Otimização via enxame de partículas para treinamento de redes neurais artificiais do tipo ELM: Um estudo de caso para predição de séries temporais

Vinicius Ferraço Arruda
Universidade Federal do Espírito Santo - UFES
Departamento de Informática
CEP 29060-270, Vitória, ES
viniciusferracoarruda@gmail.com

Renato A. Krohling
Universidade Federal do Espírito Santo - UFES
Programa de Pós-graduação em Informática - PPGI
Departamento de Engenharia de Produção
CEP 29060-270, Vitória, ES
krohling.renato@gmail.com

RESUMO

O uso de Redes Neurais Artificiais (RNA) para o problema de predição de séries temporais tem mostrado resultados promissores. No entanto, métodos de treinamento padrão de RNAs baseados em gradiente como o algoritmo *backpropagation* pode cair em mínimos locais e a aprendizagem da rede fica estagnada. Além disso, o treinamento de RNA com *backpropagation* pode ser muito demorado. A fim de resolver estas questões, o algoritmo ELM (abreviado do inglês: *Extreme Learning Machine*) foi desenvolvido. Uma vez que o algoritmo ELM requer a aprendizagem dos parâmetros da rede, apresenta-se o algoritmo PSO (abreviado do inglês: *Particle Swarm Optimization*) usado em conjunto com o algoritmo ELM, a fim de encontrar os parâmetros ótimos para a rede. Neste artigo, o algoritmo híbrido ELM+PSO é aplicado a problemas de predição de séries temporais univariada e multivariada.

PALAVRAS CHAVE. Redes neurais artificiais. Extreme Learning Machine. Otimização via enxame de partículas. Predição de séries temporais.

Tópicos: Metaheurística.

ABSTRACT

The use of Artificial Neural Networks (ANN) for the time series prediction problem has shown promising results. However, standard training methods of ANNs based on gradient as the backpropagation algorithm may get trap into local minima and the learning of the network stagnates. Moreover, the training of ANN with backpropagation may be very time consuming. In order to tackle these issues, the Extreme Learning Machine (ELM) algorithm was developed. Since the ELM algorithm requires the learning of parameters of the network, we present the Particle Swarm Optimization (PSO) algorithm used in conjunction with the ELM algorithm in order to find optimal parameters for the network. In this paper, the hybrid algorithm ELM+PSO is applied to univariate and multivariatand time series prediction problems.

KEYWORDS. Artificial neural networks. Extreme Learning Machine. Particle swarm optimization. Time series prediction.

Paper topics: Metaheuristic.

1. Introdução

A predição de séries temporais [Box et al., 2015] tem sido amplamente estudada ao longo dos últimos anos e vários modelos foram desenvolvidos. Dentre eles, as Redes Neurais Artificiais (abreviado por RNA) ganharam destaque por serem estimadores não-lineares poderosos. Contudo, o algoritmo para treinamento de RNAs conhecido como *backpropagation* [Rumelhart et al., 1988], possui algumas desvantagens, como a convergência prematura em mínimos locais e alto tempo de execução. Para minimizar esses problemas, foi desenvolvido um método de aprendizado de máquina que não se baseia no gradiente do erro médio como o *backpropagation*, chamado Máquina de Aprendizado Extremo (do inglês: *Extreme Learning Machine*, abreviado por ELM) [Huang et al., 2004], [Huang et al., 2006].

O algoritmo ELM apresenta rápida convergência em relação ao *backpropagation*, porém a determinação aleatória dos pesos de entrada e vieses (do inglês: *bias*) dos neurônios da camada oculta pode levar a um desempenho não ótimo [Huang et al., 2004], [Huang et al., 2006]. Visando encontrar pesos de entrada e vieses dos neurônios da camada oculta ótimos que implicam em um melhor desempenho do ELM, tem sido utilizado em conjunto o algoritmo Otimização via Enxame de Partículas (em inglês: *Particle Swarm Optimization*, abreviado por PSO) [Kennedy e Eberhart, 1995], [Pacífico e Ludermir, 2012] com bons resultados.

Neste trabalho, um híbrido entre os algoritmos ELM e PSO, seguindo uma abordagem semelhante ao IPSO-ELM [Han et al., 2013], será aplicado para problemas de séries temporais. Este trabalho está organizado da seguinte forma: na seção 2 será descrito como o problema de séries temporais será tratado para o algoritmo híbrido e uma breve descrição dos algoritmos ELM e PSO. Na seção 3 será apresentado o algoritmo híbrido. Na seção 4 será apresentado os resultados experimentais. Conclusões e direções para trabalhos futuros finalizam o artigo na seção 5.

2. Conceitos básicos

2.1. Rede Neurais Artificiais aplicado à predição de séries temporais

O problema de predição de séries temporais utilizando RNAs é considerado como a obtenção da relação do valor da série temporal em um determinado instante de tempo com μ valores anteriores. Também é possível obter a relação entre m valores futuros com μ valores anteriores, ou seja, prevendo não um mas m valores. Como mostrado por Donate et al. [2012], mais de um valor a ser predito pode ter informações importantes faltando para sua predição, sendo a melhor escolha ter apenas um único valor de saída da rede neural.

Os valores da série temporal podem ser amostrados em intervalos de τ, sendo interessante para algumas aplicações como séries temporais financeira, onde os dados são valores diários de segunda à sexta, em que se possa analisar esta série de segunda em segunda, terça em terça, e assim sucessivamente. Para isto, devemos inserir na rede neural, μ valores amostrados de cinco em cinco, logo, instancia-se a variável τ com o valor cinco. A equação (1) mostra a relação a ser obtida pela rede neural.

$$S_t = f(S_{t-1\tau}, S_{t-2\tau}, \ldots, S_{t-\mu\tau}) \tag{1}$$

S_1		S_1	$S_{1+\tau}$	$\cdots$	$S_{1+(\mu-1)\tau}$	$S_{1+\mu\tau}$
S_2		S_2	$S_{2+\tau}$	$\cdots$	$S_{2+(\mu-1)\tau}$	$S_{2+\mu\tau}$
S_3	$\rightarrow$	S_3	$S_{3+\tau}$	$\cdots$	$S_{3+(\mu-1)\tau}$	$S_{3+\mu\tau}$
$\vdots$		$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
S_t		$S_{t-\mu\tau}$	$S_{t-(\mu-1)\tau}$	$\cdots$	$S_{t-\tau}$	S_t

Figura 1: Processo de preparação dos dados da série temporal.

