New Strategies for Initialization and Training of Radial Basis Function Neural Networks

D. G. B. Franco and M. T. A. Steiner

***Abstract*— In this paper we proposed two new strategies for initialization and training of Radial Basis Function (RBF) Neural Network. The first approach takes into consideration the "error" between the input vector p of the network and the x-axis, which are the centers of radial functions. The second approach takes into account the "error" between the input vector p and the network output. In order to check the performances of these strategies, we used Brazilian financial market data for the RBF networks training, specifically the adjusted prices of the 10 greater weighted shares in the *Bovespa* index at the time of data collection – from April 8th, 2009 to October 31th, 2014. The first approach presented a 52% of improvement in the mean squared error (MSE) compared to the standard RBF network, while the improvement for the second approach was 38%. The strategies proved to be consistent for the time series tested, in addition to having a low computational cost. It is proposed that these strategies be improved by testing them with the Levenberg-Marquardt algorithm.**

***Keywords*— Radial Basis Function Neural Network, Financial Market, Time Series Forecasting.**

I. INTRODUÇÃO

COM a crescente globalização econômica, cada vez mais, o mercado financeiro se torna indispensável às economias nacionais. De acordo com o *World Federation of Exchanges* [1], o valor capitalizado do mercado à vista de ações em dezembro de 2015 era de mais de US$ 62 trilhões. Esse montante representa algo em torno de 80% do PIB (Produto Interno Bruto) mundial [2], o que mostra sua significância como potencial de alavancagem de empreendimentos.

Porém, existem riscos relacionados ao mercado de ações. Segundo Assaf Neto [3], 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."

Uma das áreas mais importantes da literatura financeira inclui estudos que tentam prever o comportamento do mercado de ações. Porém, como não é fácil quantificar fatores políticos, a maioria dos estudos se concentra nas variáveis econômicas onde há dados disponíveis, como distribuição de dividendos, taxas de juros, inflação, crescimento da produção industrial, entre outros [4]. Além disso, são muitas as técnicas de otimização disponíveis na literatura para tais previsões, dentre as quais destacam-se as redes neurais artificiais (RNAs).

O objetivo do presente artigo é propor duas novas estratégias para inicialização e treinamento da rede neural Função de Base Radial (*RBF*). Na primeira estratégia, a adição de neurônios à rede neural é feita a partir de um cálculo probabilístico, que leva em consideração o "erro" entre o vetor de entrada p e o próprio eixo x ($y = 0$, ou seja, a cada iteração o vetor "erro" permanece inalterado). Já na segunda estratégia, também probabilística, a adição de neurônios à rede acarreta em pequena diferença aos resultados, pois o "erro" é calculado entre o vetor de entrada p e a própria saída estimada pela rede neural (a cada iteração o vetor "erro" é alterado). Com o intuito de verificar o desempenho dessas estratégias, são aqui utilizados dados do mercado financeiro brasileiro, mais especificamente, os preços ajustados das 10 ações de maior peso no índice Bovespa, no decorrer do período de 8 de abril de 2009 a 31 de outubro de 2014, para realizar o treinamento das redes *RBF*.

O trabalho está organizado da seguinte maneira: na seção 2 apresenta-se a revisão de literatura sobre redes neurais artificiais e sua aplicação à predição de séries temporais; a seguir, na seção 3, é apresenta uma introdução às RNAs, seguida de uma descrição um pouco mais detalhada das *RBF* e a sua aplicação no *software* MATLAB; na seção 4 são apresentadas detalhadamente as duas novas estratégias de treinamentos propostas. Já na seção 5 é apresentada a metodologia de seleção e tratamento dos dados para torná-los *inputs* da rede neural; na seção 6 são apresentados e discutidos os resultados, tanto em termos de desempenho das redes neurais testadas quanto em relação à sua eficácia em prever valores reais de cotações, do mundo "real"; por fim, na seção 7 conclui-se o trabalho e deixam-se sugestões para trabalhos futuros.

II. TRABALHOS CORRELATOS: PREVISÃO DE SÉRIES TEMPORAIS

O tempo é uma variável importante em várias tarefas de reconhecimento de padrões, tais como previsão de séries temporais, reconhecimento de voz, detecção de movimento, visão [5], [6]. Porém muitos algoritmos de treinamento de RNAs não são capazes de implementar mapeamentos temporais dinâmicos como, por exemplo, o algoritmo *backpropagation* (algoritmo por retropropagação do erro), que pode aprender apenas mapeamentos estáticos. Um artifício utilizado para conseguir o processamento temporal utilizando essas redes envolve o uso de janelas de tempo, no qual a rede utiliza trechos sequenciais dos dados temporais como se eles formassem um padrão estático. Há extensa literatura a respeito da previsão de séries temporais por RNAs, isoladamente ou de forma híbrida,

D. G. B. Franco, Pontifícia Universidade Católica do Paraná (PUC-PR), Curitiba, Paraná, Brasil, david.barros@pucpr.br

M. T. A. Steiner, Pontifícia Universidade Católica do Paraná (PUC-PR), Curitiba, Paraná, Brasil, maria.steiner@pucpr.br

em combinação com outras técnicas. A seguir, têm-se alguns exemplos destas abordagens dos últimos três anos.

