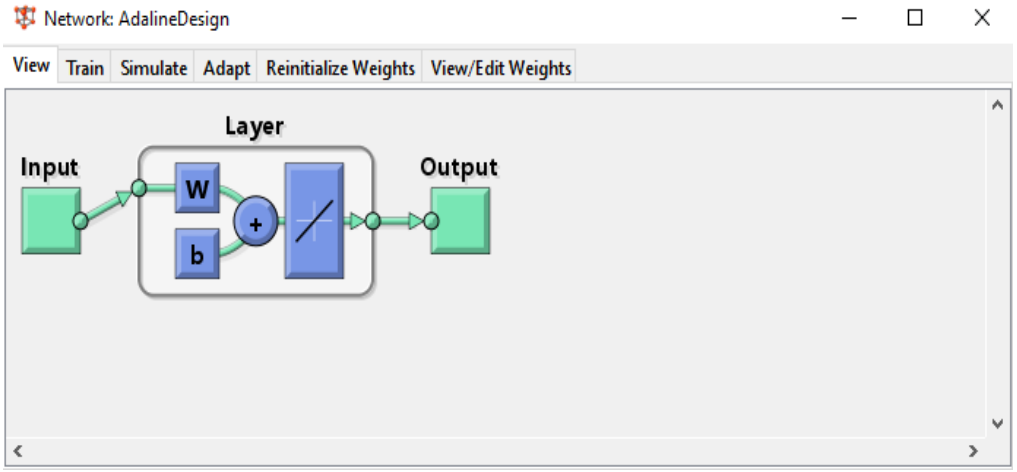
Figura 14: Treinamento para uma rede *Adaline*.



Fonte: O próprio autor.

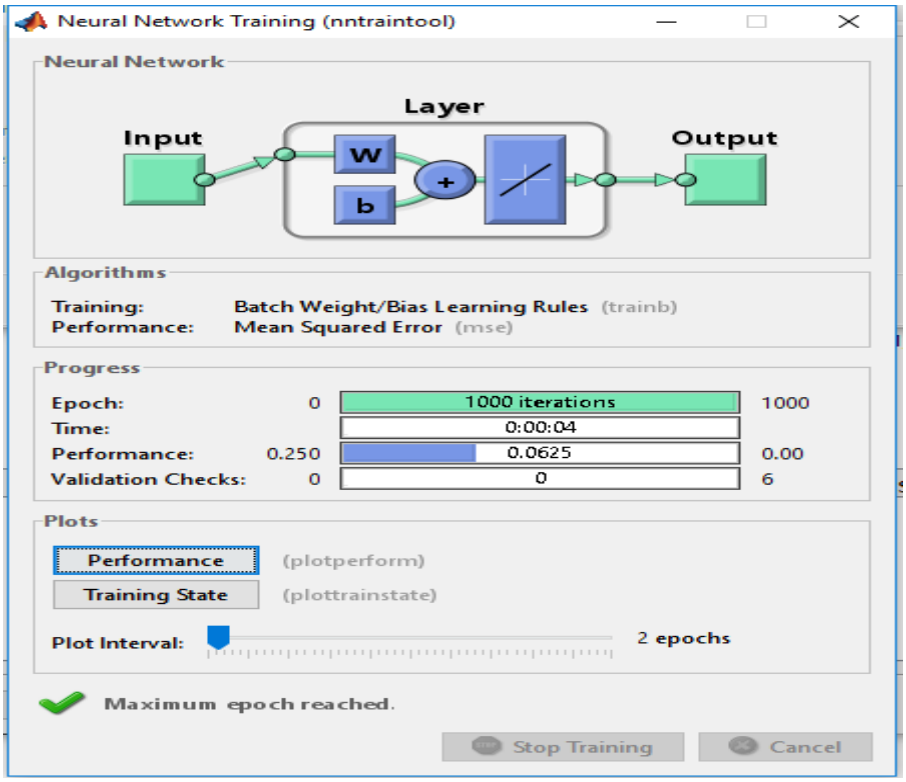
A fim de obter mais dados para análise futura nesse trabalho, foram elaboradas cinco redes *Adaline*: uma para ilustrar o *design* e quatro para seu treinamento. A única diferença inicial entre as redes de treinamento é a sua taxa de aprendizado (*learning rate*), a qual está no intervalo [0;1]. A Figura 15 ilustra a rede *Adaline* como projeto (*design*). As Figuras 16, 17, 18 e 19 ilustram as redes de treinamento levando em conta seus resultados obtidos.

