Uso de Redes Neurais Artificiais na Determinação dos Zeros de Funções Polinomiais

Ircílio Chissolucombe

***Resumo*—A Inteligência Artificial tem sido muito utilizada nos últimos anos em pesquisas e na resolução de problemas quotidianos. Neste trabalho utilizou-se uma das técnicas da Inteligência Artificial denominada de Redes Neurais Artificiais (RNA) para determinar os zeros de funções polinomiais. A ferramenta computacional utilizada foi o SNNS (*Stuttgart Neural Network Simulator*) desenvolvido na Universidade de Stuttgart-Alemanha e as redes foram do tipo *feedforward* com o algoritmo *backpropagation*.**

***Palavras-chave*— Redes Neurais Artificiais, Funções Polinomiais.**

I. Introdução

VÁRIAS são as definições usadas no intuito de definir, de forma precisa, o que é a inteligência artificial (IA), mas a grande maioria delas acaba caindo em discussões filosóficas e psicológicas sobre o que é inteligência e o que é artificial. Rich & Knight (1993) definiram a inteligência artificial como sendo o estudo de como fazer os computadores realizarem coisas que, no momento, as pessoas fazem melhor. Outra definição utilizada no meio acadêmico é a de que um programa de computador seria considerado inteligente se realizasse uma tarefa que, se fosse realizada por um ser humano, seria considerada inteligente. O nome inteligência artificial surgiu após a realização de um seminário realizado em 1956 na cidade de Darthmouth, atualmente muitos pesquisadores concordam que o nome adotado não foi o correto o melhor seria chamar de racionalidade computacional (Russell & Norving, 2003). Os sistemas complexos que executam uma seqüência de passos previamente definidos não devem ser confundidos com os sistemas inteligentes, pois uma das características dos sistemas inteligentes é a capacidade de aprender e de se adaptar a um ambiente desconhecido ou a uma situação nova (Bauchspiess, 2002).

Os princípios teóricos que regem a inteligência artificial são baseados em duas abordagens. A abordagem cognitiva, também chamada de simbolista, que dá ênfase à forma como o ser humano raciocina. Esta abordagem tem como objetivo encontrar uma explicação para comportamentos inteligentes baseada em processos algoritmos como a lógica de predicados. Os sistemas especialistas usam este tipo de abordagem. A outra abordagem é a chamada conexionista, a qual dá ênfase ao modelo de funcionamento do cérebro, dos neurônios e das conexões neurais. A abordagem conexionista considera ser virtualmente impossível transformar em algoritmos e/ou reduzir a uma seqüência de passos lógicos e aritméticos tarefas que a mente humana faz com facilidade e rapidez como, por exemplo, o reconhecimento de rostos. As Redes Neurais Artificiais (RNA) é uma técnica de IA que se enquadra nessa abordagem. Esta técnica permite estimar funções lineares e não lineares treinando-se previamente a rede com pares de exemplos (entradas, saída) oriundos destas funções.

É apresentado neste trabalho um estudo onde se utilizou as redes neurais artificiais na determinação dos zeros de funções polinomiais de 2º grau. Utilizando-se a ferramenta computacional SNNS (Stuttgart Neural Network Simulator) desenvolvido na Universidade de Stuttgart-Alemanha sendo as redes do tipo *feedforward* com o algoritmo *backpropagation*.

II. Redes Neurais Artificiais

Uma rede neural é um processador maciço paralelamente distribuído e constituído de unidades de processamento simples, que tem propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso. Ela se assemelha ao cérebro em dois aspectos: (a) o conhecimento é adquirido pela rede a partir de seu ambiente por meio de um processo de aprendizagem; (b) forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido (Haykin, 2001). As principais características de uma rede neural são: (a) não linearidade; (b) mapeamento de entrada e saída; (c) adaptabilidade; (d) resposta à evidência; (e) informação contextual; (f) tolerância à falha e (g) uniformidade de análise e projeto.

I. Chissolucombe é professor da Escola de Tecnologia da Faculdade Projeção. Contato: ircilio.chissolucomb@projecao.br.

III. HISTÓRICO

Aqui é apresentada uma cronologia da evolução das redes neurais artificiais desde a sua criação até aos dias de hoje.

1943 - O psiquiatra e neuroanatomista Warren McCulloch e o matemático Walter Pitts propuseram um modelo simplificado de neurônio biológico. O neurônio foi definido como tendo dois estados, ativo ou inativo que corresponderia ao zero e um da álgebra booleana. Esse modelo de neurônio artificial não apresentava técnicas de aprendizado.