Gomes e Ludermir [7] propuseram uma rede neural treinada por *Levenberg-Marquardt* com otimização de seus parâmetros por *Simulated Annealing* e *Tabu Search*. Já Song *et al*. [8] propuseram uma rede neural otimizada por *Modified Particle Swarm Optimization* (*MPSO*), que introduz um operador mutacional adaptativo ao algoritmo *PSO* (*Particle Swarm Optimization*).

Uma rede neural com parâmetros otimizadas por *Particle Swarm Optimization*, para três séries temporais sazonais, foi proposta por Adhikari, Agrawal e Kant [9]. Em Yu, Chen e Tang [10], teve-se a aplicação de uma rede de *Elman* à predição da defasagem temporal (*time delay*) entre comunicações baseadas na Internet, para dados decompostos pelo algoritmo *EMD* (*Empirical Mode Decomposition*).

Ouyang e Yin [11] propuseram uma rede neural não-causal ponderada, ou seja, que utilizou, além dos valores passados, também os valores futuros, ambos ponderados para atribuir maior peso aos *inputs* mais atuais, que desempenharam um papel mais crítico nos processos de treinamento e predição.

Foi proposta por Chand e Chandra [12] uma rede neural multiobjetivo treinada por estratégias evolutivas. Já em Smith e Jin [13] foi proposto o uso de redes neurais recorrentes em conjuntos (*ensembles*), gerados por estratégia evolutiva multiobjetivo.

Por sua vez, Donate e Cortez [14] propuseram duas novas abordagens de redes neurais evolutivas (*EANN*) para seleção automática de parâmetros da rede: rede neural evolutiva esparsamente conectada (*SEANN*), que gerou uma arquitetura de rede para previsão vários passos à frente, e rede neural evolutiva com seleção automática de defasagem temporal (*TEANN*) que, além da arquitetura da rede, também selecionou, automaticamente, a defasagem temporal das séries que alimentarão a rede.

Em Wang, Zeng e Chen [15] temos a proposta de redes neurais *MLP* com parâmetros selecionados por evolução diferencial adaptativa (*ADE*). E em Du, Leung e Kwong [16], de redes neurais otimizadas por variações do algoritmo de evolução diferencial adaptativa.

O trabalho de Huang e Zhang [17], por sua vez, utiliza uma rede neural dinâmica, enquanto que o trabalho de Egrioglu, Aladag e Yolcu [18,19] propõe um método híbrido, que combina *fuzzy c-means* e redes neurais.

Hamzaçebi, Akay e Kutay [20] utilizaram redes neurais com estratégia direta de previsão vários passos à frente (em oposição às estratégias iterativas). Em Ruiz-Aguilar, Turias e Jiménez-Come [21], testou-se o uso de redes neurais híbridas, em conjunto com o modelo *SARIMA*, o qual é inicialmente aplicado para capturar os padrões lineares e sazonais da série temporal. Por fim, Kourentzes, Barrow e Crone [22], utilizaram conjuntos de redes neurais para previsão.

III. RNAS, *RBFS* E SUA UTILIZAÇÃO NO MATLAB

Nesta seção é realizada uma breve introdução às RNAs, seguida de uma descrição um pouco mais detalhada das *RBF* e a sua aplicação por meio do *software* MATLAB.

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*. Já o primeiro estudo sobre o aprendizado de RNA 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 bastante utilizada até os dias de hoje [5, 23].

O Perceptron, em sua forma simples, apareceria em 1958, proposto por Frank Rosenblatt e o algoritmo *backpropagation* surgiria somente em 1986, proposto por Rumelhart e colaboradores [5, 23]. Desde então, centenas de trabalhos na área de RNAs vêm sendo desenvolvidos.

A função de ativação aplicada a cada neurônio da maioria das redes multicamadas, como as *MLPs*, 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 dada em função da distância entre seus vetores de entrada e de peso. Uma dessas redes é a rede neural *RBF*. Este nome se deve à utilização, pelos neurônios da camada oculta, serem de funções de base radial [6, 24].

Cada uma das camadas de uma rede neural *RBF* executa um papel específico. A camada inicial de processamento, cujos neurônios utilizam funções de base radial, agrupa os dados em grupos (*clusters*) por meio de hiperelipsóides no espaço de entrada dos padrões da rede, diferentemente das redes neurais *MLP* que particionam o espaço de entrada através de hiperplanos. Esta camada radial transforma um conjunto de padrões de entrada não-linearmente separáveis em um conjunto de saídas linearmente separáveis. A camada seguinte da rede *RBF*, que utiliza funções lineares, classifica os padrões recebidos da camada anterior [5].

Várias metodologias têm sido propostas para o treinamento de redes *RBF*. Em muitos destes métodos, o treinamento é 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. No segundo estágio são ajustados os pesos dos neurônios de saída, lineares. Como a saída dos neurônios da camada radial é um vetor linearmente separável, os pesos podem ser determinados por modelos lineares, como a regra delta [5, 25].

