ESTUDO COMPARATIVO DE REDES NEURAIS ARTIFICIAIS PARA PREVISÃO DE SÉRIES TEMPORAIS FINANCEIRAS

David Gabriel de Barros Franco
Pontifícia Universidade Católica do Paraná
david.barros@pucpr.br

Maria Teresinha Arns Steiner
Pontifícia Universidade Católica do Paraná
maria.steiner@pucpr.br

Resumo

Este trabalho visa comparar Redes Neurais Artificiais (RNAs), especificamente o Perceptron de Múltiplas Camadas, a Rede Alimentada Adiante Focada Atrasada no Tempo, a Rede Neural de Base Radial e a Rede de Camada Recorrente, para previsão de séries temporais financeiras, notadamente o valor futuro de ações do mercado de capitais. Fez-se uso do *software* MATLAB para as simulações. Foi realizado um experimento visando testar as redes em várias situações, principalmente no que diz respeito ao número de neurônios na camada oculta e às variações dos pesos iniciais. Para as simulações foram escolhidas as 10 ações de maior peso na composição do Índice Bovespa: Vale (VALE5), Petrobras (PETR4), Itaú-Unibanco (ITUB4), Bradesco (BBDC4), Vale (VALE3), AMBEV (ABEV3), Petrobras (PETR3), Itausa (ITSA4), Banco do Brasil (BBAS3) e BM&FBOVESPA (BVMF3). Os dados de entrada da rede foram as cotações históricas das ações escolhidas. A análise dos resultados foi realizada através da comparação entre os valores previstos pelas RNAs com os valores reais do histórico, medida pelo erro quadrático médio. Tais resultados se apresentaram bastante satisfatórios e os erros mínimos, da ordem de 10^{-11}, se deram para a rede neural de Camada Recorrente.

Palavras-chave: Redes Neurais Artificiais. Previsão de Séries Temporais. Mercado de Ações.

Abstract

This study aim to compare Artificial Neural Networks (ANNs), namely the Multilayer Perceptron, the Focused Time Lagged Feedforward Network, the Radial Basis Function network and Layer-Recurrent Neural network for forecasting financial time series, notably the future value of shares of the capital market. It was made use of MATLAB software for simulations. An experiment was performed aiming to evaluate the networks in various situations, especially with regard to the number of neurons in the hidden layer and the variation of the initial weights. For the simulations were chosen ten shares of the largest weight in the Bovespa index: Vale (VALE5), Petrobras (PETR4), Itaú-Unibanco (ITUB4), Bradesco (BBDC4), Vale (VALE3), AMBEV (ABEV3), Petrobras (PETR3), Itausa (ITSA4), Banco do Brasil (BBAS3) e BM&FBOVESPA (BVMF3). The input of the network were the historical prices of the chosen actions. The analysis was performed by comparing the values predicted by the ANNs with the real historical values, measured by the mean square error. These results are presented very satisfactory and the minimum errors, of the order of 10^{-11}, was given by the Layer-Recurrent neural network.

Key-words: Artificial Neural Networks. Time Series Forecasting. Stock Market.

1. INTRODUÇÃO

Atualmente o mercado de ações movimenta um gigantesco volume financeiro. De acordo com dados do *World Federation of Exchanges* [16], o valor capitalizado do mercado de ações (excluindo fundos de investimento coletivo, opções e futuros) em janeiro de 2014 era de aproximadamente US$ 55 trilhões. Esse valor representa o elevado potencial de financiamento para as empresas que possuem ações listadas em tais mercados.

Porém, existem riscos relacionados ao mercado de ações. Segundo Assaf Neto [1], os riscos associados ao investimento em ações são principalmente: "*risco da empresa* captadora dos recursos e *risco do mercado*". Ainda segundo o autor, "o *risco da empresa* é aquele associado às decisões financeiras, em que são avaliados os aspectos de atratividade econômica do negócio". Já o *risco de mercado* "diz respeito às variações imprevistas no comportamento do mercado, determinadas, principalmente, por mudanças ocorridas na economia".

Este trabalho busca um método para minimizar o risco associado ao mercado financeiro para, por meio da análise das cotações passadas de uma ação, tentar prever seu comportamento futuro, diminuindo as incertezas para o investidor.

A partir do Erro Quadrático Médio (*Mean Square Error*) se poderá comparar a eficácia de cada uma das redes neurais para a previsão de séries temporais financeiras e sugerir qual delas é mais eficaz.

Para o presente trabalho, foram utilizadas as 10 ações de maior participação na composição do Índice Bovespa (Ibovespa). Tal participação é calculada pelo índice de negociabilidade, conforme a equação (1), que busca a representatividade desse título em termos de número de negócios e volume financeiro, ajustado ao tamanho da amostra [2].

$$IN = \sqrt[3]{\frac{ni}{N}} * \sqrt[3]{\left(\frac{vi}{V}\right)^2} * \frac{pi}{P} \qquad (1)$$

onde n_i é o número de negócios com a ação i no mercado à vista, N é o número total de negócios no mercado à vista, v_i é o volume financeiro gerado pelos negócios com a ação i no mercado à vista, V é o volume financeiro total do mercado à vista (todos considerando o lote-padrão de negociação), pi é o número de pregões em que o ativo foi negociado e P é o número de pregões total do período analisado.