A Figura 1 mostra o processo inteiro para a preparação da série temporal para a rede neural, onde os dados da série temporal são processados em μ colunas que representam as entradas

da rede e uma coluna que são os valores esperados de saída da rede. Estes valores processados são divididos em fatias de 60%, 20% e 20%, em ordem crescente de t, compondo respectivamente as bases de treinamento, validação e teste.

Para as séries temporais multivariada a abordagem se mantém a mesma. Utilizando o processo mostrado na Figura 1, os dados apresentados no início do processo são os valores de uma única variável da série e em seguida obtido os dados processados para esta variável, como se fosse uma série temporal de única variável. Este processo é repetido para todas as variáveis da série temporal e em seguida os dados são arranjados seguindo a equação (2) de maneira semelhante a equação (1).

$$S_{x_t} = f(S_{1_{t-1\tau}}, \ldots, S_{1_{t-\mu\tau}}, S_{2_{t-1\tau}}, \ldots, S_{2_{t-\mu\tau}}, \ldots, S_{k_{t-1\tau}}, \ldots, S_{k_{t-\mu\tau}}) \tag{2}$$

sendo k o número de variáveis da série e S_{x_t} o valor da série no tempo t para a variável x no intervalo [1, k], ou seja, é utilizado todas as variáveis da série temporal para prever o valor de uma determinada variável.

2.2. Máquina de Aprendizado Extremo

O algoritmo ELM foi desenvolvido para redes SLFN (abreviação do inglês: *Single-layer feedforward networks*, ou redes de alimentação direta de camada única) e tem se mostrado bastante promissor [Huang et al., 2006]. Os pesos dos neurônios da camada de entrada e vieses (do inglês: *bias*) são inicializados com valores aleatórios, e os pesos dos neurônios da camada de saída são calculados analiticamente, sem utilizar processos iterativos como no *backpropagation*.

A formulação matemática do ELM, apresentada em [Huang et al., 2004] e [Huang et al., 2006], é descrita por:

$$\sum_{i=1}^{\widetilde{N}} \boldsymbol{\beta}_i g(\boldsymbol{w}_i \cdot \boldsymbol{x}_j + b_i) = \boldsymbol{t}_j, \quad j = 1, \ldots, N \tag{3}$$

onde:

- $\widetilde{N}$ é o número de neurônios na camada oculta e N é o número de amostras de treinamento.
- $\boldsymbol{x}_j = [x_{j1}, x_{j2}, \ldots, x_{jn}]^\mathsf{T}$ representa cada amostra distinta.
- $\boldsymbol{t}_j = [t_{j1}, t_{j2}, \ldots, t_{jm}]^\mathsf{T}$ são as saídas esperadas pela rede, referente à amostra de entrada x_j.
- $\boldsymbol{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^\mathsf{T}$ representa o vetor de pesos que conecta o *i*-ésimo neurônio da camada oculta aos neurônios da camada de entrada.
- $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^\mathsf{T}$ representa o vetor de pesos que conecta o *i*-ésimo neurônio da camada oculta aos neurônios da camada de saída.
- b_i representa o viés associado ao *i*-ésimo neurônio da camada oculta.
- $g(\cdot)$ é a função de ativação da rede.

As N equações representadas por (3) podem ser escritas na forma simplificada $\boldsymbol{H}\hat{\boldsymbol{\beta}} = \boldsymbol{T}$, cuja forma matricial é:

$$\boldsymbol{H} = \begin{bmatrix} g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_1 + b_1) & \cdots & g(\boldsymbol{w}_{\widetilde{N}} \cdot \boldsymbol{x}_1 + b_{\widetilde{N}}) \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_N + b_1) & \cdots & g(\boldsymbol{w}_{\widetilde{N}} \cdot \boldsymbol{x}_N + b_{\widetilde{N}}) \end{bmatrix}_{N \times \widetilde{N}} \quad \hat{\boldsymbol{\beta}} = \begin{bmatrix} \boldsymbol{\beta}_1^\mathsf{T} \\ \vdots \\ \boldsymbol{\beta}_{\widetilde{N}}^\mathsf{T} \end{bmatrix}_{\widetilde{N} \times m} \quad \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^\mathsf{T} \\ \vdots \\ \boldsymbol{t}_N^\mathsf{T} \end{bmatrix}_{N \times m}$$

A determinação dos pesos de saída, que ligam os neurônios da camada oculta à camada de saída, é definida como a solução dos mínimos quadrados (do inglês: *Least-Squares*) do sistema linear $\boldsymbol{H}\hat{\boldsymbol{\beta}} = \boldsymbol{T}$, que é dada por $\hat{\boldsymbol{\beta}} = \boldsymbol{H}^\dagger \boldsymbol{T}$, onde $\boldsymbol{H}^\dagger$ é a inversa generalizada de *Moore-Penrose* da matriz $\boldsymbol{H}$ [Huang et al., 2004], [Huang et al., 2006].

2.3. Otimização por Enxame de Partículas

O algoritmo PSO é uma meta-heurística de propósito geral que pode ser aplicado em diversos tipos de problemas de otimização. É um algoritmo baseado em população, desenvolvido por Kennedy e Eberhart [1995] que foi inspirado pelo comportamento social dos bandos de pássaros.