Algumas abordagens foram propostas para a primeira fase de treinamento como, por exemplo, selecionar os centros dos agrupamentos aleatoriamente a partir dos padrões de treinamento ou utilizar técnicas de agrupamento (*clustering*), como o algoritmo *K-means-clustering* [5].

Enquanto que no *MLP* os parâmetros da rede são, geralmente, adaptados simultaneamente por um processo de otimização, de forma supervisionada (mensurando-se a diferença entre a resposta produzida pela rede e o alvo), nas redes *RBF* isso pode ser feito em dois estágios [26]:

Ajustando-se os parâmetros da camada intermediária, radial, incluindo os centros das funções radiais, suas respectivas aberturas (*spread* (σ) – ver equação (2)) e os pesos;

Calculando-se os pesos da camada de saída, linear.

O MATLAB possui algumas peculiaridades quanto à implementação da rede *RBF*, que estão descritas a seguir [27]: primeiramente é calculado o vetor de "erros" (também chamado vetor protótipo) para cada neurônio da camada intermediária por meio da equação (1), e em seguida, seleciona-se o maior "erro" e (relacionado à entrada p e ao alvo t) deste vetor.

É adicionado o primeiro neurônio à camada oculta da rede na posição (relativa ao vetor de entrada p) onde o maior erro e do vetor de "erros" se encontra; os pesos sinápticos $IW^{1,1}$ são então definidos como e. Desse modo, a rede *RBF* aqui testada não possui componente aleatório na geração de seu conjunto de pesos iniciais (uma vez que o vetor de "erros" e depende apenas da entrada p, do alvo t e do *spread*), não sendo necessária a realização de múltiplos testes visando um conjunto de pesos iniciais que melhor se adapte ao conjunto de dados.

$$e = (P' * d)'.^2/(dd * PP'), \text{ com} \quad (1)$$
$$P = \left(radbas(dist(p', p)) * \sqrt{-\log 0.5}\right)/spread$$
$$d = t'$$
$$dd = sum(t'.* t')'$$
$$PP = sum(P.* P)'$$

Onde '.' representa operações entre vetores (elemento por elemento) e ''' representa a transposta do vetor/matriz**.** A operação $dist$ recebe o vetor p (dimensão Rx1) de entrada e a matriz de pesos de entrada $IW^{1,1}$ (S^1xR) e produz um vetor contendo S^1x1 elementos. Estes elementos são a distância euclidiana entre o vetor de entrada p e o vetor $IW^{1,1}$ formada pelas colunas da matriz de pesos de entrada $IW^{1,1}$.

O vetor *bias* b^1 (S^1x1) (definido como $\sqrt{-\log 0.5}/spread$) e a saída de $dist$ (S^1x1) são multiplicados, elemento por elemento, e produzem o elemento n^1 (S^1x1) (o *bias* b^1 permite ajustar a sensibilidade do neurônio de base radial). Este elemento n^1 é então aplicado à função radial, que no caso do MATLAB é dada pela equação (2), resultando na saída a^1.

$$radbas(n) = e^{-n^2} = e^{(-n^2/2*\sigma^2)}, para\ \sigma = \sqrt{1/2} \quad (2)$$

A função de base radial atinge o máximo em "1" quando a entrada é "0". À medida que a distância entre o peso w e a entrada p diminui, a saída da função aumenta. Assim, um neurônio da base radial atua como um detector que produz "1" sempre que a entrada p é idêntica a seu peso no vetor w.

Os pesos e *bias* da segunda camada, $LW^{2,1}$ e b^2, respectivamente, são calculados simulando-se a saída da primeira camada, a^1, e então resolvendo o sistema linear dado pela equação (3).

$$[LW^{2,1}\ b^2] * [a^1; ones(1, Q)] = T \quad (3)$$

Onde Q representa o comprimento do vetor de entrada p, T é o vetor alvo t e $ones(1, Q)$ representa uma matriz preenchida com "1" e tamanho 1xQ.

Como se conhece a^1 e t, e a camada é linear, pode-se calcular os pesos para a segunda camada, que minimizem a soma do erro quadrático, isolando-se $[LW^{2,1}\ b^2]$, conforme a equação (4).

$$Wb = T/[a^1; ones(1, Q)] \quad (4)$$

Com Wb representando o conjunto de pesos e *bias* $[LW^{2,1}\ b^2]$.

Os *bias* da segunda camada, b^2, são somados ao produto de a^1 por $LW^{2,1}$ e produzem o elemento n^2 (S^2x1). Este elemento n^2 é, então, aplicado à função linear, dada pela equação (5), resultando na saída a^2.

$$purelin(n) = n \quad (5)$$

IV. PROPOSTA DE NOVAS ESTRATÉGIAS DE INICIALIZAÇÃO DA REDE *RBF*

Uma vez que o MATLAB utiliza apenas os componentes p (entrada) e t (alvo) para calcular o "erro" a partir do qual serão definidos os centros e pesos das funções radiais, não há componente aleatório, e para um mesmo conjunto de dados haverá uma mesma inicialização da rede. Pensando nisso, optamos por uma estratégia que levasse em conta um componente aleatório para a seleção destes parâmetros – centros e pesos.