As ações escolhidas, e suas respectivas participações no Ibovespa, foram:

VALE5 (8,464%) – *Vale S.A.*, setor de mineração;
PETR4 (7,603%) – *Petróleo Brasileiro S.A.*, setor de petróleo, gás e energia;
ITUB4 (7,064%) – *Itaú Unibanco Holding S.A.*, setor bancário.
BBDC4 (5,537%) – *Banco Bradesco S.A.*, setor bancário.
VALE3 (4,209%) – *Vale S.A.*, setor de mineração;
ABEV3 (4,065%) – *AMBEV S.A.*, setor de bebidas.
PETR3 (3,759%) – *Petróleo Brasileiro S.A.*, setor de petróleo, gás e energia;
ITSA4 (2,896%) – *Itausa Investimentos Itaú S.A.*, gestão de participações societárias.
BBAS3 (2,473%) – *Banco do Brasil S.A.*, setor bancário.
BVMF3 (2,405%) – *BM&FBOVESPA S.A.*, setor financeiro.

Este trabalho está organizado da seguinte forma: na seção 2 são apresentadas as redes neurais aqui utilizadas; na seção 3 são discutidos os resultados; e na seção 5 as conclusões.

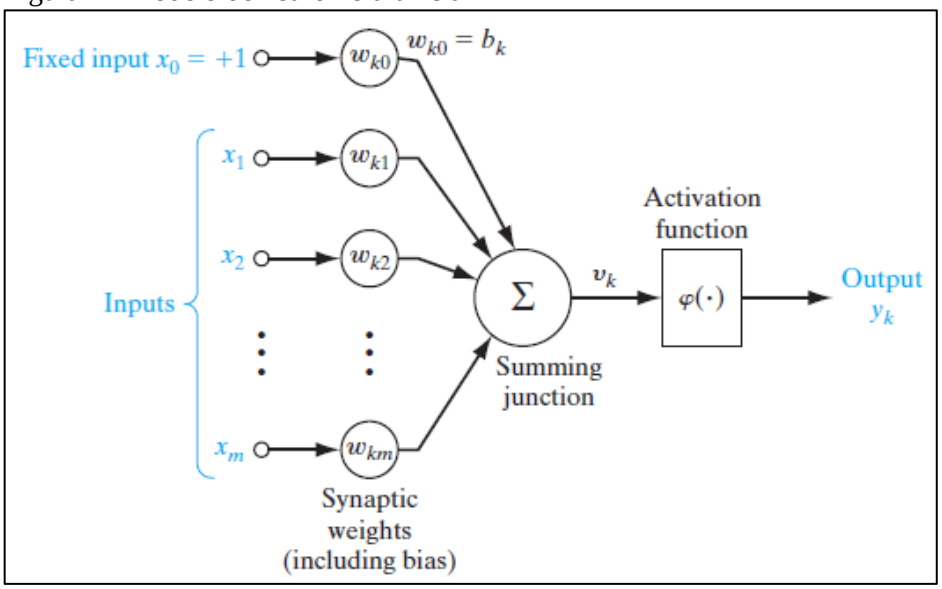
2. REDES NEURAIS ARTIFICIAIS

Uma RNA consiste em um modelo computacional que simula o comportamento do neurônio biológico, aprendendo a partir da alteração de seu estado interno a partir de *inputs* provenientes do ambiente externo. Nas palavras de Haykin [6]:

> uma rede neural é um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso.

O neurônio biológico é uma célula especializada em enviar e receber sinais e se divide basicamente em três seções: corpo celular, dendritos e axônio. O corpo celular guarda o núcleo, os dendritos recebem sinais provenientes de outros neurônios e o axônio envia sinais para outros neurônios [8].
O primeiro modelo de um neurônio artificial foi proposto por Warren McCulloch e Walter Pitts, em 1943 no trabalho intitulado *A Logical Calculus of the Ideas Immanent in Nervous Activity*. E o primeiro estudo sobre o aprendizado de redes neurais foi proposto por Donald Hebb, em 1949, que ficou conhecido como a regra de Hebb. Posteriormente foi proposta por Widrow e Hoff uma nova regra de aprendizado, a regra delta, que ainda é utilizada [4].
O *Perceptron*, em sua forma simples, apareceria em 1958, proposto por Frank Rosenblatt. Em 1986 seria aplicado o algoritmo *backpropagation* (algoritmo por retropropagação do erro) no contexto das redes neurais por Rumelhart e colaboradores [4].
Basicamente um neurônio artificial pode ser representado como na Figura 1.