Em seu formato simples, a cada passo de tempo, cada partícula computa uma nova solução através da função de aptidão (do inglês: *fitness function*) partindo de uma nova posição gerada como a combinação da melhor posição pessoal visitada por si mesma, chamada $\boldsymbol{p}_{best}$, e um fator social, com algumas perturbações aleatórias. Dependendo da topologia utilizada, o fator social pode representar a melhor posição encontrada por todas as partículas do enxame, chamada $\boldsymbol{g}_{best}$ (topologia global), ou a melhor posição encontrada entre as partículas adjacentes à si mesma e ela própria, chamada $\boldsymbol{l}_{best}$ (topologia local).

Neste trabalho foi utilizado a topologia local e além disso, um fator de constrição também foi utilizado pois, de acordo com Eberhart e Shi [2001], pode ser necessário para garantir a convergência do algoritmo. As velocidades e posições são atualizadas, respectivamente, pelas equações (4) e (5).

$$\boldsymbol{v}_i(t+1) = \lambda \cdot \left[\boldsymbol{v}_i(t) + c_1 \cdot r_1 \cdot \left(\boldsymbol{p}_{best_i} - \boldsymbol{x}_i(t)\right) + c_2 \cdot r_2 \cdot \left(\boldsymbol{l}_{best_i} - \boldsymbol{x}_i(t)\right)\right] \tag{4}$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t) \tag{5}$$

$$\text{com} \qquad \lambda = \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|} \qquad \text{e} \qquad \phi = c_1 + c_2, \quad \phi > 4$$

onde:

- $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^\mathsf{T}$ é a posição da i-ésima partícula no espaço de busca d-dimensional.
- $\boldsymbol{v}_i = [v_{i1}, v_{i2}, \ldots, v_{id}]^\mathsf{T}$ é a velocidade da i-ésima partícula no espaço de busca d-dimensional.
- $\boldsymbol{p}_{best_i} = [p_{best_{i1}}, p_{best_{i2}}, \ldots, p_{best_{id}}]^\mathsf{T}$ é a melhor posição encontrada pela i-ésima partícula.
- $\boldsymbol{l}_{best_i} = [l_{best_{i1}}, l_{best_{i2}}, \ldots, l_{best_{id}}]^\mathsf{T}$ é a melhor posição encontrada entre as partículas adjacentes da i-ésima partícula e $\boldsymbol{p}_{best_i}$.
- λ é o fator de constrição.
- c_1 e c_2 são constantes positivas.
- r_1 e r_2 são números aleatórios gerados utilizando a distribuição de probabilidade uniforme no intervalo [0, 1]. Novos números são gerados a cada cálculo da equação (4).

Os valores para c_1 e c_2 foram utilizados baseado na literatura [Kennedy e Eberhart, 1995], sendo c_1 e c_2 configurados para 2.05 e o fator de constrição configurado para 0.729. Uma análise detalhada a respeito da derivação do fator de constrição pode ser encontrada no trabalho de Clerc e Kennedy [2002].

3. Algoritmo híbrido

O algoritmo híbrido ELM+PSO, baseado no algoritmo IPSO-ELM (abreviação do inglês: *Improved Particle Swarm Optimization - Extreme Learning Machine*) [Han et al., 2013], é a combinação entre os algoritmos PSO e ELM, onde o PSO adiciona ao ELM a função de encontrar um melhor conjunto de pesos, que ligam a camada de entrada à camada oculta, e vieses da camada oculta, ao invés da escolha aleatória. A definição utilizada de melhor conjunto de pesos e vieses é a mesma apresentada por Han et al. [2013], onde o conjunto de pesos e vieses tendem a encontrar um modelo do ELM de menor erro e maior poder de generalização. A diferença entre o IPSO-ELM e o híbrido deste trabalho está no uso da topologia local ao invés da global e no uso da variação PSO com fator de constrição ao invés do APSO (abreviação do inglês: *Adaptive particle swarm optimization*).

Cada partícula do ELM+PSO é um conjunto de pesos e vieses necessários para inicializar a rede neural, onde cada peso e viés possui sua própria velocidade e posição e, seguindo a notação apresentada na seção 2.2, são representadas na seguinte forma matricial:

$$\boldsymbol{P}_i = \begin{bmatrix} \boldsymbol{w}_1^\top & b_1 \\ \boldsymbol{w}_2^\top & b_2 \\ \vdots & \vdots \\ \boldsymbol{w}_{\widetilde{N}}^\top & b_{\widetilde{N}} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} & b_1 \\ w_{21} & w_{22} & \cdots & w_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{\widetilde{N}1} & w_{\widetilde{N}2} & \cdots & w_{\widetilde{N}n} & b_{\widetilde{N}} \end{bmatrix}_{\widetilde{N}\times(n+1)} \tag{6}$$

A velocidade e posição de cada partícula é calculada, respectivamente, pelas equações (4) e (5), onde os limites do espaço de busca e da velocidade é dado no intervalo [-1, 1] e caso a posição da partícula ultrapasse os limites do espaço de busca, a melhor posição já encontrada se torna a atual e caso a velocidade da partícula ultrapasse os limites de velocidade, o limite excedido se torna a velocidade atual.