Figura 15:Rede *Adaline* – projeto.



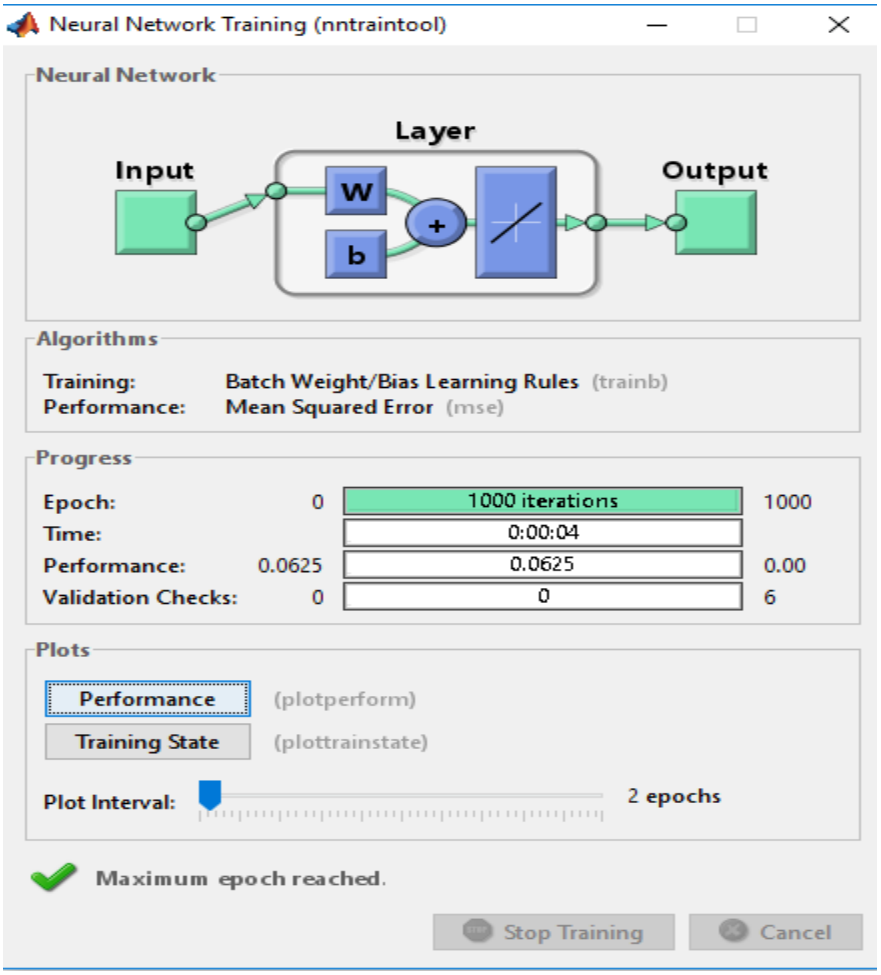
Fonte: O próprio autor, 2016.

Figura 16: Rede *Adaline* – treinamento com taxa de aprendizado 0,01.