Figura 1 - Modelo de neurônio artificial



Fonte: HAYKIN, 2009

Segundo Haykin [6] podem-se identificar três elementos básicos na estrutura de um neurônio artificial:

a) as sinapses que recebem os sinais de entrada;
b) o somatório para somar os sinais de entrada com seus respectivos pesos sinápticos;
c) a função de ativação, que restringe a amplitude da saída do neurônio.

Haykin (2000) descreve matematicamente o neurônio artificial k da Figura 1 por (2) e (3):

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \tag{2}$$

$$y_k = \varphi(u_k) \tag{3}$$

onde x_1, x_2, ... , x_m são os sinais de entrada; w_{k1}, w_{k2}, ... , w_{km} são os pesos sinápticos; v_k é o campo local induzido; b_k é o *bias*; $\varphi(.)$ é a função de ativação; e y_k é o sinal de saída, todos relacionados ao neurônio k.

A função de ativação $\varphi(.)$, que restringe a saída do neurônio pode, basicamente, ser do tipo limiar, linear ou sigmoide [6].

Neurônios biológicos ou artificiais, isoladamente, possuem baixa capacidade computacional, mas conectados em uma rede são capazes de resolver problemas de grande complexidade. Basicamente, podem apresentar uma única camada ou possuir camadas ocultas, e serem alimentadas para frente (*feedforward*) ou serem recorrentes de uma camada ou com camada oculta, quando o sinal de saída volta para a entrada da rede [4][7].

Em uma RNA, o aprendizado ocorre à medida que os pesos sinápticos são ajustados com base em alguma regra pré-estabelecida, como a regra delta (ou regra de Widrow-Hoff). Nas palavras de Braga, Carvalho e Ludermir [4]:

> aprendizado é o processo pelo qual os parâmetros livres de uma rede são ajustados por meio de uma forma continuada de estímulo pelo ambiente externo, sendo o tipo específico de aprendizado definido pela maneira particular como ocorrem os ajustes dos parâmetros livres.

2.1. O PERCEPTRON DE CAMADA ÚNICA

Segundo Haykin [6], "o *Perceptron* de camada única é a forma mais simples de uma rede neural usada para a classificação de padrões linearmente separáveis". Outro ponto a respeito do *Perceptron* de camada única é destacado por Braga, Carvalho e Ludermir [4]: "sabe-se que, independentemente do valor η (taxa de aprendizado), haverá convergência em um tempo finito, caso as classes sejam linearmente separáveis".

De um modo resumido, temos o algoritmo do *Perceptron* de camada única, com k neurônios [4][6][13]:

a) faça o vetor de pesos sinápticos $w_k(n) = 0$ e o *bias* $b_k(n) = 0$.
b) execute o somatório $v_k(n) = \sum_{j=1}^{m} w_{kj}(n) x_j(n) + b_k$, onde m é a quantidade de pesos sinápticos para o neurônio k;
c) execute a função de transferência $y_k = \varphi(v_k(n))$, onde $\varphi(.)$ é uma função de limiar, linear ou sigmoide, entre outras;
d) calcule o sinal de erro $e_k(n) = d_j(n) - y_k(n)$, onde d_j é o valor esperado.
e) calcule o novo peso sináptico $w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$, onde $\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$, com $0 < \eta < 1$, $x_j(n)$ e $0 < j < m$ ($w_{k0} = b_k$ e $x_0 = 1$) correspondendo aos sinais de entrada.

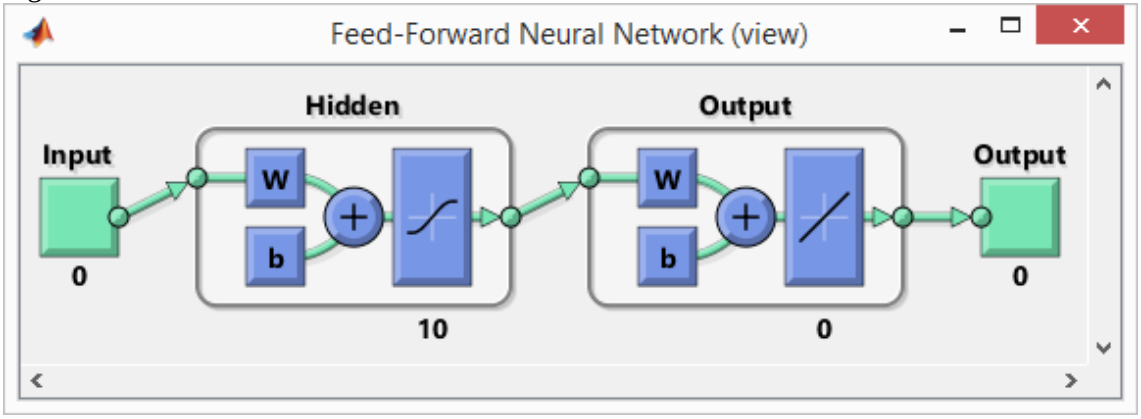
f) após todos os exemplos de treinamento terem passado pela rede uma única vez teremos findado a 1ª iteração, então calcule o erro global $\varepsilon(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n)$, para o conjunto C dos neurônios de saída da rede.
g) incremente a iteração n em uma unidade e volte ao passo 2.