A seleção da partícula $\boldsymbol{p}_{best_i}$ é dada por (7) e a seleção da partícula $\boldsymbol{l}_{best_i}$ é dada por (8) e (9).

$$\boldsymbol{p}_{best_i} = \begin{cases} \boldsymbol{P}_i, & (f(\boldsymbol{p}_{best_i}) - f(\boldsymbol{P}_i) > \eta f(\boldsymbol{p}_{best_i})) \quad \text{ou} \\ & (|f(\boldsymbol{p}_{best_i}) - f(\boldsymbol{P}_i)| < \eta f(\boldsymbol{p}_{best_i}) \quad \text{e} \quad ||\hat{\boldsymbol{\beta}}_{\boldsymbol{P}_i}|| < ||\hat{\boldsymbol{\beta}}_{\boldsymbol{p}_{best_i}}||) \\ \boldsymbol{p}_{best_i}, & \text{caso contrário} \end{cases} \tag{7}$$

$$\hat{\boldsymbol{l}}_{best_i} = \begin{cases} \boldsymbol{p}_{best_{i-1}}, & (f(\boldsymbol{p}_{best_i}) - f(\boldsymbol{p}_{best_{i-1}}) > \eta f(\boldsymbol{p}_{best_i}) \quad \text{ou} \\ & (|f(\boldsymbol{p}_{best_i}) - f(\boldsymbol{p}_{best_{i-1}})| < \eta f(\boldsymbol{p}_{best_i}) \quad \text{e} \quad ||\hat{\boldsymbol{\beta}}_{\boldsymbol{p}_{best_{i-1}}}|| < ||\hat{\boldsymbol{\beta}}_{\boldsymbol{p}_{best_i}}||) \\ \boldsymbol{p}_{best_i}, & \text{caso contrário} \end{cases} \tag{8}$$

$$\boldsymbol{l}_{best_i} = \begin{cases} \boldsymbol{p}_{best_{i+1}}, & (f(\hat{\boldsymbol{l}}_{best_i}) - f(\boldsymbol{p}_{best_{i+1}}) > \eta f(\hat{\boldsymbol{l}}_{best_i})) \quad \text{ou} \\ & (|f(\hat{\boldsymbol{l}}_{best_i}) - f(\boldsymbol{p}_{best_{i+1}})| < \eta f(\hat{\boldsymbol{l}}_{best_i}) \quad \text{e} \quad ||\hat{\boldsymbol{\beta}}_{\boldsymbol{p}_{best_{i+1}}}|| < ||\hat{\boldsymbol{\beta}}_{\hat{\boldsymbol{l}}_{best_i}}||) \\ \hat{\boldsymbol{l}}_{best_i}, & \text{caso contrário} \end{cases} \tag{9}$$

onde $\eta > 0$ é a taxa de tolerância [Han et al., 2013], $f(\cdot)$ é a função de aptidão, $|\cdot|$ é o valor absoluto, $||\cdot||$ é a norma ℓ^2 e i é o índice das partículas. As partículas $i-1$ e $i+1$ são respectivamente as partículas anterior e posterior à i. No caso de $i = 1$, então $i - 1 = K$, e no caso $i = K$, então $i + 1 = 1$, onde K é o número de partículas utilizadas no algoritmo.

De acordo com Han et al. [2013], quanto menor a norma dos pesos de saída da rede, um menor número condição da matriz $\boldsymbol{H}$ tende a ser encontrado, o que implica em um maior poder de generalização do modelo. O número condição é dado por:

$$cond(\boldsymbol{H}) = \sqrt{\frac{\lambda_{max}(\boldsymbol{H}^\top \boldsymbol{H})}{\lambda_{min}(\boldsymbol{H}^\top \boldsymbol{H})}} \tag{10}$$

onde $\lambda_{max}(\boldsymbol{H}^\top \boldsymbol{H})$ e $\lambda_{min}(\boldsymbol{H}^\top \boldsymbol{H})$ são o maior o e menor autovalor da matriz $\boldsymbol{H}^\top \boldsymbol{H}$ respectivamente.

Neste trabalho, a taxa de tolerância η foi configurada para 0.04, pois no trabalho de Han et al. [2013], este valor demonstrou em média os melhores resultados que foi encontrado através de um estudo de sensibilidade. A seguir é mostrado os algoritmos para o cálculo dos pesos de saída $\hat{\boldsymbol{\beta}}$ e da função de aptidão para uma determinada partícula $f(\boldsymbol{P})$:

Cálculo da matriz de pesos $\hat{\boldsymbol{\beta}}_{\boldsymbol{P}_i}$: Dado um conjunto de treinamento $\aleph = \{(\boldsymbol{x}_j, \boldsymbol{t}_j) | \boldsymbol{x}_j \in R^n, \boldsymbol{t}_j \in R^m, j = 1, \ldots, N\}$, função de ativação $g(\cdot)$, número de neurônios da camada oculta $\widetilde{N}$ e partícula $\boldsymbol{P}_i$:

Passo 1: Extrair os pesos $\boldsymbol{w}_i$ e vieses b_i da partícula $\boldsymbol{P}_i$.
Passo 2: Calcular a matriz de saída $\boldsymbol{H}$ da camada oculta.
Passo 3: Calcular os pesos de saída $\hat{\boldsymbol{\beta}}$ dado por $\hat{\boldsymbol{\beta}} = \boldsymbol{H}^{\dagger}\boldsymbol{T}$.

Cálculo da função de aptidão $f(\boldsymbol{P}_i)$: Dado um conjunto de validação $\aleph = \{(\boldsymbol{x}_j, \boldsymbol{t}_j) | \boldsymbol{x}_j \in R^n, \boldsymbol{t}_j \in R^m, j = 1, \ldots, N\}$, função de ativação $g(\cdot)$, número de neurônios da camada oculta $\widetilde{N}$, partícula $\boldsymbol{P}_i$ e pesos da camada de saída $\hat{\boldsymbol{\beta}}_{\boldsymbol{P}_i}$:

Passo 1: Extrair os pesos $\boldsymbol{w}_i$ e vieses b_i da partícula $\boldsymbol{P}_i$.
Passo 2: Calcular a matriz de saída $\boldsymbol{H}$ da camada oculta.
Passo 3: Utilizando os pesos de saída $\hat{\boldsymbol{\beta}}_{\boldsymbol{P}_i}$, calcular a saída da rede $\boldsymbol{S}$ dado pelo produto matricial $\boldsymbol{H}\hat{\boldsymbol{\beta}}$, onde $\boldsymbol{S} = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N]^{\mathsf{T}}$ e $\boldsymbol{s}_j = [s_{j1}, s_{j2}, \ldots, s_{jm}]^{\mathsf{T}}$.
Passo 4: Calcular a aptidão da partícula dada pela métrica da raiz do erro quadrático médio (do inglês: *root of the mean squared error*, abreviado por RMSE), dada por:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{t}_i - \boldsymbol{s}_i)^2} \tag{11}$$

Como mostrado na seção 2.1, problemas de predição de séries temporais tendem a uma melhor solução com saída de dimensão 1, portanto $m = 1$.