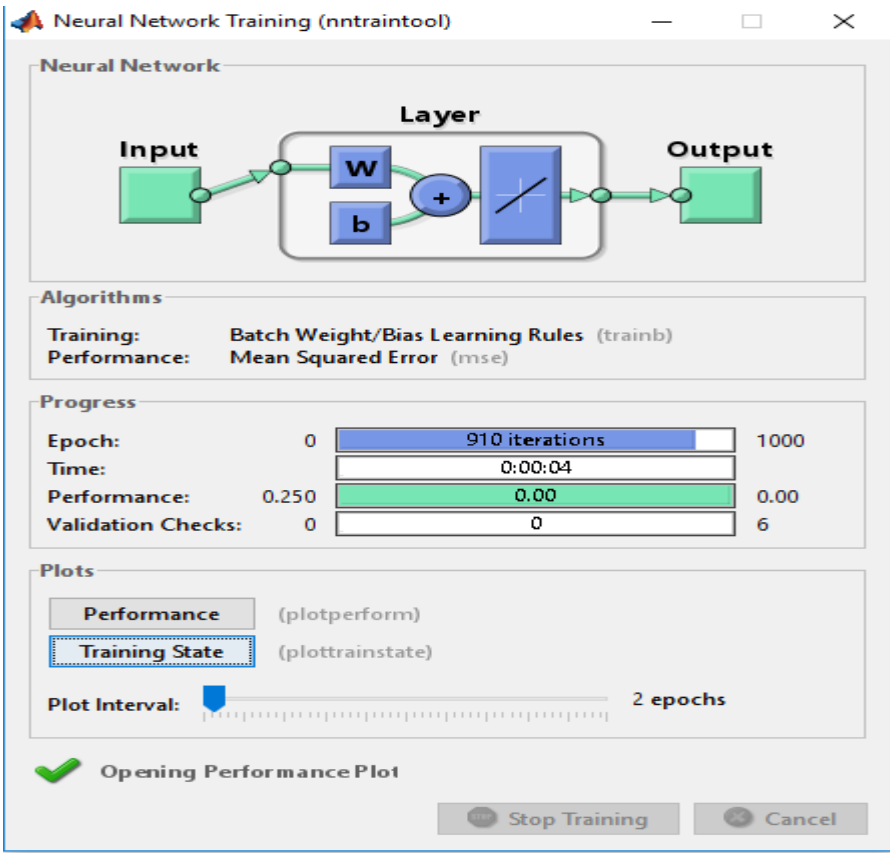
Fonte: O próprio autor, 2016.

Figura 17: Rede *Adaline* – treinamento com taxa de aprendizado 0,1.



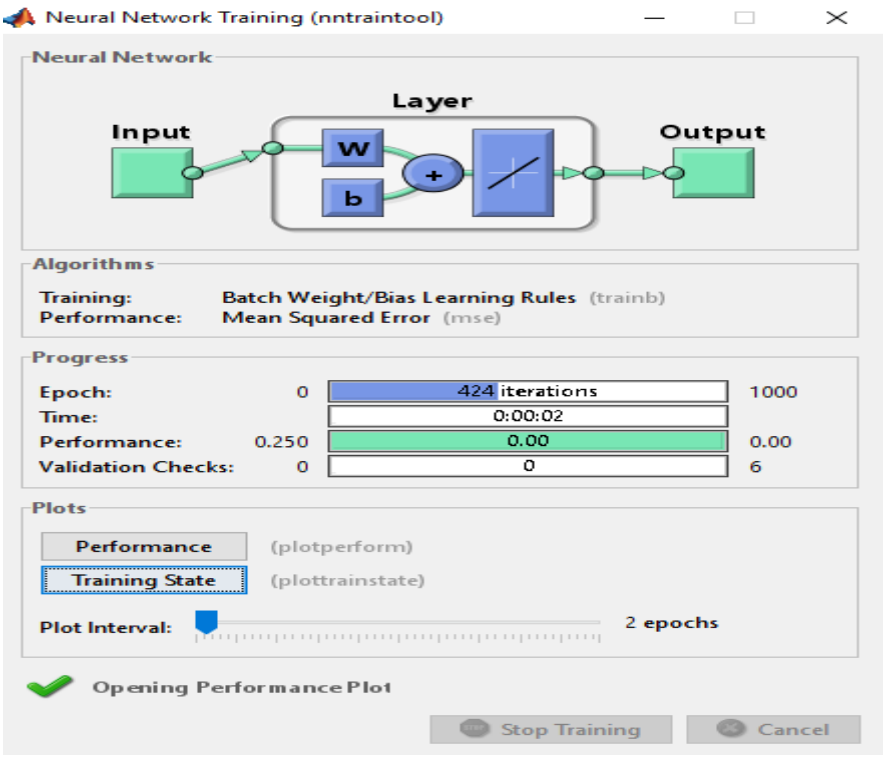
Fonte: O próprio autor, 2016.