O procedimento deverá continuar até que um erro ε suficientemente pequeno seja encontrado.

2.2. O PERCEPTRON DE MÚLTIPLAS CAMADAS

Um *Perceptron* de múltiplas camadas (*MLP*), nada mais é que uma generalização do *Perceptron* de camada única [6]. Podemos descrevê-lo como uma rede constituída pela camada de entrada, uma ou mais camadas ocultas e uma camada de saída; rede essa que apresenta um alto grau de conectividade [4][6]. Seu treinamento se dá em três etapas: a alimentação para frente (*feedforward*) da rede com os padrões de entrada, o cálculo e retropropagação (*backpropagation*) do erro, e o ajuste dos pesos [5].
Para uma rede neural *MLP* como a representada na Figura 2, teríamos o algoritmo de treinamento como o mostrado na sequência [6][13].

Figura 2 – Rede Neural *MLP*



Fonte: o autor, 2014

Propagação *forward*:
a) inicialize os pesos sinápticos w_{kj} e *bias* b_k (da camada oculta), e pesos sinápticos w_{lj} e *bias* b_l (da camada de saída) com valores aleatórios;
b) execute o somatório $v_k(n) = \sum_{j=1}^{m} w_{kj}x_j(n) + b_k(n)$ para cada neurônio k da camada oculta, onde x_j são os dados de entrada da rede;
c) calcule a função de transferência $y_k(n) = \varphi_k(v_k(n))$ para cada neurônio da camada oculta, onde $\varphi_k(.)$ é, geralmente, uma função sigmoide;
d) na camada de saída, execute o somatório $v_l(n) = \sum_{j=1}^{m} w_{lj}v_k(n) + b_l(n)$ para cada neurônio l da camada de saída, caso haja mais de um;
e) então calcule a função de transferência $y_l(n) = \varphi_l(v_l(n))$ para cada neurônio l da camada de saída, onde $\varphi_l(.)$ é, geralmente, uma função linear;

Propagação *backward*:
a) calcule o erro da camada de saída usando a fórmula $\delta_l(n) = (d_j(n) - y_l(n))\varphi_l'(v_l(n))$, onde d_j é o valor esperado e $\varphi_l'(.)$ é a derivada da função de transferência do neurônio de saída;

b) ajuste os pesos sinápticos da rede na camada de saída de acordo com a regra delta generalizada:

$$w_{lj}(n+1) = w_{lj}(n) + \alpha\left(w_{lj}(n-1)\right) + \eta\delta_l(n)y_l(n)$$

onde η é a taxa de aprendizagem e α é a constante de *momentum*.

c) calcule o erro da camada oculta fazendo $\delta_k(n) = \varphi'_k(v_k(n))\sum_l \delta_l(n)w_{lj}(n)$, onde $\varphi'_k(.)$ é a derivada da função de transferência do neurônio oculto;

d) ajuste os pesos sinápticos da rede na camada oculta de acordo com a regra delta generalizada:

$$w_{kj}(n+1) = w_{kj}(n) + \alpha\left(w_{kj}(n-1)\right) + \eta\delta_k(n)y_k(n)$$

Após todos os exemplos de treinamento terem passado pela rede uma única vez teremos findado a 1ª iteração, então calcule o erro global $\varepsilon(n) = \frac{1}{2}\sum_{l\in C} e_l^2(n)$, para o conjunto C dos neurônios de saída da rede, onde $e_l(n) = d_j(n) - y_l(n)$.

As iterações devem prosseguir até que um valor ε suficientemente pequeno seja encontrado. Como nos explica Steiner [13]: "o processo de aprendizagem da rede neural pode ser visto como um problema de minimização com função objetivo ε no espaço de w".

2.3. REDE NEURAL FUNÇÃO DE BASE RADIAL

A função de ativação aplicada a cada neurônio da maioria das redes multicamadas utiliza como argumento o produto escalar do vetor de entrada e do vetor de pesos desse neurônio. Existem, porém, redes multicamadas em que a ativação de um neurônio pode ser função da distância entre seus vetores de entrada e de peso. Uma dessas redes é a rede neural de Função de Base Radial, *RBF* (*Radial Basis Function*). Este nome se deve à utilização, pelos neurônios da camada intermediária, de funções de base radial. As funções radiais mais comumente usadas são mostradas na Tabela 1, onde x é o vetor de entrada, μ o centro da função e σ a largura da função.