A partir da equação (6), foi calculado o novo vetor de "erros".

$$e = |p - t|^{\alpha} \quad (6)$$

Onde $p = 0$ para todas as iterações (estratégia 1); $p = saída\ da\ rede\ na\ iteração\ anterior$ (estratégia 2; $p = 0$ na primeira iteração); e t são os alvos da rede.

O parâmetro α permite acentuar ($\alpha > 1$) ou suavizar ($0 < \alpha < 1$) a função resultante.

A partir desse novo vetor e faz-se uma seleção probabilística dos pesos da rede, que leva em consideração seus maiores componentes, de acordo com o algoritmo:

1. Os erros são ordenados de forma decrescente (suas posições originais são armazenadas para posterior uso);
2. É calculada a probabilidade de seleção do erro proporcional a seu valor;
3. O erro é selecionado comparando-se sua probabilidade de seleção e um número pseudoaleatório gerado a partir de uma distribuição uniforme, no intervalo $(0,1)$; este erro será o peso do neurônio;
4. A posição do erro no vetor, armazenada na etapa 1, é utilizada como centro da função radial.

A Fig. 1, a seguir, compara a localização dos centros radiais, para os métodos padrão e para as estratégias 1 e 2.

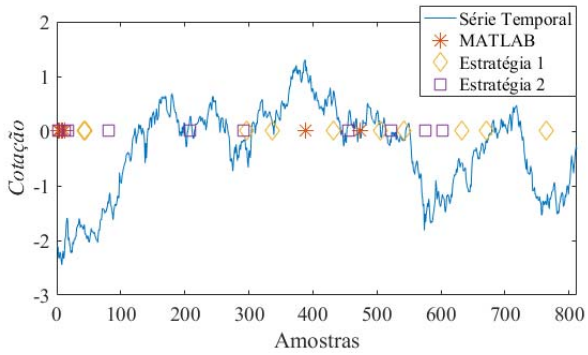

Figura 1. Localização dos centros pelo método original (MATLAB) e pelas estratégias 1 e 2.

Percebe-se que, na rede padrão, os centros das funções radiais estão concentrados aproximadamente nos pontos onde a série temporal apresenta os maiores e menores valores e que a estratégia 1 é a que apresenta a distribuição mais uniforme dos centros das funções radiais.

V. METODOLOGIA

As séries temporais utilizadas nos testes foram as 10 ações de maior participação na composição do Ibovespa, o qual representa o valor atual, em moeda corrente, de uma carteira teórica de ações constituída em 2/1/1968.

A participação de cada ação no Ibovespa é calculada pelo índice de negociabilidade (IN), conforme a equação (7), que busca a representatividade desse título em termos de número de negócios e volume financeiro [28].

$$IN = \sqrt[3]{n_i/N} * \sqrt[3]{(v_i/V)^2} * p_i/P \quad (7)$$

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); p_i é o número de pregões em que o ativo foi negociado e P é o número de pregões total do período analisado.

Assim sendo, as ações escolhidas e suas respectivas participações na composição do Ibovespa, por ocasião da pesquisa, estão apresentadas na Tabela I.

TABELA I
AÇÕES ESCOLHIDAS E RESPECTIVOS PERCENTUAIS DE PARTICIPAÇÃO

ITUB4	9,770%	*Itaú Unibanco Holding S.A.,* setor bancário
PETR4	8,042%	*Petróleo Brasileiro S.A.*, setor de petróleo, gás e energia
BBDC4	7,330%	*Banco Bradesco S.A.*, setor bancário
VALE5	5,710%	*Vale S.A.*, setor de mineração
ABEV3	5,397%	*AMBEV S.A.*, setor de bebidas
PETR3	5,048%	*Petróleo Brasileiro S.A.*, setor de petróleo, gás e energia
VALE3	4,289%	*Vale S.A.*, setor de mineração
BRFS3	3,617%	*BRF S.A.*, setor de alimentos processados
ITSA4	3,041%	*Itausa Investimentos Itaú S.A.*, gestão de participações societárias
CIEL3	2,880%	*Cielo S.A.*, serviços financeiros

Para as simulações foram utilizadas um total de 1.355 cotações de fechamento, ajustadas, para cada uma das dez ações, no período que se estende desde 8 de abril de 2009 até 31 de outubro de 2014. Esse valor de 1.355 corresponde ao menor histórico disponível (CIEL3), de modo que todas as redes testadas tivessem a mesma quantidade de padrões para treinamento e teste (60% treinamento e 40% teste). Tais valores foram normalizados, de acordo com a equação (8), para que tivessem média igual a "0" e desvio padrão igual a "1". Esta medida visa melhorar o desempenho da rede durante o seu treinamento [25].

$$z = x - \mu/\sigma \quad (8)$$

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

O problema de previsão de séries temporais é a busca pela relação entre valores y no período t, representado por y_t, e valores de períodos anteriores $\{t-1, t-2, \dots, t-k\}$, como definido pela equação (9), a seguir [29].

$$y_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-k}) \quad (9)$$

Onde y_t denota a previsão dada pela rede neural e k a defasagem temporal – *time delay*.