1949 - Donald Hebb publicou o livro intitulado "A Organização do Comportamento". Neste livro Donald Hebb propôs uma regra de aprendizagem conhecida como regra de aprendizagem Hebbiana. O mesmo demonstrou matematicamente que a aprendizagem de redes neurais pode ser conseguida através da variação dos pesos das conexões entre os neurônios.

1957 - Frank Rosenblatt criou o Mark Perceptron que foi o primeiro modelo de redes neurais a ter sucesso. Rosenblatt demonstrou que se fossem acrescidos pesos ajustáveis, os neurônios artificiais de McCulloch e Pitts poderiam ser utilizados para treinar certos tipos de padrões em duas categorias, dada a sua saída binária. Uma das idéias do mesmo foi à elaboração da arquitetura back perceptron e o algoritmo back-coupled error correction algorithm, que é capaz de adaptar os pesos de um conjunto de entradas de uma unidade de associação, de acordo com a saída desejada. Devido à profundidade do trabalho e a grande contribuição que o mesmo deu para a era moderna, muitos o consideram como o criador da neurocomputação. A arquitetura da rede era composta por camada de entrada e camada de saída e tinha apenas saídas discretas (binárias), sendo usado para resolver problemas de reconhecimento de padrões.

1960 - Bernard Widrow e Hoff desenvolveram os modelos ADALINE (Adaptive Linear Element) e MADALINE (Many Adaptive Linear Element) para reconhecimento de padrões. O modelo ADALINE possuía saídas analógicas em vez de discretas e utilizava um algoritmo de aprendizado baseado no método do gradiente. Este processo minimiza o quadrado dos erros das saídas. Este algoritmo é conhecido como regra delta ou regra de Widrow e Hoff. Tanto o back-coupled error correction e a regra delta eram utilizados para treinar redes cuja arquitetura não tinha camadas intermediárias.

1969 - Minsky e Papert publicaram o livro "Perceptrons". Neste livro Minsky e Papert provaram que redes formadas com apenas uma camada de neurônios independente do algoritmo de aprendizagem só era capaz de resolver problemas de associação de padrões quando os conjuntos de pares de padrões fossem linearmente separáveis. Naquela época apesar de já se ter conhecimento do Perceptron multicamadas (com camadas intermediárias) desconhecia-se um algoritmo de aprendizagem que pudesse resolver problemas linearmente separáveis. As observações de Minsky e Papert foram devastadoras para a linha de pesquisa em redes neurais artificiais, ficando a abordagem conexionista para segundo plano em detrimento da abordagem cognitiva (sistemas especialistas) durante a década de 70 até início da década de 80.

1982 - John Hopfield físico renomado escreveu vários artigos sobre neurocomputação que acabou chamando atenção a matemáticos e tecnólogos. Também houve nessa época um grande avanço no desenvolvimento da microeletrônica possibilitando a implementação de arquiteturas de redes mais sofisticadas.

1986 - David Rumelhart e James McClelland publicaram o livro "Processamento Paralelo Distribuído", neste livro estava descrita uma nova regra de aprendizagem denominada de regra delta generalizada também conhecida como algoritmo back propagation que permitia utilizar o Perceptron multicamadas para resolver problemas com padrões que não fossem linearmente separáveis. As funções de ativação foram substituídas por funções sigmóides. A descoberta do novo algoritmo associada aos avanços tecnológicos provocou uma nova explosão nas pesquisas envolvendo a neurocomputação.

IV. NEURÔNIO BIOLÓGICO

O sistema nervoso é formado, basicamente, por células nervosas que estão interconectadas de forma precisa e especifica, formando os chamados circuitos neurais. Este sistema detecta estímulos externos e internos que podem ser físicos e químicos e desencadeia as respostas musculares e glandulares.

O principal componente do sistema nervoso é uma célula denominada de neurônio (Figura 1). Um cérebro humano possui aproximadamente 15 bilhões destas células que são responsáveis por todas as funções do sistema. Segundo Tafner (1998) existem vários tipos de neurônios sendo os componentes básicos dos mesmos os seguintes: (a) corpo do neurônio (soma): constituído de núcleo e pericário, que dá suporte metabólico a toda célula; (b) axônio (fibra nervosa): prolongamento único e grande que aparece no soma, responsável pela condução do impulso; (c) sinapses: é a estrutura dos neurônios através da qual ocorrem os processos de comunicação entre os mesmos, ou seja, ocorre a passagem do sinal neural; (d) dendritos: são prolongamentos menores em forma de ramificações que emergem do pericário, responsáveis pela comunicação entre os neurônios através das sinapses.

