CONEM 2016
CONGRESSO NACIONAL DE ENGENHARIA MECÂNICA
21 - 25 AGOSTO DE 2016
FORTALEZA - CEARÁ

MÁQUINA DE APRENDIZADO EXTREMO COMBINADO COM OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS APLICADO EM PREVISÃO DE SÉRIES TEMPORAIS

Rafael Bartnik Grebogi, rafagrebogi@gmail.com[1]
Deivid Zancheta de Lima, dd_lages@hotmail.com[2]
Eduardo Massashi Yamao, massashi_yamao@yahoo.com.br[3]
Marco Antonio Reichert Boaretto, marco.boaretto@hotmail.com[3]
Leandro dos Santos Coelho, leandro.coelho@pucpr.br[4]

[1]Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), Universidade Federal do Paraná (UFPR), Curitiba, PR, Brasil, e Departamento de Processos Industriais, Instituto Federal de Santa Catarina (IFSC), Lages, SC, Brasil
[2]Curso Técnico de Eletromecânica, Departamento de Processos Industriais, Instituto Federal de Santa Catarina (IFSC), Lages, SC, Brasil
[3]Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), Universidade Federal do Paraná (UFPR), Curitiba, PR, Brasil
[4]Programa de Pós-Graduação em Engenharia de Produção e Sistemas (PPGEPS), Escola Politécnica, Pontifícia Universidade Católica do Paraná (PUCPR) e Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), Universidade Federal do Paraná (UFPR), Curitiba, PR, Brasil

Resumo: *O objetivo deste artigo é otimizar a aplicação de máquinas de aprendizado extremo para previsão de séries temporais. O algoritmo de máquinas de aprendizado extremo, neste contexto, serve como uma alternativa às limitações relacionadas ao custo computacional e tempo de treinamento das redes neurais artificiais do tipo perceptron multi-camadas. É aplicada a técnica de otimização por enxame de partículas clássica junto com mais duas de suas variações como alternativa de método de otimização linear das ponderações do algoritmo de máquinas de aprendizado extremo. Para testar esta abordagem de otimização são comparadas aplicações em três estudos de caso de previsão de séries temporais. As previsões realizadas pela máquina de aprendizado extremo combinada com a otimização por enxame de partículas revelam resultados promissores, mostrando que a combinação das duas técnicas é uma eficiente forma de previsão de séries temporais ou mesmo identificação de sistemas não-lineares.*

Palavras-chave: *Máquina de aprendizado extremo, otimização,otimização por enxame de partículas, identificação de sistemas, previsão de séries temporais.*

1. INTRODUÇÃO

A Computação Natural (De Castro, 2006) é constituída por novas abordagens de computação caracterizadas por uma maior proximidade com a natureza. A computação natural pode ser vista como uma versão computacional do processo de "extração de ideias" da natureza para o desenvolvimento de sistemas "artificiais", ou então a utilização de meios e mecanismos naturais para realizar computação.

Os paradigmas da área da Computação Natural denominada Inteligência Computacional, em termos gerais, buscam conceber abordagens vinculadas ao desenvolvimento de sistemas inteligentes que imitem (simulem ou emulem) aspectos do comportamento humano, tais como: aprendizado, percepção, raciocínio, evolução e adaptação. Neste contexto, algoritmos evolutivos, inteligência de enxames, raciocínio probabilístico, sistemas nebulosos e Redes Neurais Artificiais (RNAs) se destacam como promissores paradigmas. O foco principal deste artigo envolve as RNAs.

Em termos de RNAs, a aprendizagem ou treinamento corresponde ao processo de ajuste dos parâmetros livres da rede através de um mecanismo de apresentação de estímulos ambientais, conhecidos como padrões (ou dados) de treinamento. A maneira pela qual o ambiente influencia a rede em seu aprendizado define o paradigma de aprendizagem. Existem três paradigmas principais: aprendizado supervisionado, aprendizado não-supervisionado (ou auto-organizado) e aprendizado por reforço (Haykin, 2008).