A Fig. 2 ilustra o pré-processamento de uma série temporal até se chegar aos conjuntos de treinamento e teste, para $k = 3$.

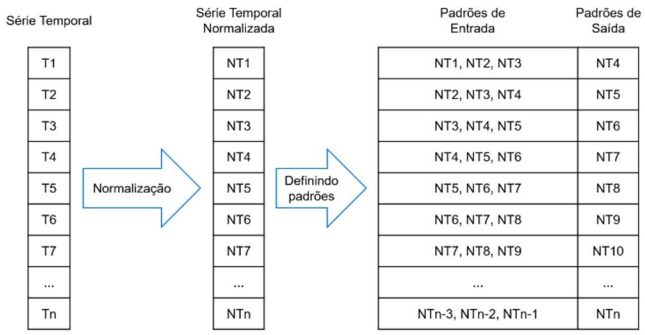

Figura 2. Pré-processamento dos *inputs* da RNA.

Para todas as redes aqui testadas foram utilizadas defasagens temporais, para a entrada da rede, no intervalo $[1,5]$, ou seja, $k \in [1,5]$, uma vez que a análise de autocorrelação indicou que a correlação das séries aqui testadas com suas versões atrasadas diminuía à medida que a defasagem aumentava. Para cada rede *RBF* foi utilizado um *spread* (σ) com valores variando no intervalo $[1, 10]$, igual para todas as funções radiais numa mesma rede.

Para a rede *RBF* padrão, que não altera seu conjunto de pesos iniciais, foi realizado apenas um teste. Para as novas estratégias de inicialização foram realizadas 10 repetições para cada rede, uma vez que possuem um componente aleatório em sua inicialização, e selecionado o melhor resultado. O número máximo de neurônios foi definido como sendo igual a 20.

VI. OBTENÇÃO DOS RESULTADOS

A medição da acurácia da rede foi feita através do *MSE* para o conjunto de testes. Os melhores resultados obtidos por cada técnica são apresentados a seguir (Tabelas II a IV). A coluna *Simulação* (Tabelas III e IV; para a rede *RBF* padrão, Tabela II, fez-se apenas uma simulação, visto que a rede não altera seu conjunto de pesos iniciais) indica em qual das 10 variações de pesos iniciais foi obtido o melhor resultado; a coluna *Delay* indica a defasagem temporal da série que alimentou a rede neural; a coluna *MSE* refere-se ao erro do conjunto de testes; a coluna *T* mostra o tempo total (em segundos) necessário para a execução das simulações, ou seja, o tempo necessário para se encontrar a melhor solução. As redes *RBF* padrão não realizaram nenhuma repetição, por isso seus tempos são expressivamente menores. Os testes foram realizados em um computar com as seguintes especificações: processador Intel Core i7-2600 3,4 GHz; memória *ram* de 16 GB 1333 MHz; *SSD* de 128 GB; Windows 8.1; MATLAB R2014a.

TABELA II
DESEMPENHO DA REDE *RBF* PADRÃO

Ação	*Spread*	*Delay*	*MSE*	T(s)
ITUB4	7	5	0,0277	21,41
PETR4	8	4	0,0121	40,16
BBDC4	7	5	0,0616	32,34
VALE5	9	5	0,0106	39,14
ABEV3	8	1	0,0708	28,49
PETR3	2	4	0,0044	40,64
VALE3	9	5	0,0092	31,31
BRFS3	6	5	0,0656	41,45
ITSA4	8	5	0,0273	42,57
CIEL3	9	5	0,0476	19,95

TABELA III
DESEMPENHO DA REDE *RBF* MODIFICADA (ESTRATÉGIA 1)

Ação	Simulação	*Spread*	*Delay*	*MSE*	T(s)
ITUB4	8	3	4	0,0250	310,56
PETR4	9	8	5	0,0080	305,94
BBDC4	1	4	4	0,0420	256,96
VALE5	8	10	5	0,0101	314,91
ABEV3	2	2	5	0,0037	316,81
PETR3	9	9	5	0,0039	277,64
VALE3	9	9	4	0,0088	185,45
BRFS3	2	2	5	0,0149	209,67
ITSA4	7	10	5	0,0270	198,97
CIEL3	6	10	5	0,0166	249,99

TABELA IV
DESEMPENHO DA REDE *RBF* MODIFICADA (ESTRATÉGIA 2)

Ação	Simulação	*Spread*	*Delay*	*MSE*	T(s)
ITUB4	9	6	5	0,0246	267,39
PETR4	1	10	5	0,0083	206,87
BBDC4	9	5	5	0,0486	281,08
VALE5	4	10	5	0,0099	271,27
ABEV3	4	3	4	0,0071	214,46
PETR3	9	2	3	0,0040	268,95
VALE3	6	4	4	0,0089	219,37
BRFS3	3	2	5	0,0160	267,85
ITSA4	10	10	4	0,0272	215,02
CIEL3	9	5	4	0,0549	283,33

As redes *RBF* padrão obtiveram um *MSE* médio de 0,0337, considerando as 10 ações; as redes *RBF* modificadas segundo a estratégia 1, por sua vez, obtiveram um *MSE* médio de 0,0160, ou seja, com uma melhoria de aproximadamente 52% em relação às redes *RBF* padrão; enquanto que as redes *RBF* modificadas segundo a estratégia 2 obtiveram um *MSE* médio de 0,0210, uma melhoria de 38% em relação à *RBF* padrão. Quanto ao tempo, as redes *RBF* padrão foram as que obtiveram os melhores resultados, uma vez que não precisaram realizar repetições: enquanto as redes *RBF* padrão precisaram de 33,75 segundos, em média, as redes *RBF* modificadas pela estratégia 1 precisaram de 262,69 segundos e as redes *RBF* modificadas pela estratégia 2 precisaram de 249,56 segundos.