4. Resultados Experimentais

Para efeito de comparação foi executado o algoritmo ELM junto ao ELM+PSO utilizando a métrica da raiz do erro quadrático médio (RMSE), definido pela equação (11), e calculado estatísticas para mostrar os resultados obtidos.

4.1. Configurações

Os algoritmos ELM e ELM+PSO foram executados 30 vezes para cada série temporal. Todos os experimentos foram executados no ambiente MATLAB 8.2.0.701 (R2013b) em uma máquina com 4 GB de memória RAM, processador AMD Phenom II X2 B53 2.8 GHz, sistema operacional Windows 7. Tanto para o ELM quanto para o ELM+PSO, os pesos de entrada e os vieses foram obtidos no intervalo [-1, 1], e como função de ativação foi utilizado a sigmóide: $g(x) = \frac{1}{1+e^{-x}}$. Para o algoritmo híbrido, o espaço de busca e velocidade das partículas foi estabelecido no intervalo [-1, 1]. O número de partículas utilizado foi 20 e o número de iterações 50. Os dados das séries temporais aqui utilizadas foram normalizados entre [0, 1] e uma vez obtido os resultados o processo inverso é realizado, redimensionando-os de volta a escala original.

4.2. Séries temporais univariada

Para os resultados, foi utilizado cinco séries temporais univariada. A tabela 1 mostra as séries temporais utilizadas com o número de amostras que cada uma possui e os valores encontrados em um estudo de sensitividade simples para μ, τ e número de neurônios. As Figuras 2 e 3 mostram os gráficos das séries temporais univariadas utilizadas e uma breve descrição de cada uma é mostrada a seguir:

Darwin SLP: Valores mensais da pressão atmosférica no nível do mar da cidade de Darwin, Austrália, no período 1882 - 1998.
Energia da Austrália: Produção de eletricidade mensal na Austrália a partir de janeiro de 1956 até agosto de 1995. Os dados desta série são apresentados em Gigawatt-hora (GWh).
Tráfego no túnel: Número de carros que passam por um determinado túnel diariamente, coletados no intervalo entre os dias 1 de novembro de 2003 e 16 de novembro de 2005.

Mackey-Glass: Mackey-Glass é uma equação diferencial não linear, conhecida por ser utilizada em testes de desempenho de preditores de séries temporais [Singh e Balasundaram, 2007], descrita a seguir:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x(t-\tau)^{10}} - bx(t)$$

onde $a = 0.2$, $b = 0.1$, a condição inicial $x(0) = 1.2$. Em particular, para a série temporal Mackey-Glass 17 e Mackey-Glass 30, $\tau = 17$ e $\tau = 30$, respectivamente. Foram geradas 6000 amostras e seguindo a estratégia abordada no trabalho de Singh e Balasundaram [2007], as 3500 primeiras amostras foram descartadas considerando apenas as 2500 últimas amostras.

Série temporal	Número de amostras	μ	τ	Número de neurônios
Darwin SLP	1400	10	1	10
Energia da Austrália	476	5	3	10
Trânsito no Túnel	747	10	1	30
Mackey-Glass 17	2500	5	1	10
Mackey-Glass 30	2500	5	1	10

Tabela 1: Séries temporais univariada.