Tabela 1 - Funções de base radial mais comuns

Função gaussiana	$f(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \sigma > 0$
Função multiquadrática	$f(x) = \sqrt{(x-\mu)^2 + \sigma^2}, \sigma > 0$
Função *thin-plate-spline*	$f(x) = (x-\mu)^2 \ln(x-\mu)$

Fonte: BRAGA; CARVALHO e LUDERMIR, 2011

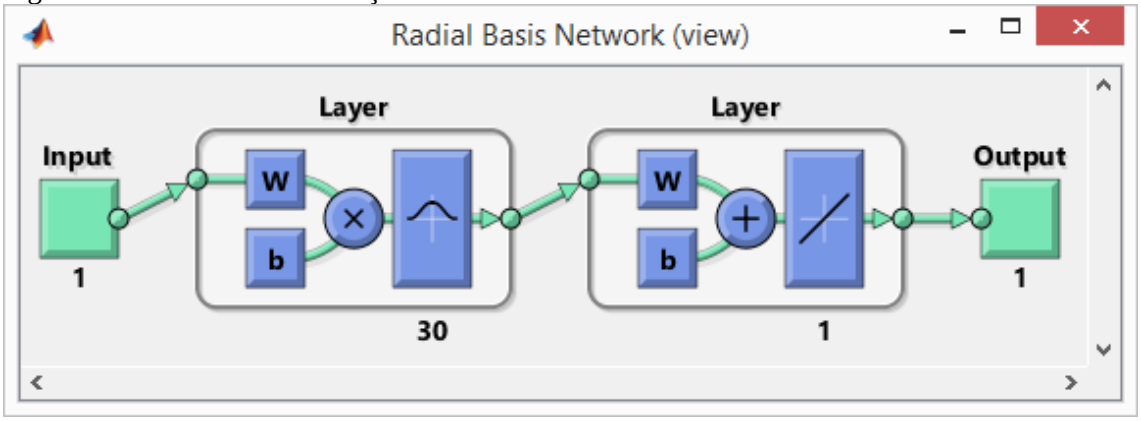
Cada camada de uma rede *RBF* desempenha um papel específico. A primeira camada, cujos neurônios utilizam funções de base radial, agrupa os dados em grupos (ou *clusters*), por meio de hiperelipsóides no espaço de padrões de entrada, diferentemente do *MLP* que particiona o espaço de entrada através de hiperplanos. Esta camada transforma um conjunto de padrões de entrada não-linearmente separáveis em um conjunto de saídas linearmente separáveis. A segunda camada, que é a camada de saída, procura classificar os padrões recebidos da camada anterior [4].

A distância euclidiana $\|x - \mu\|$ do vetor de entrada x ao vetor centro μ serve de entrada para a função, que retorna o valor de ativação da unidade intermediária. A resposta gerada em um neurônio k de saída será dada por (4).

$$y(x) = \sum_{i=1}^{k} w_{ji}\phi(\|x - \mu\|) + b \quad (4)$$

A Figura 3 mostra a representação de uma rede neural *RBF*.

Figura 3 – Rede Neural de Função de Base Radial



Fonte: o autor, 2014

Durante o projeto de redes neurais *RBF* é necessário definir o número de neurônios da camada intermediária. Uma alternativa para isso é definir o número de neurônios como igual ao número de padrões de entrada. Quando o número de funções radiais é igual ao número total de padrões de treinamento, cada centro pode ser situado sobre um vetor de entrada. Com isso, a rede mapeia com exatidão o vetor de entrada para a saída correta. Contudo, essa interpolação exata é indesejável, principalmente no caso de exemplos com ruído, pois pode levar ao *overfitting* [4].

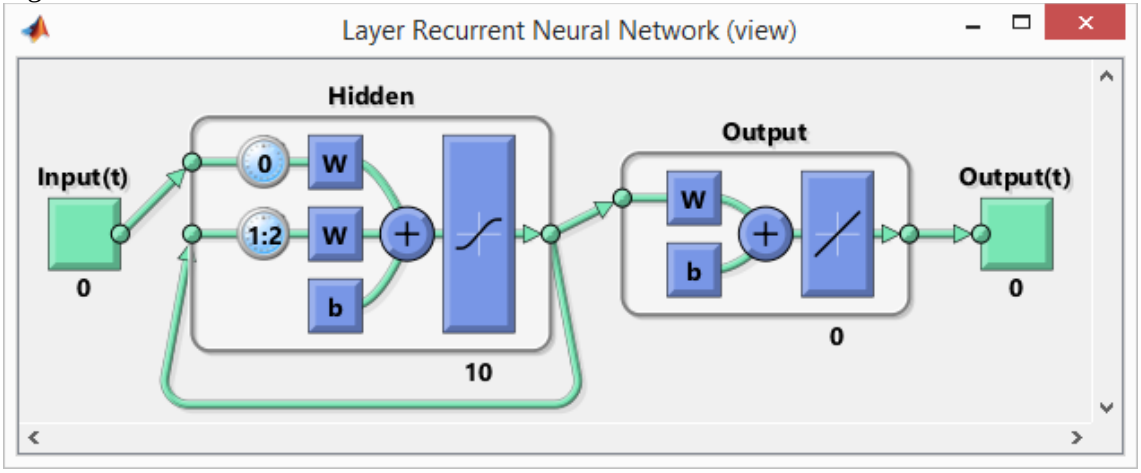
Redes função de base radial tendem a ter muito mais neurônios na camada oculta que as redes *Perceptron* de múltiplas camadas, uma vez que os neurônios com a função de ativação sigmoide produzem saída para uma ampla região do espaço de entrada, enquanto os neurônios radiais apenas respondem a uma região relativamente pequena deste mesmo espaço. Desta forma, quanto maior o espaço de entrada (em termos de números de entradas e alcance das mesmas) maior a quantidade de neurônios radiais requeridos.