É interessante notar que, embora tenham executado 10 simulações, as redes modificadas necessitaram de um tempo proporcional (para cada simulação) 25% menor àquele das redes padrão do MATLAB. Isso porque utilizaram um modelo simplificado de cálculo do vetor de "erros".

A Fig. 3, a seguir, compara o melhor (0,0037; ABEV3, estratégia 1) e o pior (0,0549; CIEL3, estratégia 2) resultados obtidos pelas estratégias propostas em relação aos valores reais das cotações. A linha vertical vermelha faz a separação entre os conjuntos de treinamento e teste. Nota-se que o conjunto de teste apresenta pior desempenho (embora as curvas real e predita sejam praticamente coincidentes) em relação ao conjunto de treinamento, em ambas as situações, uma vez que não foi apresentado à rede e sua função é, justamente, verificar a capacidade de generalização (extrapolação) da rede neural.

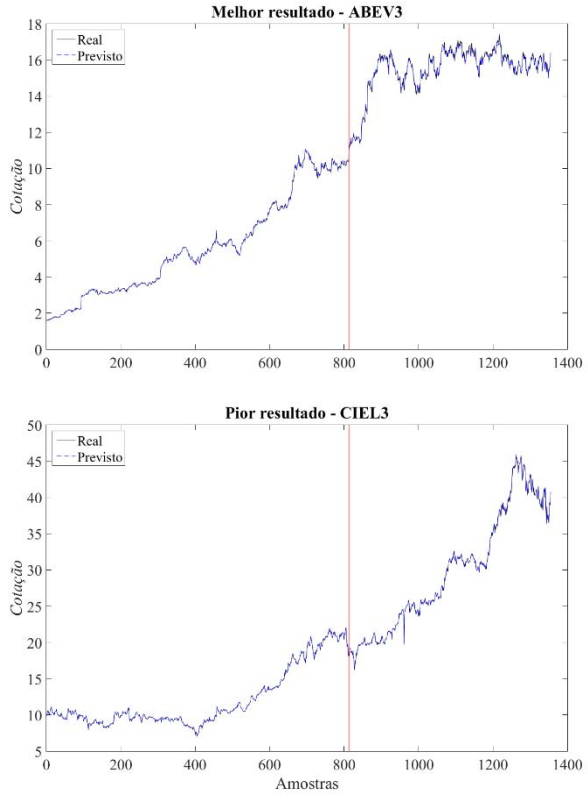

Figura 3. Comparação dos valores reais com os resultados obtidos pelas estratégias 1 e 2 (melhor e pior resultados, respectivamente).

Os resultados obtidos para as demais ações foram similares e os devios-padrão para cada estratégia foram: 0,0256 (padrão

MATLAB); 0,0121 (Estratégia 1) e 0,0180 (Estratégia 2). Mais uma vez, a Estratégia 1 se mostrou superior, uma vez que apresentou resultados restritos a um intervalo menor que as outras estratégias.

A Fig. 4, a seguir, compara o intervalo de distribuição dos resultados para as respectivas estratégias. Em cada um dos gráficos da Fig. 4, a linha vermelha marca a mediana e a caixa azul delimita o intervalo entre o 25º e o 75º percentis. As linhas tracejadas representam os pontos fora deste intervalo, ou seja, os extremos mínimo e máximo.

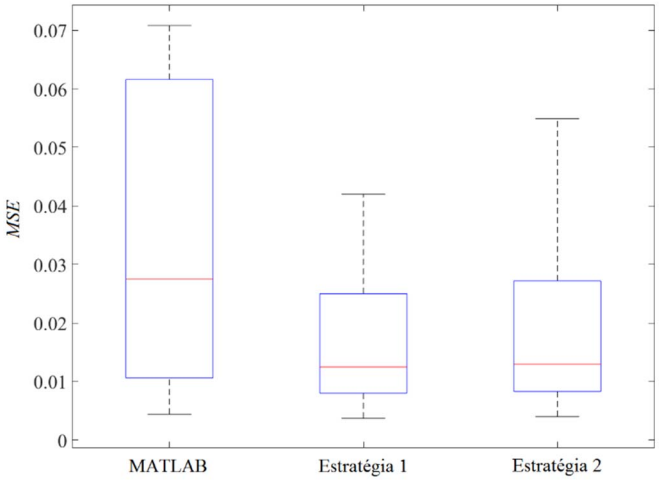

Figura 4. Comparação do intervalo de distribuição dos resultados para cada estratégia.