Figura 18: Rede *Adaline* – treinamento com taxa de aprendizado 0,5.



Fonte: O próprio autor, 2016.

Figura 19: Rede *Adaline* – treinamento com taxa de aprendizado 1,0.



Fonte: O próprio autor, 2016.

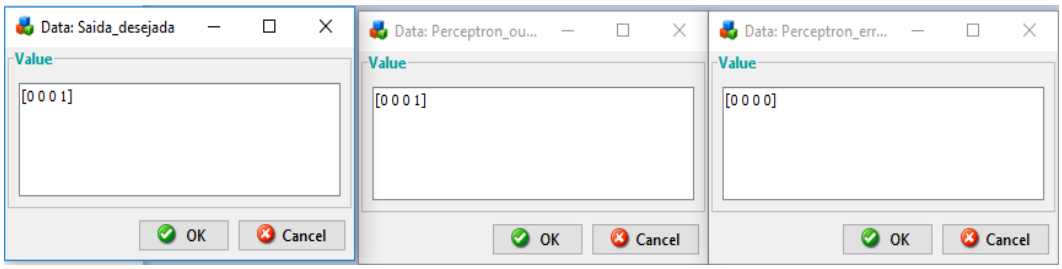
4 ANÁLISE DOS RESULTADOS

Após simular dois modelos para redes neurais, deve-se fazer um estudo qualitativo e quantitativo a respeito de alguns de seus parâmetros como erro, tempo de simulação e desempenho da rede.

1) Erro

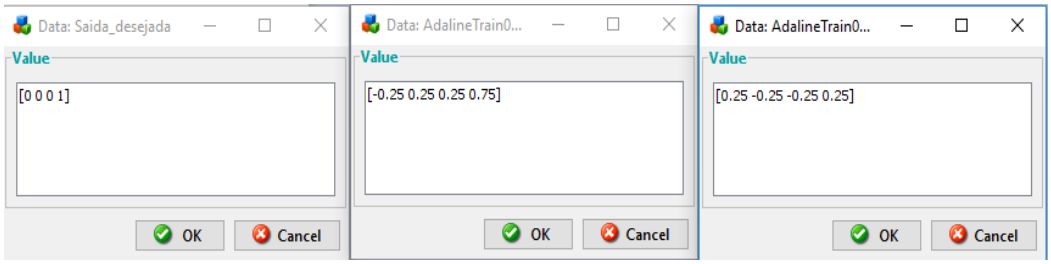
Para as condições propostas na seção anterior, pode-se afirmar que a rede *perceptron* possui erro nulo, dado que sua saída y é exatamente igual a sua saída desejada y_d, como representado na Figura 20. Já para a rede *Adaline* o erro foi distinto de zero em todos os quatro casos que se usou a taxa de aprendizado, ou seja, suas saídas correntes não acompanharam o valor desejado.

Figura 20: Rede *perceptron* – saídas desejada e corrente, bem como seu erro.