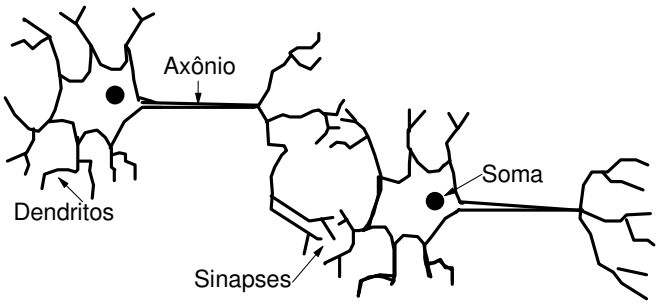


Fig. 1. Modelo de um neurônio biológico (modificado - Ruffier et al., 1998).

O sinal de uma célula a outra se faz mediante uma reação eletroquímica. Substâncias químicas transmissoras são lançadas das sinapses e entram pelos dendritos aumentando ou baixando o potencial elétrico do corpo da célula. Quando o potencial chega a um limiar, um pulso elétrico é mandado pelo axônio, este pulso espalha-se ao longo das conexões existentes pelo axônio chegando a outras sinapses e lançando transmissores ao corpo de outras células (Medeiros, 2003).

V. NEURÔNIO ARTIFICIAL

O neurônio artificial procura simular as funções e o comportamento de um neurônio biológico. No neurônio artificial os dendritos foram substituídos pelas entradas, os pesos simulam as sinapses. Os estímulos recebidos pelas entradas são processados pela função soma (combinação linear), e o limiar de disparo do neurônio foi substituído pela função de transferência (Figura 2).

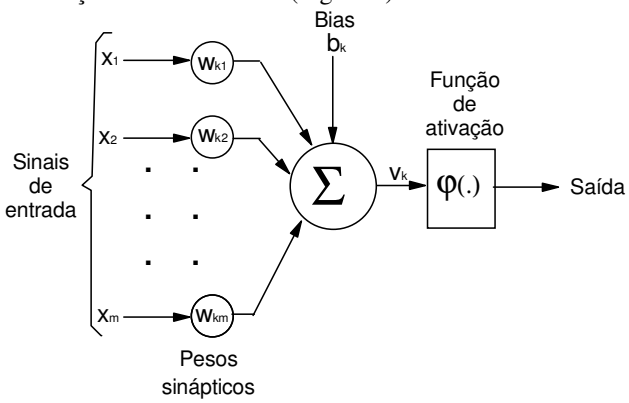


Fig. 2. Modelo de um neurônio artificial (modificado - Haykin, 2001).

O combinador linear u_k é obtido a partir do valor da soma ponderada das entradas pelo valor dos seus respectivos pesos sinápticos, ficando expresso da seguinte forma:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

(1)

onde k representa o neurônio em questão, x_j o valor da entrada j, m o número de entradas, w_{kj} o valor do peso do neurônio j na entrada k.

O neurônio recebe também um sinal externo denominado de bias b_k que tem como função aumentar ou diminuir a entrada líquida da função de ativação. O sinal de saída y_k é obtido da seguinte forma:

$$y_k = \varphi(u_k + b_k) = \varphi(v_k)$$

(2)

onde φ representa a função de ativação e $v_k = u_k + b_k$.

O neurônio irá receber um conjunto de sinais de entrada e então é computada a soma ponderada entre os sinais de entrada e os pesos das conexões. O resultado disto é aplicado a uma função que é denominada de função de ativação ou função de transferência. Existe um valor de u_k que quando atingido provoca um disparo ocasionado pela função de ativação este valor é chamado de limiar. Existem vários tipos de função de ativação (Figura 3) dentre as quais podemos destacar as seguintes:

- *Função Limiar ou de Heaviside*:

$$y_k = \begin{cases} 1 \text{ se } v_k \geq 0 \\ 0 \text{ se } v_k < 0 \end{cases}$$

(3)

- *Função linear*:

$$y_k = \alpha v_k$$

(4)

- *Função rampa ou linear de limites rígidos*:

$$\varphi(v) = \begin{cases} +\gamma \text{ se } v_k \geq +\gamma \\ v_k \text{ se } -\gamma < v_k < +\gamma \\ -\gamma \text{ se } v_k \leq 0 \end{cases}$$

(5)

- *Função degrau*:

$$\varphi(v) = \begin{cases} +1 \text{ se } v_k \geq 0 \\ -1 \text{ se } v_k < 0 \end{cases}$$

(6)

- *Função logística*:

$$\varphi(v) = \frac{1}{1 + e^{v_k}}$$

(7)

• *Função hiperbólica*:

$$\varphi(v) = \frac{1 - e^{-2v_k}}{1 + e^{-2v_k}}$$

(8)