Como uma alternativa às limitações das abordagens de RNAs do tipo perceptron de múltiplas camadas (MLP, do inglês *Multilayer Perceptron*), particularmente referente ao custo computacional vinculado à fase de treinamento dos pesos sinápticos, particularmente quando este envolve o ajuste dos pesos dos neurônios da camada intermediária e à necessidade de especificação de múltiplos parâmetros, foram propostas as Máquinas de Aprendizado Extremo (ELMs, do inglês *Extreme Learning Machines*) (Huang *et al*., 2004).

As ELMs são uma arquitetura de RNAs com uma camada de entrada, uma camada intermediária e uma camada de saída. Sua arquitetura é equivalente à da MLP, mas os pesos e o número de neurônios da camada intermediária em ELMs são definidos arbitrariamente e a priori, enquanto os pesos da camada de saída são os únicos parâmetros ajustáveis durante o processo de treinamento supervisionado. O ajuste desses pesos da camada de saída leva a um problema de otimização linear, responsável pelo fato de o treinamento de uma ELM ser ao menos uma ordem de magnitude mais rápido do que o treinamento, geralmente usando método da retropropagação do erro (*error backpropagation*), de uma MLP (Kulaif, 2014).

Uma característica relevante da ELM é a forma como são calculadas as ponderações de saída da RNA, através do cálculo de uma inversa generalizada, o que transforma o problema num sistema linear (Huang *et al.,* 2004; Huang *et al.,* 2006).

Diversos autores criaram métodos com o objetivo de melhorar o desempenho do ELM, porá citar a abordagem apresentada em Silva *et al.* (2011), que explora o algoritmo de procura em grupo para otimizar as ponderações do algoritmo. Este trabalho busca mostrar como o algoritmo de otimização por enxame de partículas (em inglês, *Particle Swarm Optimization*, PSO), um paradigma da inteligência de enxames (*swarm intelligence*) pode ser capaz de melhorar o desempenho do ELM. Além de apresentar duas variações do algoritmo PSO clássico.

O restante deste artigo está organizado da seguinte forma. A seção 2 apresenta os fundamentos das RNAs e o método de aprendizagem ELM. A seção 3 traz as explicações sobre otimização de problemas e sobre o algoritmo PSO e suas variações. A seção 4 traz algumas definições sobre séries temporais e também apresenta os 3 estudos de caso escolhidos para validar tanto a RNAs e o ELM, quanto o algoritmo de otimização. A seção 5 expõe os resultados obtidos através das simulações, enquanto a seção 6 traz as conclusões deste trabalho.

2. FUNDAMENTOS DE REDES NEURAIS ARTIFICIAIS E MÁQUINA DE APRENDIZADO EXTREMO

O neurônio pode ser dividido em três partes (Haykin, 2008): a função de rede, que determina como as entradas *x* da rede serão ponderadas e somadas, a função de ativação, e a ponderação de saída ligando a camada de neurônios até a camada de saída, sendo o modelo final representado pela Eq. (1) tal que

$$\sum_{i=1}^{\tilde{N}} \beta_i g\left(w_j \cdot x_j + b_i\right) = y_j \,, \quad j = 1, \ldots, N \,, \tag{1}$$

onde β_i é a ponderação de saída do i-ésimo neurônio, $\tilde{N}$ é o número de neurônios, $w_i \cdot x_j$ denota o produto interno entre w_i e x_j, y é o sinal de saída do neurônio, $g(\cdot)$ é a função de ativação, b_i é a polarização, em inglês, *bias*, e x é o sinal de entrada.

Conforme Huang (2004), o principal requerimento de uma RNA para aproximar uma função $f(\cdot)$, é que a função $F(\cdot)$, que descreverá as entradas-saídas mapeadas, esteja perto o suficiente, num ponto de vista Euclidiano em todas as amostras, sendo que a relação é descrita pela Eq. (2). Esta função de aproximação é realizada pela rede neural tal que

$$\|F(x) - f(x)\| < \varepsilon \quad \text{para todo } \boldsymbol{x}, \tag{2}$$

onde ε é o erro de aproximação, que é um número positivo e pequeno. Desde que o número de amostras *N* seja grande o suficiente e a RNA esteja equipada com parâmetros livres suficientes, o erro de aproximação pode ser suficientemente pequeno para a tarefa (Huang *et al.*, 2004).