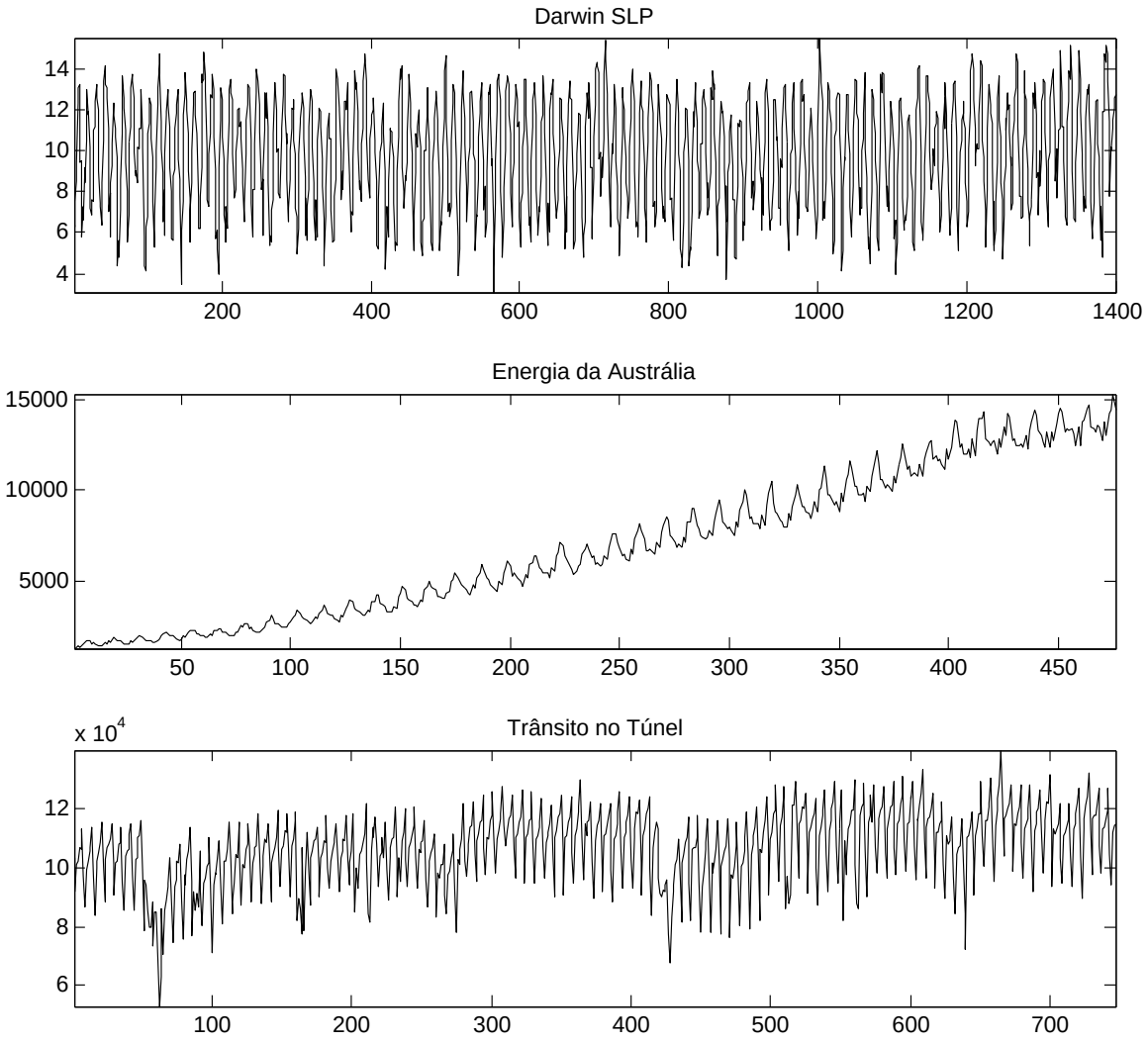

Figura 2: Gráfico das séries temporais Darwin SLP, Energia da Austrália e Trânsito no Túnel.

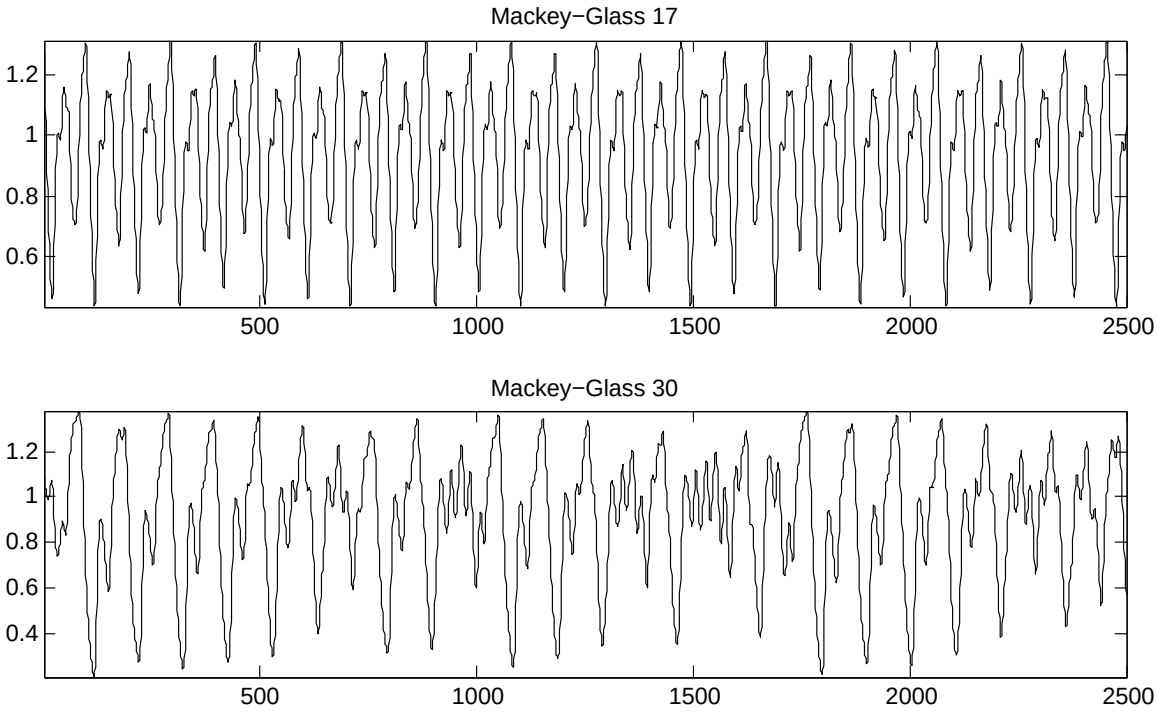

Figura 3: Gráfico das séries temporais Mackey-Glass 17 e 30.

4.3. Séries temporais multivariada

Uma única série temporal multivariada foi estudada neste trabalho. A série temporal são os índices do IBOVESPA dos dias úteis de 2 de janeiro de 2004 até 16 de fevereiro de 2016, com 4 variáveis sendo os índices de abertura, mais alto, mais baixo e de fechamento de um determinado dia, totalizando 3015 amostras para cada variável. As configurações utilizadas para μ, τ e número de neurônios é de 5, 1 e 30, respectivamente. A Figura 4 mostra o gráfico dos índices de abertura. Os demais gráficos seguem o mesmo padrão não tendo diferença visualmente.