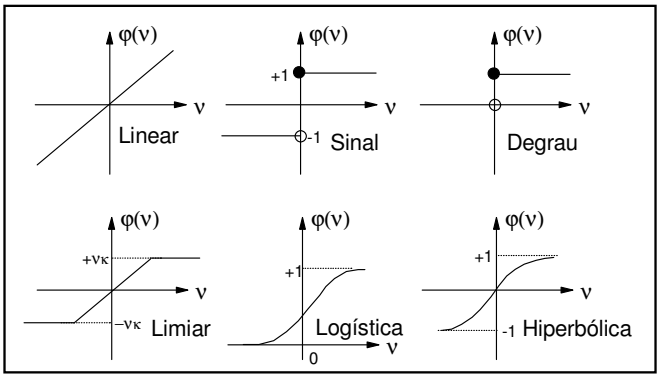


Fig. 3. Gráficos de algumas funções de transferências (modificado - Loesch & Sari, 1996).

VI. Aprendizado em Redes Neurais Artificiais

A propriedade mais importante das redes neurais artificiais é a habilidade de aprender a partir de seu ambiente e com isso melhorar o seu desempenho, este aprendizado ocorre quando a rede neural atinge uma solução generalizada para uma determinada classe de problemas. Mendel e McClaren em 1970 (citados por Haykin, 2001) definiram o aprendizado como sendo um processo interativo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente no qual a rede está inserida. Esta definição implica na seguinte seqüência de passos: (a) a rede neural é estimulada pelo ambiente; (b) a rede neural sofre modificações nos seus parâmetros livres como resultado desta estimulação e (c) a rede neural responde de maneira nova ao ambiente, devido às modificações ocorridas na sua estrutura interna.

O conjunto de regras bem definidas para a solução de um problema de aprendizado é denominado de algoritmo de aprendizado. Existem vários tipos de algoritmos de aprendizado sendo a principal diferença entre eles a maneira como os pesos são ajustados. Os processos de aprendizagem podem ser agrupados em dois paradigmas principais, sendo eles o aprendizado supervisionado e o aprendizado não supervisionado. Existem outros paradigmas que são casos particulares dos paradigmas citados acima, que é o caso do aprendizado por reforço (supervisionado) e o aprendizado por competição (não-supervisionado).

No aprendizado supervisionado tem-se a existência de um agente externo ou supervisor neste caso é apresentado a rede um conjunto de par de exemplos, ou seja, as entradas com as saídas desejadas. Os parâmetros livres são ajustados para que se tenha uma função de custo mínimo (erro). Já no caso do aprendizado não-supervisionado, também conhecido como auto-organização, não existe supervisor, são apresentados à rede somente os padrões de entrada. Neste caso a rede desenvolve a habilidade de formar novas representações internas para codificar características de entrada e criar novas classes ou grupos automaticamente. O paradigma de aprendizado supervisionado utiliza os algoritmos regra delta de Widrow-Hoff e a regra delta generalizada (*backpropagation*). Já para o paradigma de aprendizado não supervisionado tem-se as redes de auto-organização de Kohonen.

VII. Aprendizado Supervisionado (Derivação do Algoritmo *Back Propagation*)

Durante a década de 70 houve uma redução nas pesquisas utilizando redes neurais isto devido à falta de um algoritmo de treinamento que permitisse efetuar o ajuste dos pesos das camadas intermediárias. O algoritmo *back propagation* é uma extensão da regra delta, por isso é também conhecido como regra delta generalizada. O ajuste dos pesos é feito utilizando-se o método do gradiente descendente. Uma função de custo é minimizada e a mesma é definida pela soma dos erros quadráticos. O valor do erro no sinal de saída j na interação n é definido por:

$$e_j(n) = d_j(n) - y_j(n)$$

(9)

onde $d_j(n)$ é a saída desejada para neurônio j, $y_j(n)$ é a saída estimada pela rede para o neurônio j sendo j o neurônio de saída.

Assim a função de custo a ser minimizada que está associada ao erro total é definida por:

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

(10)

onde C representa todos os neurônios da camada de saída da rede.