A partir da diferença entre a saída real da função desconhecida e saída estimada pela rede, se obtém o vetor de sinal de erro e_i, sendo este vetor utilizado para ajustar os parâmetros livres da RNA, a fim de minimizar a diferença entre a saída real e a estimada. Para realizar o ajuste dos parâmetros livres existem diversos métodos, dentre os quais podem ser destacados a Aprendizagem *Online* utilizando retropropagação e o ELM, o qual será abordado neste trabalho. Maiores detalhes sobre o método de Retropropagação podem ser encontrados em (Haykin, 2008; Huang *et al.,* 2004).

2.1. Máquina de Aprendizado Extremo

O algoritmo ELM proposto por Huang *et al.* (2004), apresenta diversas vantagens em relação a outros algoritmos. O algoritmo também tende a alcançar não somente o menor erro de treinamento, mas também a menor norma das ponderações, Silva *et al.* (2011).

Uma das características provadas pelos autores é que as ponderações de entrada e a polarização (*bias*) da camada oculta de uma rede composta por apenas uma camada de neurônios ocultos, chamadas de *SLFN*, do inglês, *Single Layer Feedforward Neural Networks*, podem ser assinalados aleatoriamente se a função de ativação for infinitamente diferenciável. Assim, a SLFN pode ser considerada um simples sistema linear e as ponderações de saída podem ser

analiticamente calculadas através de uma operação de inversa generalizada de Moore-Penrose. O algoritmo ELM faz uso da matriz inversa para uma solução de mínimos quadrados com norma mínima para o cálculo das ponderações de saída β. Fazendo uso do modelo de neurônio descrito pela Eq. (1), pode se escrever as N equações de forma compacta como,

$$H\beta = Y_R \ , \tag{3}$$

onde Y_R é a saída real, e H é chamada de matriz de saída da camada oculta de neurônios, onde i-ésima coluna de H é o i-ésimo nó de saída oculto com respeito as entradas $x_1, x_2, \dots, x_N$ (Huang *et al.,* 2004; Silva *et al.,* 2011).

Se a função de ativação for infinitamente diferenciável, é possível provar que o número de neurônios necessários será $\tilde{N} \leq N$, para isso são formulados alguns teoremas sobre a SLFN em questão, os quais podem ser obtidos em Silva *et al.* (2011).

A partir do método de obtenção da inversa por uma solução de mínimos quadrados com norma mínima, citado anteriormente, tem-se a Eq. (4). Por conveniência, a matriz inversa generalizada de Moore-Penrose da matriz Hserá denotada por H^{ρ}, assim,

$$\hat{\beta} = H^{\rho} Y_R \ , \tag{4}$$

onde a solução especial $\hat{\beta}$ é uma das soluções de mínimos quadrados de um sistema linear geral dado pela Eq. (3), significando que o menor erro de treinamento pode ser alcançado por esta solução, como pode ser demonstrado a seguir, através da Eq. (5).

$$\left\| H\hat{\beta} - Y_R \right\| = \| HH^{\rho} Y_R - Y_R \| = \min_{\beta} \| H\beta - Y_R \| \ . \tag{5}$$

Como foi afirmado anteriormente, teoricamente, este algoritmo funciona para qualquer função de ativação infinitamente diferenciável, como por exemplo, a função sigmóide.

Neste trabalho serão empregados duas funções de ativação distintas, uma linear e outra não-linear. A função de ativação linear é dada por,

$$z_{lin}(u) = uk_{lin} \ , \tag{6}$$

onde k_{lin} é uma constante positiva.

A função de ativação não-linear ficará a cargo da função de ativação variável, que a partir deste momento será denominada TAF, do inglês *Tunable Activation Function*, e pode ser definida pela seguinte equação,

$$z_{TAF}(u) = \frac{\zeta}{1+e^{-\frac{u-\delta}{\mu}}} + \alpha \ , \tag{7}$$

onde ζ é o fator de mapeamento, por ele alterar o comprimento e o alcance do mapeamento, α é o fator de deslocamento vertical, δ é o parâmetro de posicionamento horizontal, e μ é o fator do ângulo de inclinação, ele determina o formato de $z_{TAF}(u)$. Uma situação especial da função TAF é quando $\alpha = \delta = 0$ e $\zeta = \mu = 1$, resultando na função sigmóide.