VII. CONCLUSÕES

As estratégias de inicialização propostas para as redes *RBF* se mostraram eficientes, uma vez que proporcionaram uma redução de 52% (estratégia 1) e 38% (estratégia 2) no *MSE*, se comparadas à versão padrão do MATLAB. Comparativamente aos resultados obtidos por outros pesquisadores, cujos *MSEs* variaram em função das séries temporais testadas e seu respectivo comprimento e que estão compreendidos no intervalo $[5{,}74e-5; 1{,}426e8]$, as duas estratégias aqui propostas, cujos *MSEs* ficaram compreendidos no intervalo $[0{,}0037; 0{,}0549]$, se mostraram consistentes, uma vez que foram testadas para 10 séries temporais distintas. As estratégias são de simples implementação e de baixo custo computacional, embora necessitem de um tempo total maior para alcançar a melhor solução, o que pode ser uma limitação, dependendo da aplicação.

Resta saber como se comportariam para séries temporais de menor tamanho, que dificultam o aprendizado por não possuírem exemplos suficientes para treinamento, validação e teste.

Além disso, propõe-se que estas estratégias sejam exploradas e aperfeiçoadas em trabalhos futuros, onde se poderia realizar o treinamento das redes *RBF* pelos algoritmos de *Levenberg-Marquardt*, tanto pela estratégia de interrupção precoce (*early stopping*), que interrompe o treinamento caso não haja decremento significativo no erro do conjunto de validação, quanto pela regularização Bayesiana, que busca minimizar, também, a média da soma quadrática dos pesos e *bias* da rede.

REFERÊNCIAS

[1] World Federation of Exchanges, *Domestic Market Capitalization*. FRA, 2016. Disponível em: <http://www.world-exchanges.org/home/index.php/statistics/monthly-reports>. Acesso em: 02 fev. 2016.

[2] World Bank, *World Development Indicators database*. USA, 2015. Disponível em: < http://databank.worldbank.org/data/download/GDP.pdf>. Acesso em: 02 fev. 2016.

[3] A. Assaf Neto, Mercado financeiro. 9. ed. SP: Atlas, 2009. 318p.

[4] S. Hamori, An empirical investigation of stock markets: the CCF approach. *Research Monographs in Japan-U.S. business & Economics*. USA: Springer Science+Business Media, 2003. 130p.

[5] P. Braga, A. P. L. F. Carvalho, T. B. Ludermir, *Redes neurais artificiais: teoria e aplicações*. 2. ed. RJ: LTC, 2011. 226p. BRAGA, Antônio de P.; CARVALHO, André P. de L. F.; LUDERMIR, Teresa B. *Redes neurais artificiais: teoria e aplicações*. 2. ed. RJ: LTC, 2011. 226p.

[6] M. Tkác, R. Verner, Artificial neural networks in business: Two decades of research. *Applied Soft Computing*, v. 38, p. 788-804, 2016.

[7] G. S. S. Gomes, T. B. Ludermir, Optimization of the weights and asymmetric activation function Family of neural network for time series forecasting. *Expert Systems with Applications*, v. 40, p. 6438-6446, 2013.

[8] L. Song, H. Qing, Y. Ying-Ying, L. Hao-Ning, *Prediction for chaotic series of optimized BP neural network based on modified PSO*. In: 26th Chinese Control and Decision Conference (CCDC '14), CHN, p. 697-702, 2014.

[9] R. Adhikari, R. K. Agrawal, L. Kant, *PSO based Neural Network vs. traditional statistical models for seasonal time series forecasting*. In: IEEE 3rd International Advance Computing Conference (IACC '13), IND, p. 719-725, 2013.

[10] F. Yu, D. Chen, X. Tang, *Time Delay Prediction Method Based on EMD and Elman Neural Network*. In: 6th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC '14), CHN, v. 2, p. 368-371, 2014.

[11] Y. Ouyang, H. Yin, *Time series prediction with a non-causal neural network*. In: IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr '14), UK, p. 25-31, 2014.

[12] S. Chand, R. Chandra, *Multi-objective cooperative coevolution of neural networks for time series prediction*. In: International Joint Conference on Neural Networks (IJCNN '14), CHN, p. 190-197, 2014.

[13] C. Smith, Y. Jin, Evolutionary multi-objective generation of recurrent neural network ensembles for time series prediction. *Neurocomputing*, v. 143, p. 302-311, 2014.

[14] J. P. Donate, P. Cortez, Evolutionary optimization of sparsely connected and time-lagged neural networks for time series forecasting. *Applied Soft Computing*, v. 23, p. 432-443, 2014.

[15] L. Wang, Y. Zeng, T. Chen, Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. *Expert Systems with Applications*, v. 42, p. 855-863, 2015.

[16] W. Du, S. Y. S. Leung, C. K. Kwong, Time series forecasting by neural networks: A knee point-based multiobjective evolutionary algorithm approach. *Expert Systems with Applications*, v. 41, p. 8049-8061, 2014.