Considerando que a minimização do erro de cada padrão individual levará a minimização do erro total, assim a função que define o erro quadrado para um padrão individual é definida por:

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

(11)

A regra delta sugere que a variação dos pesos seja definida de acordo com o gradiente descendente do erro em relação ao peso. Esse ajuste a ser feito no peso irá depender de outros parâmetros dinâmicos da rede, tais como, o

combinador linear $v_j(n)$ e a saída estimada pela rede $y_j(n)$ definidas como:

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n)$$

(12)

$$y_j(n) = \varphi(v_j(n))$$

(13)

A correção do peso $\Delta w_{ji}(n)$ a ser aplicado ao peso sináptico $w_{ji}(n)$ é proporcional ao gradiente instantâneo $\partial\varepsilon(n)/\partial w_{ji}(n)$. Este gradiente representa um fator sensitivo, que determina a direção de procura no espaço de pesos para o peso sináptico w_{ji}, sendo expresso da seguinte forma:

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

(14)

onde η é uma constante que determina a taxa de aprendizado, também chamada de taxa de aprendizado do algoritmo *back-propagation.*

O uso do sinal negativo na Equação 14 significa que o gradiente é descendente no espaço de pesos. Aplicando a regra da cadeia na Equação 14 tem-se que:

$$\frac{\partial\varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial\varepsilon(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

(15)

As derivadas parciais da Equação 7 podem ser obtidas derivando as Equações 9, 11, 12 e 13 com relação a $e_j(n)$, $y_j(n)$, $v_j(n)$ e $w_{ji}(n)$:

$$\frac{\partial\varepsilon(n)}{\partial e_j(n)} = e_j(n)$$

(16)

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

(17)

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n))$$

(18)

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

(19)

Assim a correção do peso $\Delta w_{ji}(n)$ a ser aplicado ao peso sináptico $w_{ji}(n)$ definido na Equação 14 pode ser agora definido da seguinte forma:

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

(20)

onde o gradiente local representado por $\delta_j(n)$ é definido por:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\varphi_j'(v_j(n))$$

(21)

Quando j é um neurônio da camada de saída o resultado desejado é conhecido, enquanto quando o mesmo refere-se a um neurônio de uma camada intermediária não se tem o resultado desejado, por isso não se pode utilizar a derivada parcial descrita na Equação 9 para calcular o gradiente local conforme mostrado na Equação 21, sendo assim a mesma é reescrita ficando da seguinte forma:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial y_j(n)} \varphi_j'(v_j(n))$$

(22)

Para calcular a derivada parcial $\partial E(n)/\partial y_j(n)$ reescreve-se a Equação 11 substituindo-se o índice j pelo índice k, sendo que j representa agora um nó intermediário e k um nó de saída:

$$E(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n)$$

(23)

Neste caso a derivação da Equação 23 com respeito a função sinal $y_j(n)$ é dada por:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)}$$

(24)

Posteriormente, com o uso da regra da cadeia reescreve-se a derivada parcial $\partial e(n)/\partial y_j(n)$ ficando assim sua forma equivalente:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)} \cdot \frac{\partial v_k(n)}{\partial y_j(n)}$$

(25)

Sendo o erro $e_k(n)$ e o combinador linear $v_k(n)$ definidos por:

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n))$$

(26)

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

(27)

Obtendo as derivadas parciais das Equações 26 e 27 com relação às variáveis $v_k(n)$ e $y_j(n)$ tem-se:

$$\frac{\partial e_k(n)}{\partial v_j(n)} = -\varphi_k'(v_k(n))$$

(28)

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

(29)

Substituindo-se as derivadas parciais obtidas nas Equações 28 e 29 na Equação 25 tem-se:

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n)\varphi_k'(v_k(n))w_{kj}(n) = -\sum_k \delta_k(n) w_{kj}(n)$$

(30)

Finalmente, obtêm-se o gradiente local $\delta_j(n)$ para o caso onde j representa um nó intermediário, substituindo o termo obtido na Equação 30 na Equação 22:

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n) w_{ki}(n)$$

(31)

VIII. Modelagem com Redes Neurais Artificiais

O conjunto de dados utilizado na modelagem foi obtido gerando-se aleatoriamente valores de *a*, *b* e *c* entre 0 e 1. Estes valores representam os coeficientes de uma função polinomial de 2º grau. Sendo *a* o coeficiente em x^2, *b* o coeficiente em x e *c* o termo independente. Os valores do termo independente *c* foram multiplicados por -1 para que não ocorresse nenhuma situação onde a raiz quadrada fosse negativa. A partir dos coeficientes *a*, *b* e *c* se calculou os valores das raízes x_1 e x_2 utilizando-se as seguintes expressões:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

(32)

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

(33)

Gerou-se um total de 400 pares de exemplos (entrada/saída), onde os coeficientes *a*, *b* e *c* eram as variáveis de entrada e os valores das raízes x_1 e x_2 as variáveis de saída (Figura 4). O total de 400 pares de exemplo foi dividido em dois conjuntos, um para treinamento com 320 pares de exemplos e outro para teste com 80 pares de exemplos. Uma análise estatística foi feita para verificar a representatividade dentro da população (total) dos conjuntos de dados de treinamento e de testes (Tabela 3.1).