Esta função de ativação é sugerida em, Wang *et al.* (2012), como uma forma de corrigir uma falha da função sigmóide, que é a restrição no mapeamento não-linear da rede, diminuindo a precisão do treinamento.

3. OTIMIZAÇÃO

Os algoritmos de otimização são métodos de procura ou busca, que têm por objetivo obter soluções para problemas de otimização (Engelbrecht, 2005). Ainda de acordo com Engelbrecht (2005), estes problemas podem ou não serem sujeitos a uma, ou mais restrições lineares, ou não-lineares. As soluções para os problemas nem sempre são simples, sendo que elas podem consistir de diferentes tipos de dados, tais como variáveis inteiras ou de ponto flutuante.

De maneira geral, pode se definir um problema de otimização da seguinte forma,

$$\begin{aligned} &\min_{\vec{x}} \vec{f}(\vec{x}) \\ &\text{sujeito a} \ \ \vec{g}(\vec{x}) \geq 0 \ e \ \vec{h}(\vec{x}) = 0 \\ &\text{para} \ \ \vec{x}_{min} \leq \vec{x} \leq \vec{x}_{max} \ , \end{aligned} \tag{8}$$

onde, $\vec{f}(\vec{x})$ é a função a ser minimizada, $\vec{g}(\vec{x})$ e $\vec{h}(\vec{x})$ são restrições lineares e não-lineares, respectivamente, e $\vec{x}_{min}$ e $\vec{x}_{max}$ são os limites inferior e superior, respectivamente, do espaço de busca do algoritmo de otimização.

Os problemas também podem apresentar algumas dificuldades a mais para o algoritmo, como por exemplo, as características podem variar com o tempo, desqualificando soluções encontradas anteriormente e consideradas boas, ou, os objetivos podem ser conflitantes.

3.1. PSO

O algoritmo de otimização por enxame de partículas, *PSO*, foi proposto inicialmente por Kennedy e Eberhart em 1995, (Kennedy e Eberhart, 1995). O algoritmo se baseia no comportamento social de um bando de pássaros procurando por comida, onde cada pássaro seria uma partícula, e no algoritmo, esta partícula seria uma possível solução para o problema.

A partir de Engelbrecht (2005), a explicação do funcionamento do algoritmo pode ser simplificada ao imaginar as partículas "voando" pelo espaço de busca multidimensional. Durante este "voo", as partículas trocam experiências com seus vizinhos, a fim de que a nova posição da partícula seja ajustada de acordo com suas próprias experiências e a experiência de seus vizinhos.

A posição da partícula e a sua velocidade são dadas por Eq. (9) e Eq. (10), respectivamente,

$$x_i(t+1) = x_i(t) + v_i(t+1), \tag{9}$$

$$v_i(t+1) = w \cdot v_i(t) + r_1 c_1\big(lP_i(t) - x_i(t)\big) + r_2 c_2\big(gP(t) - x_i(t)\big), \tag{10}$$

onde x_i é a posição da partícula i, r_1 e r_2 são valores aleatórios gerados com distribuição uniforme no intervalo [0 1], v_i é a velocidade da partícula i, t é a iteração atual do algoritmo, w é a ponderação de inércia da partícula, lP_i (*local best*) é a melhor posição que a partícula alcançou até a iteração t e gP (*global best*) é a melhor posição alcançada entre todas as partículas do enxame até a iteração t.

Os coeficientes de aceleração c_1 e c_2 atuam em conjunto com os valores aleatórios r_1 e r_2 na influência estocástica dos componentes social e cognitivo. Valores de magnitude pequena para estes parâmetros garantem uma trajetória suave para a partícula, já valores altos causam maior aceleração e movimentos bruscos. Eberhart e Shi (1999), afirmam que experiências anteriores mostraram empiricamente que os melhores valores para os coeficientes de aceleração são ambos iguais a 2 para a maioria das aplicações, (Eberhart e Shi, 1999).