Vários métodos têm sido propostos para o treinamento de redes neurais *RBF*. Na maioria desses métodos, o treinamento é classificado como híbrido, uma vez que é dividido em dois estágios. No primeiro estágio, o número de funções radiais e seus parâmetros são determinados por métodos não-supervisionados. O segundo estágio de treinamento ajusta os pesos dos neurônios de saída. Como a saída dos neurônios da camada intermediária é um vetor linearmente separável, os pesos podem ser determinados por modelos lineares, como a regra delta, mostrada no *Perceptron* de camada única [4][6].

2.4. REDE NEURAL DE CAMADA RECORRENTE

A rede neural de Camada Recorrente (*Layer-Recurrent Neural Network - LRN*) é uma versão generalizada da rede neural de Elman [15]. Nela há um *loop* de realimentação, com um *delay* simples, em cada camada da rede, exceto na última camada (Figura 4). A rede neural original de Elman tinha apenas duas camadas e usava a função de transferência tangente sigmoide hiperbólica na camada oculta e a função de transferência linear na camada de saída.

Figura 4 – Rede Neural de Camada Recorrente

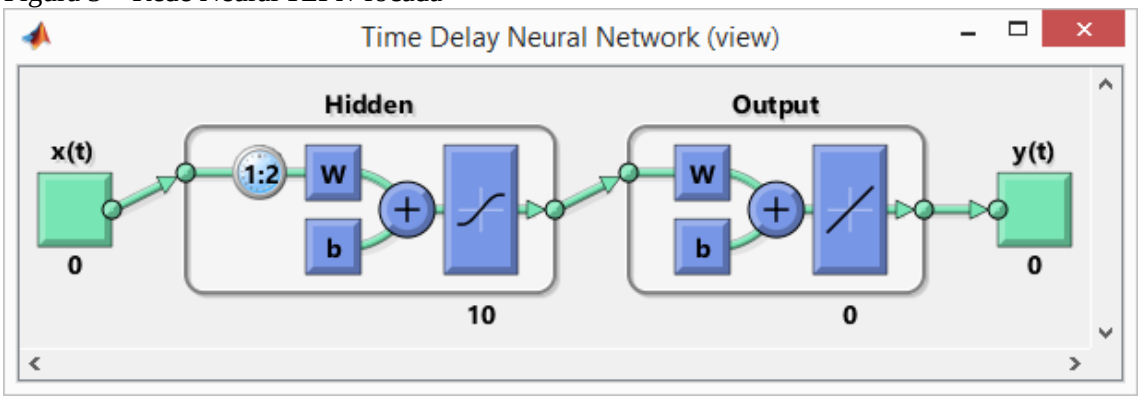


Fonte: o autor, 2014

2.5. REDE NEURAL ALIMENTADA ADIANTE FOCADA ATRASADA NO TEMPO

A rede neural alimentada adiante focada atrasada no tempo (*TLFN* focada, *focused time lagged feedforward neural network*, ou ainda, *FTDNN*, *focused time-delay neural network*) permite o processamento de padrões de séries temporais, pois possui uma memória de linha de atraso derivada (que consiste de p operadores de atraso unitário, caracterizados por z^{-1}, que operam sobre $x(n)$ produzindo sua versão atrasada $x(n-1)$, $x(n-2)$ e assim por diante) aliada ao *MLP* [6], como mostrado na Figura 5.

Figura 5 – Rede Neural *TLFN* focada



Fonte: o autor, 2014

O treinamento da rede *TLFN* focada se dá pelo algoritmo de retropropagação padrão, descrito anteriormente, com os dados de entrada correspondendo a $x(n)$ e seus p valores passados: $x(n-1), x(n-2), \ldots x(n-p)$.

3. RESULTADOS

Para as simulações foram utilizados um total de 496 cotações de fechamento do pregão para cada uma das quatro ações, no período que se estende desde 27 de fevereiro de 2012 até 25 de fevereiro de 2014. Tais valores foram normalizados, de acordo com (5), para que tivessem média igual a zero e desvio padrão igual a 1. Tal medida visa melhorar o desempenho durante o treinamento da rede, como proposto por LeCun, apud Haykin [6].

$$z = \frac{x - \mu}{\sigma} \quad (5)$$

onde x é o valor a ser normalizado e μ e σ são, respectivamente, a média e o desvio padrão da série.