Várias arquiteturas de redes foram testadas sendo a arquitetura 3-12-1 (3 neurônios na camada de entrada, 12 neurônios na camada intermediária e 1 neurônio na camada de saída) a que apresentou o melhor resultado. O motivo de se ter apenas um neurônio na camada de saída deve-se ao fato que os valores x_1 e x_2 foram analisados separadamente.

Nas Figuras 5, 6, 7 e 8 são apresentados os resultados para a variável de saída, x_1 obtidos nas fases de treinamento e de teste. O coeficiente de correlação obtido foi de 0,9999 na fase de treinamento e 0,9993 na fase de teste.

Tab. 1. Estudo estatístico do conjunto de dados de entrada e de saída.

Variáveis		Parâmetros Estatísticos			
		Média	Desvio padrão	Mínimo	Máximo
Coef. (a)	Treinamento	0.51	0.28	0.01	0.99
	Teste	0.48	0.30	0.02	0.99
Coef. (b)	Treinamento	0.50	0.29	0.01	1.00
	Teste	0.52	0.31	0.00	0.99
Coef. (c)	Treinamento	-0.51	0.29	-1.00	-0.01
	Teste	-0.48	0.29	-0.99	0.00
Zero (x_1)	Treinamento	0.68	0.48	0.01	3.92
	Teste	0.64	0.43	0.01	2.31
Zero (x_2)	Treinamento	-3.79	11.35	-118.45	-0.19
	Teste	-3.76	7.13	-49.40	-0.32

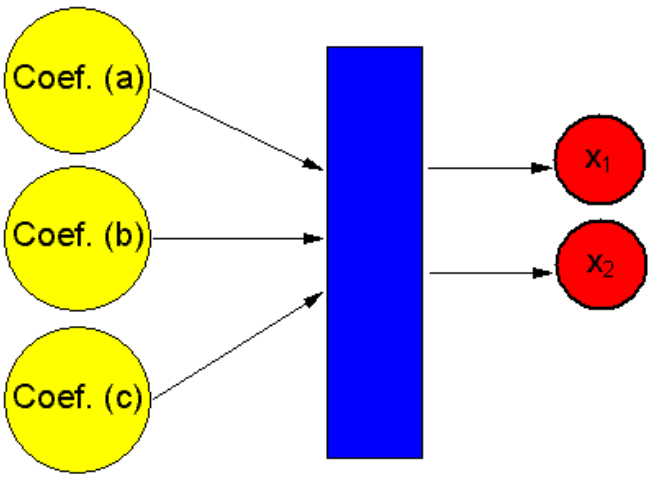


Fig. 4. Variáveis de entrada e de saída da rede neural.

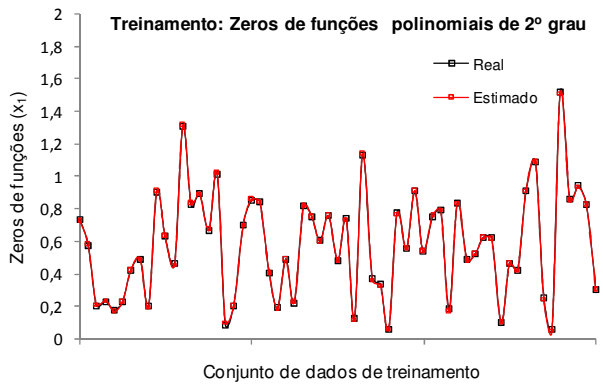


Fig. 5. Variáveis de entrada e de saída da rede neural.

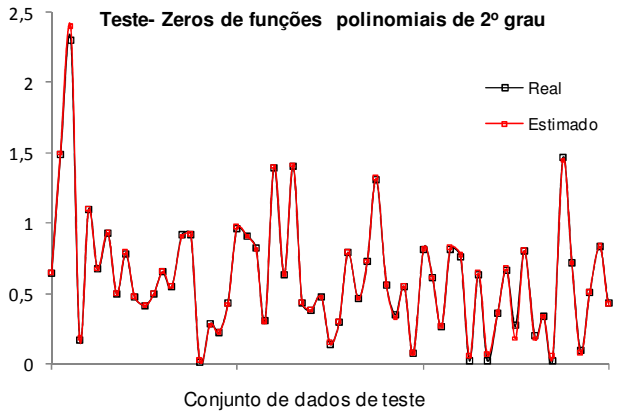


Fig. 6. Resultado da fase de teste para a variável de saída x1.