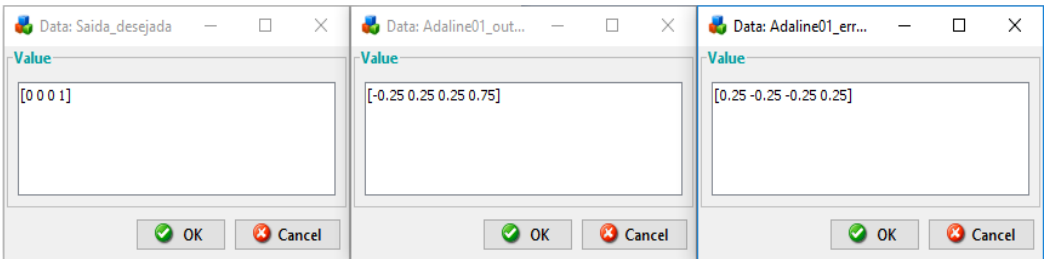
Fonte: O próprio autor, 2016.

Figura 21: Rede *Adaline* (taxa de aprendizado = 0,01) – saídas desejada e corrente, bem como seu erro.



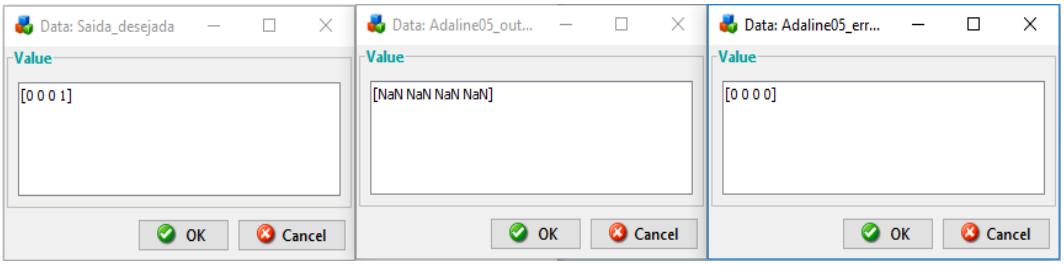
Fonte: O próprio autor, 2016.

Figura 22: Rede *Adaline* (taxa de aprendizado = 0,1) – saídas desejada e corrente, bem como seu erro.



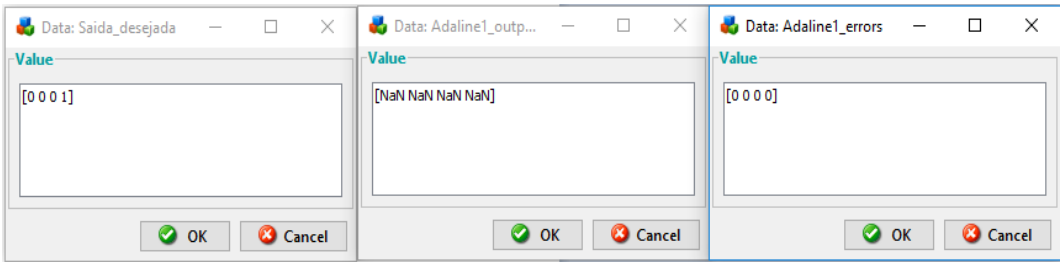
Fonte: O próprio autor, 2016.

Figura 23: Rede *Adaline* (taxa de aprendizado = 0,5) – saídas desejada e corrente, bem como seu erro.



Fonte: O próprio autor, 2016.

Figura 24: Rede *Adaline* (taxa de aprendizado = 1,0) – saídas desejada e corrente, bem como seu erro.



Fonte: O próprio autor, 2016.

Vale ressaltar que alguns erros da rede *Adaline* resultaram em erro nulo, porém isso é uma distorção, uma vez que valores desejados e correntes ficaram totalmente distintos.

2) Tempo de simulação (*epoch*) e desempenho

Todas as simulações realizadas nesse trabalho tiveram uma duração de *epoch*= 1000. Assim, a diferença crucial nesse parâmetro ocorre quando as redes conseguem completar seu treinamento. Pensando nisso, o Quadro1 é apresentado, analisando cada método.

Quadro1: Análise dos métodos com base no número de iterações, tempo e erro.