Para cada ação foi simulada uma RNA com uma única camada oculta, com o número de neurônios variando entre 1 e 30 (incrementos de 1). Também foram feitas 30 repetições com cada rede, mantendo-se todos os valores inalterados, com exceção do conjunto de pesos iniciais, gerados pelo algoritmo de Nguyen-Widrow [12], que possui um componente inicial aleatório [15]. A exceção foi a rede *RBF*, que não altera seu conjunto de pesos iniciais no MATLAB, por isso para ela fez-se apenas 1 experimento. Foi definido como 500 o número máximo de iterações para as redes *MLP* e *TLFN* focada, e 1000 iterações para a rede *LRN*; este conceito não se aplica à rede *RBF*, uma vez que seu treinamento se dá com base em um número máximo de neurônios. Outro limitador do treinamento das redes foi o número de checagens de validação (*validation checks*), que interrompe o treinamento quando a melhora no *MSE* se torna insignificante; esse valor foi definido como 100 para as redes *MLP*, *TLFN* focada e *LRN* (a rede *RBF* não fez uso desse limitador). Para a rede *MLP* foi usada uma janela de previsão variando entre 1 e 5, já para as redes *LRN* e *TLFN* focada foi utilizada uma memória de linha de atraso, com operadores de atraso unitário variando entre 1 e 5 (todos com incremento de 1). Por fim, para a rede *RBF* também foi usada uma janela de previsão variando entre 1 e 5, e spread variando entre 1 e 10 (também com incrementos de 1).

Para as redes *MLP*, *TLFN* focada e *LRN* foram utilizadas as funções tangente sigmoide para a camada oculta e linear para a camada de saída. Para a rede *RBF* foi utilizada a função radial padrão do MATLAB, $radbas(n) = e^{-n^2}$.

Para o treinamento optou-se pelo método de *Levenberg-Marquardt*, descrito em THE MATHWORKS INC. [14]. Assim como os métodos quase-Newton, o algoritmo de *Levenberg-Marquardt* foi projetado para se aproximar, em velocidade, aos métodos de treinamento de segunda ordem sem ter que computar a matriz Hessiana.

Os melhores resultados obtidos por cada técnica são apresentados a seguir, nas Tabelas 2 e 3.

Tabela 2 – Desempenho das redes neurais *MLP* e *RBF*

Ação	***MLP***				***RBF***		
	Neurônios	**Janela**	***MSE***	**Tempo**	**Janela**	***MSE***	**Tempo**
VALE5	10	4	0,0408	1,13e+03	3	0,0385	30,55
PETR4	5	3	0,0664	1,21e+03	4	0,0645	30,56
ITUB4	2	4	0,0776	1,15e+03	4	0,0741	30,53
BBDC4	4	4	0,0502	1,14e+03	1	0,0534	25,12
VALE3	17	3	0,0675	1,18e+03	5	0,0652	30,39
ABEV3	8	3	0,1123	1,13e+03	5	0,1063	30,32
PETR3	7	3	0,0407	1,16e+03	4	0,0402	30,23
ITSA4	8	5	0,0486	1,20e+03	5	0,0477	30,23
BBAS3	22	2	0,0678	1,15e+03	4	0,0647	30,05
BVMF3	4	2	0,0574	1,13e+03	2	0,0565	30,06

Fonte: o autor, 2014

Tabela 3 – Desempenho das redes neurais *LRN* e *TLFN* focada

Ação	*LRN*				*TLFN* focada			
	Neurônios	**Janela**	***MSE***	**Tempo**	**Neurônios**	**Janela**	***MSE***	**Tempo**
VALE5	5	1	3,51e-11	6,73e+04	28	4	0,0394	1,19e+03
PETR4	17	2	3,13e-11	6,32e+04	30	5	0,0683	1,20e+03
ITUB4	15	2	1,85e-11	6,53e+04	19	1	0,0778	1,12e+03
BBDC4	11	3	9,50e-11	6,09e+04	15	4	0,0528	1,23e+03
VALE3	20	3	3,76e-12	6,49e+04	8	4	0,0676	1,28e+03
ABEV3	16	1	1,61e-11	6,04e+04	13	5	0,1111	1,26e+03
PETR3	23	2	3,43e-11	6,23e+04	3	4	0,0400	1,270+03
ITSA4	12	3	3,12e-11	6,94e+04	19	5	0,0494	1,40e+03
BBAS3	7	2	5,85e-11	6,01e+04	3	5	0,0667	1,28e+03
BVMF3	19	1	2,47e-11	6,27e+04	9	2	0,0544	1,37e+03

Fonte: o autor, 2014

Com base nos valores do erro quadrático médio, a melhor técnica de previsão foi a rede de camada recorrente *LRN*. Ela obteve valores de *MSE* na ordem de 10^{-11}, enquanto as outras técnicas ficaram na ordem de 10^{-1} e 10^{-2}. Isso corrobora o exposto por Braga, Carvalho e Ludermir [4] e Haykin [6], quando dizem que a estrutura da rede deve ser adequada à finalidade de predição dinâmica, seja por possuir operadores de atraso unitário ou recorrências. A rede *RBF* obteve resultados semelhante à rede *MLP* e *TLFN* focada, porém com um tempo expressivamente menor (cerca de 50 vezes menor).