[17] M. Huang, R. Zhang, *The application of dynamic intelligent neural network in time series forecasting*. In: International Conference on Electrical and Control Engineering (ICECE '11), CHN, p. 2630-2633, 2011.

[18] E. Egrioglu, C. H. Aladag, U. Yolcu, Fuzzy time series forecasting with a novel hybrid approach combining fuzzy c-means and neural networks. *Expert Systems with Applications*, v. 40, p. 854-857, 2013.

[19] E. Egrioglu *et al.*, Fuzzy time series forecasting based on Gustafson-Kessel fuzzy clustering. *Expert Systems with Applications*, v. 38, p. 10355-10357, 2011.

[20] C. Hamzaçebi, D. Akay, F. Kutay, Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert Systems with Applications*, v. 36, p. 3839-3844, 2009.

[21] J. J. Ruiz-Aguilar, I. J. Turias, M. J. Jiménez-Come, Hybrid approaches based on SARIMA and artificial neural networks for inspection time series forecasting. *Transportation Research Part E*, v. 67, p. 1-13, 2014.

[22] N. Kourentzes, D. K. Barrow, S. F. Crone, Neural network ensemble operators for time series forecasting. *Expert Systems with Applications*, v. 41, p. 4235-4244, 2014.

[23] A. Almási, S. Wozniak, V. Cristea, Y. Leblebici, T. Engbersen, Review of advances in neural networks: Neural design technology stack. *Neurocomputing*, v. 174, p. 31-41, 2016.
[24] S. Galeshchuk, Neural networks performance in exchange rate prediction. *Neurocomputing*, v. 172, p. 446-452, 2016.
[25] S. Haykin, *Redes neurais: princípios e prática*. 2. ed. RS: Bookman, 2000. 903p.
[26] F. Schwenker, H. A. Kestler, G. Palm, Three learning phases for radial-basis-function networks. *Neural Networks*, USA, v. 14, n. 4-5, 2001.
[27] THE MATHWORKS INC, *Neural network toolbox user's guide*. USA, 2014. Disponível em: <http://www.mathworks.com.au/help/pdf_doc/nnet/nnet_ug.pdf>. Acesso em: 02 out. 2014.
[28] BM&FBOVESPA, *Metodologia do Índice Bovespa*. SP, 2014c. Disponível em: <http://www.bmfbovespa.com.br/Indices/download/Nova-Metodologia-do-Indice-Bovespa-R.pdf>. Acesso em 25 jul. 2014.
[29] J. P. Donate, X. Li, G. G. Sánchez, A. S. de Miguel, Time series forecasting by evolving artificial neural networks with genetic algoriths, differential evolution and estimation of distribution algorithms. *Neural Computing and Applications*, v. 22(1), p. 11-20, 2013.

David Gabriel de Barros Franco possui graduação em Engenharia de Produção pela Pontifícia Universidade Católica do Paraná (PUCPR; 2008/2012) e mestrado em Engenharia de Produção e Sistemas também pela PUCPR (2013/2015). Atualmente é doutorando em Engenharia de Produção e Sistemas pela PUCPR (2015/2019). Atua na área de Engenharia de Produção, com ênfase em Pesquisa Operacional, subáreas: Modelagem, Análise e Simulação de Sistemas Produtivos (*software* ARENA); Processos Decisórios; Análise e Previsão de Séries Temporais; Sistemas Inteligentes (Redes Neurais Artificiais); Programação Matemática e Otimização (Programação Linear, Inteira, Mista e Não-Linear; Otimização Global: Algoritmos Genéticos, *Simulated Annealing*, *Tabu Search*, *Direct Search*, *Multiobjective Optimization*, *Swarm Optimization*, dentre outras técnicas); e Mineração de Dados (*Data Mining* e *Big Data*). É especialista no *software* MATLAB.

Maria Teresinha Arns Steiner possui licenciatura em Matemática pela Universidade Federal do Paraná (UFPR; 1978); graduação em Engenharia Civil pela UFPR (1981); mestrado em Engenharia de Produção pela UFSC (1988); doutorado em Engenharia de Produção pela UFSC (1995); pós-doutorado pelo ITA (2005) e pós-doutorado pelo IST de Lisboa (2014). Atuou na UFPR de agosto de 1978 a outubro de 2010 na UFPR. Após 32 anos de UFPR, se aposentou das atividades de graduação, mas continua a exercer as suas atividades de docência e de pesquisa no Programa de Pós-Graduação em Engenharia de Produção (PPGEP-UFPR) como professora sênior. Desde fevereiro de 2011 vem atuando na PUCPR, no curso de graduação em Engenharia de Produção e no Programa de Pós-Graduação em Engenharia de Produção e Sistemas (PPGEPS). Tem experiência na área de Engenharia de Produção, subárea de Pesquisa Operacional, com ênfase em *KDD* e Problemas de Reconhecimento de Padrões (Análise de Crédito Bancário, Engenharia de Avaliações, Diagnóstico Médico, dentre outros) e em Problemas de Otimização Combinatória, com destaque aos Problemas de Roteamento de Veículos e de Localização de Facilidades. Utiliza Procedimentos Exatos, Heurísticos e Meta-heurísticos.