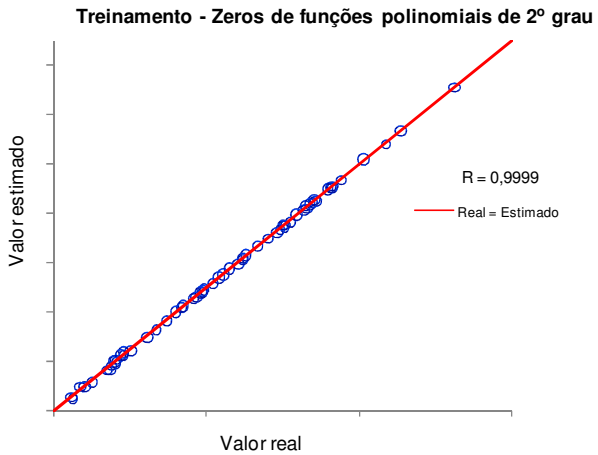


Fig. 7. Valores reais vs. valores estimados para a fase de treinamento da variável de saída x1.

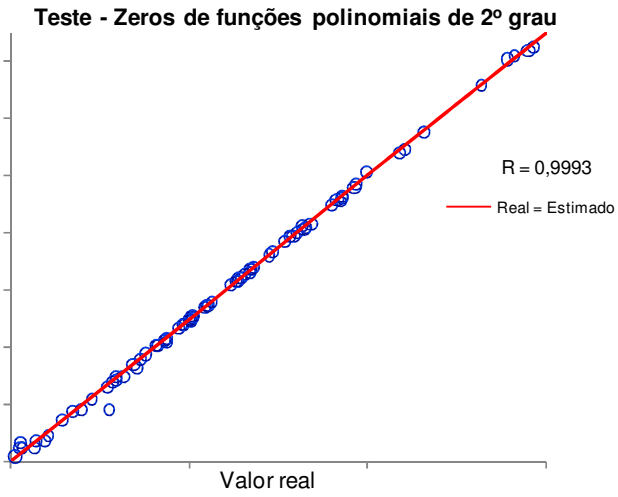


Fig. 8. Valores reais vs. valores estimados para a fase de teste da variável de saída x1.

Nas Figuras 9, 10, 11 e 12 são apresentados os resultados para a variável de saída ,obtidos nas fases de treinamento e de teste. O coeficiente de correlação obtido foi de 1,0 na fase de treinamento e 0,9972 na fase de teste.

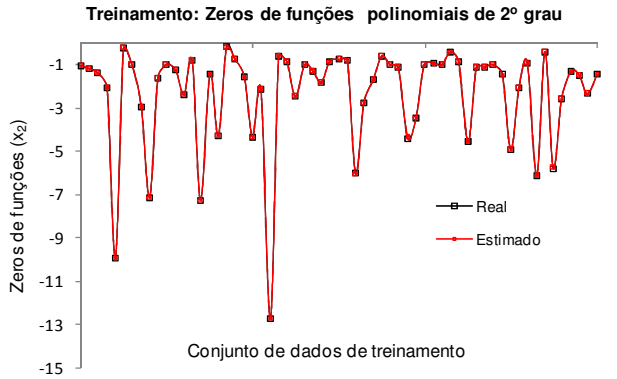


Fig. 9. Resultado da fase de treinamento para a variável de saída x_2.

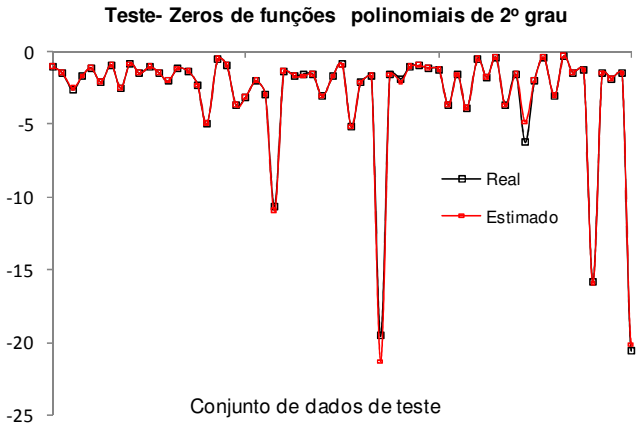


Fig. 10. Resultado da fase de teste para a variável de saída x_2.

Para verificar a importância relativa de cada variável de

entrada efetuou-se uma análise sensitiva de acordo com o proposto por Garson (1991). Os resultados da análise sensitiva são apresentados nas Figuras 13.a e 13.b.