Método	*Perceptron*	*Adaline* (1,0)	*Adaline* (0,5)	*Adaline* (0,1)	*Adaline* (0,01)
Iterações/tempo (*s*)	5/0	424/2	910/4	1000/4	1000/4
Erro nulo	Sim	Não	Não	Não	Não

Fonte: O próprio autor, 2016.

Com base no Quadro1, nota-se que apenas a rede *perceptron* atinge o erro nulo, confirmando uma de suas características que é a de se obter erro nulo para um número finito de iterações. Quanto à rede *Adaline* não se pode afirmar nada sobre erro nulo, pois para o período de simulação escolhido, tem-se confirmação de erro não nulo onde a taxa de aprendizado foi maior ou igual a 0,5. Entende-se então que valores de taxa de aprendizado no intervalo (0,1; 0,5) são mais interessantes quando se pensa em convergência de iterações.

5 CONCLUSÃO

Com base no estudo realizado, unindo-se teoria das redes neurais artificiais e métodos computacionais com o auxílio do MATLAB, fez-se um estudo introdutório das RNAs e aplicações de dois de seus modelos de aprendizagem.

Os modelos *perceptron* e *Adaline* são de natureza simples e resolvem apenas problemas com características lineares. Contudo é possível resolver com eles problemas de natureza mais complexa através da utilização de portas de limiar de maior complexidade ou através de funções não-lineares.

Toda a pesquisa realizada veio a ser confirmada, mediante as simulações, ou seja, que a principal diferença entre os modelos *perceptron* e *Adaline* é que o *perceptron* se caracteriza como separador linear e o *Adaline* como aproximador linear de funções. Dessa forma, entende-se que o uso do *perceptron* como rede neural simples é válido para resolução de problemas de classificação, como por exemplo, o estado de maturação de frutos. Já a rede *Adaline* é utilizada para aproximação de funções, como estimativa de massa e volume de frutos.

Nota-se que as redes neurais artificiais podem ser aplicadas em vários setores, como o setor agrícola, mas também no setor econômico, setor elétrico, etc. Portanto, um vasto campo de pesquisa está à disposição para aprofundar nesse tão requisitado método computacional.

REFERÊNCIAS

[1] Fabricio Breve. Redes Neurais Artificiais. Disponível em <http://www.fabriciobreve.com/material/cin/CIN-06-RedesNeuraisArtificiais.pdf>, acessado em 13/06/2016.

[2] Fernando Osório e João Ricardo Bittencourt. Sistemas Inteligentes baseados em Redes Neurais Artificiais aplicados ao Processamento de Imagem. Disponível em<http://osorio.wait4.org/oldsite/wia-unisc/wia2000-full.pdf>, acessado em 13/06/2016.

[3] BRAGA, A; CARVALHO, A; LUDERMIR, T. Redes neurais artificiais: teoria e aplicações. 2.ed. Rio de Janeiro: LTC, 2001.

[4] CARVALHO, J. (2015). Seleção e classificação inteligente de mangas por análise de imagem. Fevereiro de 2015. 301 folhas. Tese - Universidade Federal de Campina Grande. Campina Grande, fevereiro de 2015.

[5] Marcelo S. Portugal e Luiz Gustavo L. Fernandes. Redes Neurais Artificiais e Previsão de Séries Econômicas: Uma Introdução.Disponível em <http://www.inf.ufrgs.br/~danielnm/docs/intro_rna.pdf>, acessado em 13/06/2016.

[6] André Cardon e Daniel Nehme Muller. Introdução às Redes Neurais Artificiais. Disponível em <http://www.inf.ufrgs.br/~danielnm/docs/intro_rna.pdf>, acessado em 11/06/2016.

[7] Barbon. RNA – Arquiteturas e Treinamento. Disponível em <http://www.barbon.com.br/wp-content/uploads/2013/08/RNA_Aula2.pdf>, acessado em 13/06/2016.