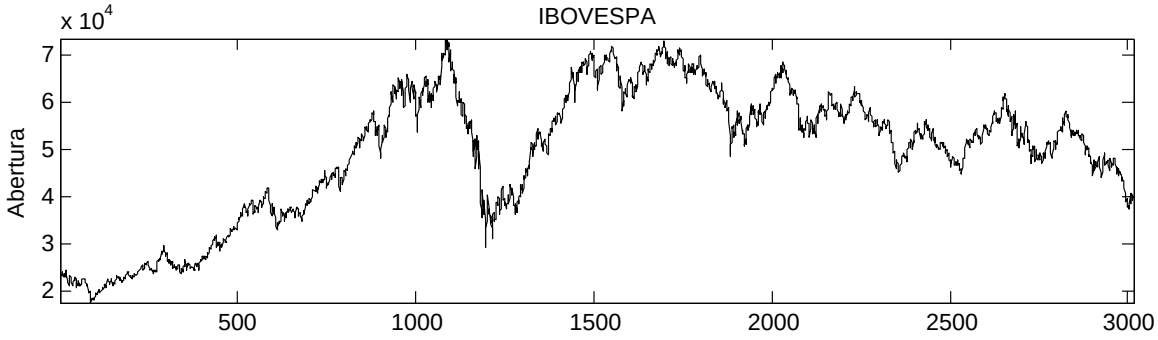

Figura 4: Gráfico dos índices de abertura da IBOVESPA.

4.4. Resultados

As Figuras 5 e 6 mostram o box-plot do RMSE das execuções dos algoritmos. As Tabelas 2 e 3 contém a média e o desvio padrão dos RMSEs da partição de teste, da norma dos pesos de saída e do número condição da matriz $\boldsymbol{H}$ da partição de treinamento.

O tempo de execução das séries temporais univariada também estão na Tabela 2. A média e desvio padrão do tempo de execução total da série multivariada para o ELM foi de 0.063 ± 0.032 segundos e para o híbrido 336.945 ± 0.952 segundos, pois como todas as variáveis possuem o mesmo número de amostras e a mesma configuração dos parâmetros, o número de instruções

executadas pelos algoritmos foi exatamente o mesmo para todas as variáveis, sendo assim, o tempo foi medido para a execução da série inteira com todas as suas variáveis. Para estimar o quanto cada variável consumiu de tempo de execução, é seguro dividir este valor pelo número de variáveis da série.

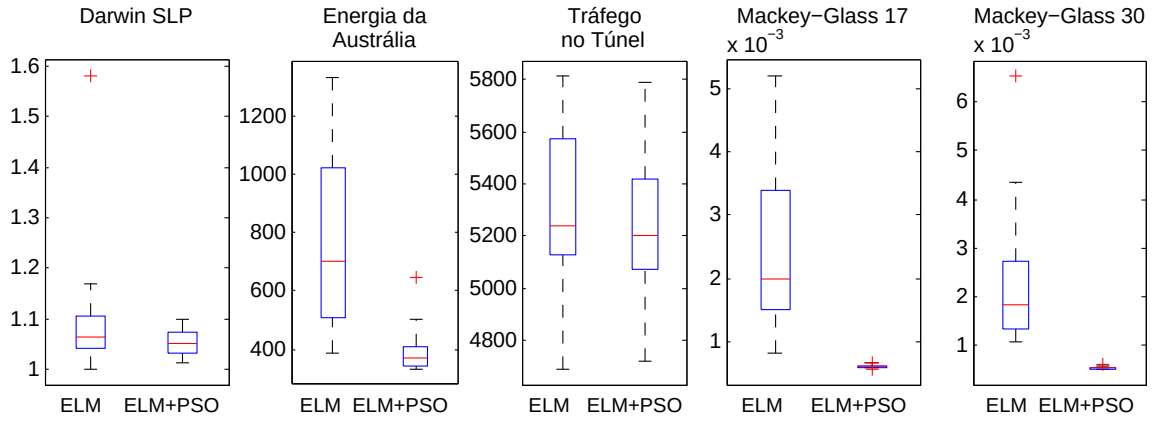

Figura 5: RMSE das séries temporais univariada.

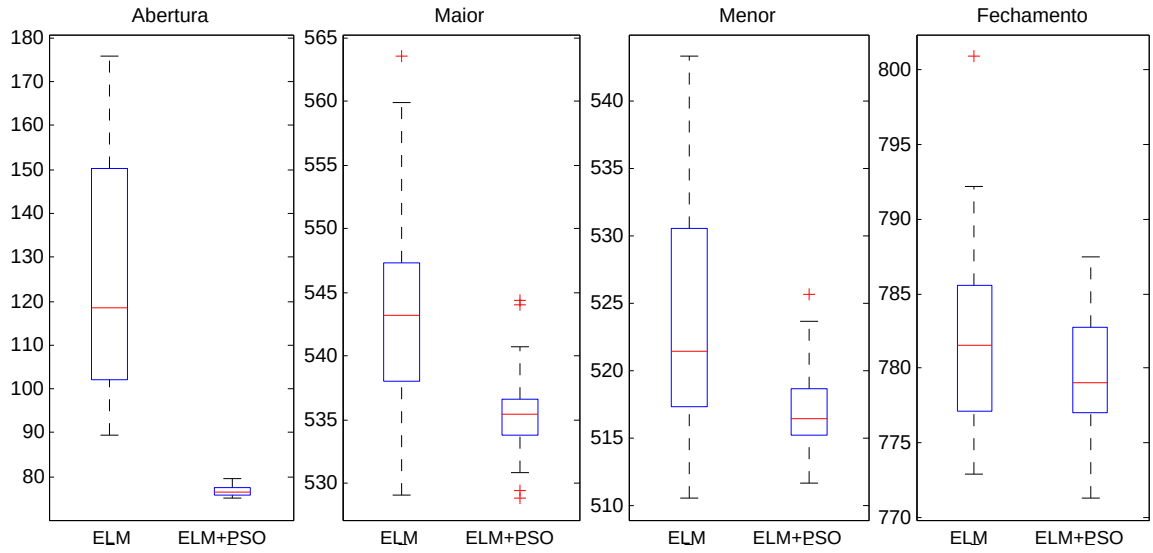

Figura 6: RMSE da série temporal multivariada.