Outra questão a se observar é a quantidade ótima de neurônios na camada oculta, que variou de ação para ação. Tal fato impossibilita a predição do número ótimo de neurônios na camada oculta por outro meio que não seja a experimentação.

4. CONCLUSÕES

Por utilizarem camadas ocultas com funções não lineares, as redes mostradas neste trabalho estão aptas a fazerem previsões de razoável acuidade mesmo em momentos de volatilidade no mercado, como se pode ver pelos resultados apresentados. Pode-se até mesmo dizer que momentos de maior variabilidade são mais fáceis de serem apreendidos por essas redes do que momentos de preços constantes. Isto permite sua utilização em um grande número de problemas econômicos, visto que devido ao grande número de agentes envolvidos em tais cenários, seu comportamento se torna extremamente não-linear, o que compromete os métodos clássicos de previsão.

É importante ressaltar a eficácia de recorrências na predição temporal comparativamente às outras técnicas (janelas de tempo e operadores de atraso unitário), ou seja, a estrutura da rede neural deve ser compatível com o processamento temporal para se obter um melhor resultado, como mostraram os resultados da rede *LRN*, em comparação com as outras redes.

O fato de janelas temporais maiores permitirem uma melhor predição pode indicar uma mais forte interdependência entre séries consecutiva de cotações, o que contradiz a ideia de eficiência do mercado de capitais, pelo menos para curtos espaços de tempo.

A restrição mais importante a se ressaltar quanto ao método das RNAs é o fato de não se poder definir de antemão o número ótimo de neurônios na camada oculta, pois aqui não vale a ideia de que quanto maior melhor. Apenas por meio de um experimento combinatório se pode dizer qual a quantidade ótima de neurônios para a camada oculta, dentro dos limites pré-estabelecidos pelo experimentador.

5. REFERÊNCIAS

[1] ASSAF NETO, Alexandre. **Mercado financeiro**. 9. ed. São Paulo: Atlas, 2009. 318p.

[2] BM&FBOVESPA. **Índice Bovespa – Ibovespa**. São Paulo, 2014. Disponível em: <http://www.bmfbovespa.com.br/indices/ResumoCarteiraTeorica.aspx?Indice=IBOVESPA&idioma=pt-br>. Acesso em: 25 jan. 2014.

[3] BM&FBOVESPA. **Metodologia do Índice Bovespa**. São Paulo, 2014. Disponível em: <http://www.bmfbovespa.com.br/Indices/download/Nova-Metodologia-do-Indice-Bovespa-R.pdf>. Acesso em 25 jan. 2014.

[4] BRAGA, Antônio de P.; CARVALHO, André P. de L. F.; LUDERMIR, Teresa B. **Redes neurais artificiais**: teoria e aplicações. 2. ed. Rio de Janeiro: LTC, 2011. 226p.

[5] FAUSETT, Laurene V. **Fundamentals of neural networks**: architectures, algorithms, and applications. 1. ed. USA: Prentice Hall, 1994. 461p.

[6] HAYKIN, Simon. **Redes neurais**: princípios e prática. 2. ed. Porto Alegre: Bookman, 2000. 903p.

[7] HAYKIN, Simon. **Neural Networks and Learning Machines**. 3. ed. New Jersey: Prentice Hall, 2009. 906p.

[8] KIERNAN, John A. **Neuroanatomia humana de Barr**. 7. ed. Barueri: Manole, 2003. 521p.

[9] KÖCHE, José Carlos. **Fundamentos de metodologia científica**. 29. ed. Petrópolis: Vozes, 2011. 182p.

[10] LAKATOS, Eva Maria; MARCONI, Marina de A. **Metodologia científica**. 6. ed. São Paulo: Atlas, 2011. 314p.

[11] PINHEIRO, Juliano L. **Mercado de capitais**: fundamentos e técnicas. 5. ed. São Paulo: Atlas, 2009. 500p.

[12] SKUTOVÁ, Jolana. **Weights Initialization Methods for MLP Neural Networks**. Ostrava, 2008. Disponível em: <http://transactions.fs.vsb.cz/2008-2/1636_SKUTOVA.pdf>. Acesso em: 11 fev. 2014.

[13] STEINER, Maria T. A. **Redes neurais artificiais**. 2012. 42 *slides*. Apresentação em *Power Point*.

[14] THE MATHWORKS INC. ***Levenberg-Marquardt Backpropagation***. Natick, 2014. Disponível em: <http://www.mathworks.com/help/nnet/ref/trainlm.html>. Acesso em: 04 jan. 2014.

[15] THE MATHWORKS INC. **Neural network toolbox user's guide**. Natick, 2013. Disponível em: <http://www.mathworks.com.au/help/pdf_doc/nnet/nnet_ug.pdf>. Acesso em: 02 jan. 2014.

[16] WORLD FEDERATION OF EXCHANGES. **Domestic Market Capitalization**. Paris, 2014. Disponível em: <http://www.world-exchanges.org/statistics/monthly-reports>. Acesso em: 07 jan. 2014.