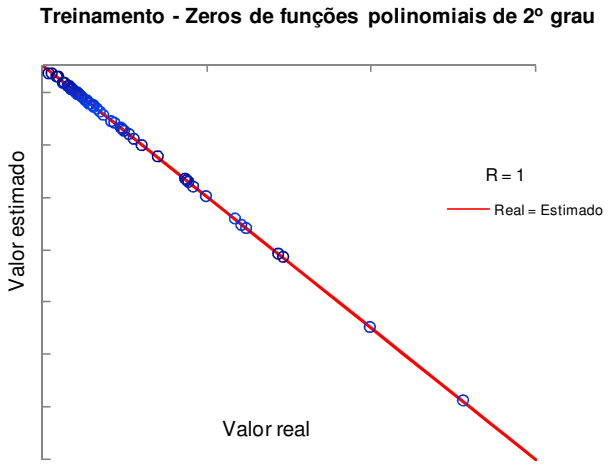


Fig. 11. Valores reais vs. valores estimados para a fase de treinamento da variável de saída x_2.

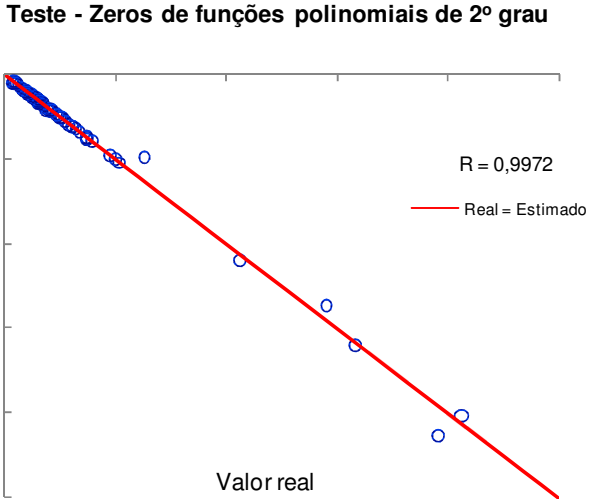
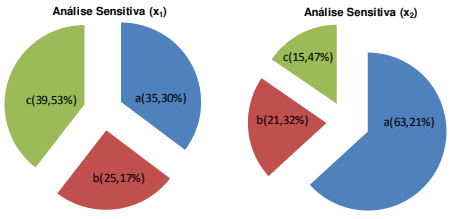


Fig. 12. Valores reais vs. valores estimados para a fase de teste da variável de saída x_2.



(a) (b)

Fig. 13. Importância relativa (%) das variáveis de entrada.

IX. Conclusão

Os resultados obtidos após o treinamento com as RNA permitiu concluir que as RNA são uma ferramenta eficaz na estimativa de funções não lineares. Os valores estimados na fase de teste onde foram apresentados a rede exemplos não apresentados na fase de treinamento foram excelentes com coeficientes de correlação muito próximos de um e erros muito próximos de zero. A análise sensitiva mostrou que dependendo da raiz que se deseja calcular tem-se uma variação considerável na importância relativa das variáveis de entrada.

Este trabalho não tem como propósito a substituição dos métodos tradicionais utilizados para determinar os zeros de funções polinomiais de 2º grau. O mesmo teve como objetivo mostrar a eficácia da ferramenta RNA na aproximação de funções não lineares.

Referências

BAUCHSPIESS, A. (2002). Sistemas Inteligentes aplicados à solução de problemas de engenharia., Semana da Engenharia Civil, Mini-Curso, Faculdade de Tecnologia, Departamento de Engenharia Civil, Universidade de Brasília, DF. 50p.

GARSON, G. D. (1991). Interpreting neural-network connection weigts. AI Expert, 6(7), 47-51.

HAYKIN, S. (2001). Redes Neurais: Princípios e Prática. Bookman, 2o edição, Porto, Alegre, RS, 900p.

LOESCH, C. & SARI, S.T. (1996). Redes Neurais Artificiais: Fundamentos e modelos. Editora da FURB, Blumenau, SC, 166p.

MEDEIROS, L.F. (2003). Redes Neurais em Delphi. Visual Books Editora. Florianópolis, SC. 116p.

RICH, E. & KNIGHT, K. (1993). Inteligência Artificial. Makron Books do Brasil Editora Ltda, 2o edição. 622p.

RUFFIER, A. P., LISBOA, E. F. & DANZIGER, A. B. (1998). Aplicação de redes neurais ao estudo da capacidade de carga de fundações à tração. COBRAM.

RUSSELL S.& NORVING, P. (2003). Inteligência Artificial. Editora Campus, Rio de Janeiro, RJ, Brasil. 1021p.

TAFNER, M.A., (1998). Redes Neurais Artificiais: Aprendizado e Plasticidade. Revista Cérebro & Mente. Universidade Estadual de Campinas. Núcleo de Informática Biomédica.

Ircílio Chissolucombe Professor da Escola de Tecnologia da Faculdade Projeção.