Série	Algoritmo	RMSE	Número Condição	Norma $\hat{\beta}$	Tempo total
Darwin	ELM	1.087 ± 0.102	376.851 ± 174.611	3.982 ± 1.622	**0.002 ± 0.008**
SLP	ELM+PSO	**1.052 ± 0.024**	**259.364 ± 63.925**	**1.087 ± 0.074**	15.039 ± 0.093
Energia	ELM	762.478 ± 283.699	35287.253 ± 18882.552	19.967 ± 13.004	**0.001 ± 0.006**
Austrália	ELM+PSO	**390.084 ± 68.965**	**14944.185 ± 10160.303**	**3.357 ± 0.584**	8.109 ± 0.105
Tráfego	ELM	5315.780 ± 303.401	7064.807 ± 1823.896	30.461 ± **8.034**	**0.006 ± 0.013**
Túnel	ELM+PSO	**5238.543 ± 267.010**	**5231.644 ± 1803.028**	**15.479** ± 9.525	41.789 ± 0.274
Mackey	ELM	0.002 ± 0.001	**32906.749 ± 17786.785**	41.842 ± 29.812	**0.005 ± 0.012**
Glass 17	ELM+PSO	**0.001 ± 0.000**	54879.718 ± 30750.591	**30.405 ± 13.390**	10.563 ± 0.057
Mackey	ELM	0.002 ± 0.001	**46870.368 ± 32168.808**	35.323 ± 21.978	**0.006 ± 0.013**
Glass 30	ELM+PSO	**0.001 ± 0.000**	87448.483 ± 78186.189	**33.525 ± 14.188**	10.516 ± 0.051

Tabela 2: Resultados das séries temporais univariada.

Variável	Algoritmo	RMSE	Número Condição	Norma $\hat{\beta}$
Abertura	ELM	123.795 ± 25.690	**17172.997 ± 3386.637**	6.982 ± 2.417
	ELM+PSO	**76.736 ± 1.192**	23209.263 ± 7129.160	**2.618 ± 0.486**
Maior	ELM	543.657 ± 7.371	18253.787 ± 3991.045	6.603 ± 2.440
	ELM+PSO	**535.445 ± 3.650**	**14305.630 ± 2748.684**	**2.301 ± 0.284**
Menor	ELM	523.212 ± 7.774	17163.312 ± 3566.802	7.824 ± 2.366
	ELM+PSO	**517.030 ± 3.092**	**14387.142 ± 2775.236**	**2.688 ± 0.214**
Fechamento	ELM	781.712 ± 6.251	18929.785 ± 3841.137	6.942 ± 1.612
	ELM+PSO	**779.416 ± 3.943**	**12915.389 ± 2577.701**	**2.379 ± 0.237**

Tabela 3: Resultados da série temporal multivariada.

5. Conclusões

Os resultados indicaram em geral, em termos de RMSE, que o algoritmo híbrido obteve maior eficácia e robustez em relação ao ELM. Isto comprova a eficácia das equações (7), (8) e (9) descritas na seção 3, que obtiveram matrizes $\boldsymbol{H}$ de baixo número condição, indicando maior poder de generalização, implicando em RMSE com baixo desvio padrão e de menor média quando comparado ao ELM. O algoritmo híbrido falhou ao encontrar um número condição mais alto para as séries temporais Mackey-Glass 17 e 30 e para a variável Abertura da série temporal IBOVESPA, porém obteve sucesso ao encontrar um modelo de menor média e desvio padrão do erro.

Uma vantagem relevante encontrada durante a pesquisa, foi que o PSO permite que com poucos neurônios na camada escondida se encontre modelos eficazes. Durante o estudo elaborado para a escolha dos parâmetros, foi observado que baixo número de neurônios pioraram significativamente a qualidade do resultado para o algoritmo ELM padrão, porém o uso do PSO elevou a qualidade de maneira significativa. Uma conclusão importante sobre este fato é que, se memória for um fator crítico para o uso do algoritmo mas tempo computacional não, o uso de poucos neurônios utilizando o algoritmo híbrido basta para encontrar uma boa solução. A única desvantagem em utilizar o híbrido ao invés do ELM padrão é o aumento no tempo de execução devido ao processo iterativo do PSO.

As metas futuras para este trabalho é realizar um estudo aprofundado no problema de predição de séries temporais para redes neurais visando estabelecer uma relação entre os parâmetros μ e τ da série temporal e um número de neurônios mais adequado, visto que estes parâmetros são fatores importantes e sensíveis para os resultados.

Referências

Box, G. E., Jenkins, G. M., Reinsel, G. C., e Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Clerc, M. e Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73.

Donate, J. P., Sanchez, G. G., e de Miguel, A. S. (2012). Time series forecasting: A comparative study between an evolving artificial neural networks system and statistical methods. *International Journal on Artificial Intelligence Tools*, 21(1):1–26.

Eberhart, R. C. e Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. In *Proceedings of the IEEE Congress on Evolutionary Computation, 2001*, volume 1, p. 81–86.

Han, F., Yao, H.-F., e Ling, Q.-H. (2013). An improved evolutionary extreme learning machine based on particle swarm optimization. *Neurocomputing*, 116:87–93.

Huang, G.-B., Zhu, Q.-Y., e Siew, C.-K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. In *In Proceedings of the IEEE International Joint Conference on Neural Networks, 2004*, volume 2, p. 985–990.

Huang, G.-B., Zhu, Q.-Y., e Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501.

Kennedy, J. e Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, p. 1942–1948.

Pacífico, L. D. e Ludermir, T. B. (2012). Melhorando redes neurais do tipo extreme learning machine através da otimização por enxame de partículas com mecanismo de seleção. In *Proceedings of the Brazilian Conference on Intelligent Systems*, p. 1–11. BRACIS.

Rumelhart, D. E., Hinton, G. E., e Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Singh, R. e Balasundaram, S. (2007). Application of extreme learning machine method for time series analysis. *International Journal of Intelligent Technology*, 2(4):256–262.