A ponderação ou, peso de inércia, é um parâmetro que influencia diretamente na convergência do algoritmo. Podendo ser descrito como um mecanismo de controle do momento da partícula, ponderando a contribuição da velocidade anterior, e com isso, alterando a habilidade do algoritmo em explorar o espaço de busca. O valor deste parâmetro é de grande importância para garantir um comportamento convergente ao algoritmo. Sendo que para valores maiores que "1", o algoritmo terá a tendência de divergir. Valores altos e menores que "1" priorizam uma busca global, aumentando a diversidade do enxame, enquanto valores mais próximos de "0" priorizam uma busca local, Engelbrecht (2005).

Uma modificação bem-sucedida do algoritmo PSO implementada por diversos autores é tornar o peso de inércia dinâmico, sendo que um modo de variar o parâmetro ω é iniciando-o com o valor de 0.9 e linearmente decaindo para 0.4, (Eberhart e Shi, 1999). Desta forma o algoritmo priorizará inicialmente uma busca global, eliminando possíveis mínimos locais, e posteriormente uma busca local, para alcançar o mínimo global.

3.2. Gama PSO

Esta distribuição ainda possui poucas aplicações em algoritmos de otimização, sendo que nenhuma publicação empregando a distribuição gama à inteligência de enxames.

Ela pode não ser normalmente empregada, como a distribuição normal, mas possui diversas aplicações práticas, como a modelagem da quantidade de chuva diária em uma região (Mello *et al.,* 1994), para ajustar conjuntos de dados hidrológicos (Krishnamoorthy *et al.,* 2008), e na modelagem da quantidade de pagamento de dividendos nas reivindicações dos acionistas de uma companhia de seguros, Karapetyan (2009).

A representação Eq. (11) obtida de Temme (1992) mostra a função gama incompleta, utilizada para a resolução de problemas de matemática estatística e teoria de probabilidade devido à propriedade definida na Eq. (12). Neste caso,

$$\Gamma_2(z_\Gamma, \alpha_\Gamma) = \frac{1}{\Gamma(z_\Gamma)} \int_\alpha^\infty t^{z-1} e^{-t} dt \ , \tag{11}$$

$$\Upsilon_2(z_\Gamma, \alpha_\Gamma) = \frac{1}{\Gamma(z_\Gamma)} \int_0^\alpha t^{z-1} e^{-t} dt \ ,$$

com

$$\Gamma_2 + \Upsilon_2 = 1 \, , \tag{12}$$

onde z_Γ deve ser maior que 0. Se $\alpha_\Gamma = 0$, então, $\Gamma(z_\Gamma, 0) = \Gamma(z_\Gamma)$ que é a função gama.

O cálculo da função gama incompleta para uma determinada sequência de valores pode gerar curvas com formatos interessantes aos algoritmos de otimização, mais especificamente, ao peso de inércia w do algoritmo PSO. A Fig. 1 mostra os valores da função gama incompleta Υ_2 dada pela equação (11), a partir de uma sequência de 1000 valores de z_Γ. A sequência decrescente está dentro do intervalo [10 0.008], e o parâmetro α_Γ foi definido como $\alpha_\Gamma = 10$. Tanto o parâmetro α_Γ, quanto o intervalo de z_Γ foram estimados empiricamente.

Também é possível observar na Fig. 1 a curva da função gama com um ajuste definido pela Eq. (13), a fim de manter um valor definido na etapa final de otimização, tal que

$$w_\Upsilon = \Upsilon_2(z_\Gamma, \alpha_\Gamma) + k_\Upsilon \,, \qquad (13)$$
$$k_\Upsilon = w_2 - \Upsilon_2(z_\Gamma(\max\_it), \alpha_\Gamma) \,,$$

onde w_Υ é o peso de inércia utilizando a função gama incompleta, k_Υ é uma constante calculada a partir do valor do peso de inércia desejado na etapa final da otimização, e $z_\Gamma(\max\_it)$ é o último valor da sequência usada no cálculo.

Para a curva ajustada mostrada na Fig. 1, foi escolhido $w_2 = 0{,}35$, assim a curva tem um início próximo de 0,9, e um final próximo de 0,4. Este intervalo esta de acordo com a afirmação de diversos autores citados anteriormente, como sendo um bom intervalo de funcionamento para o peso de inércia do PSO, garantindo inicialmente uma busca global, seguida de uma busca local, aumentando as chances de convergência do algoritmo.

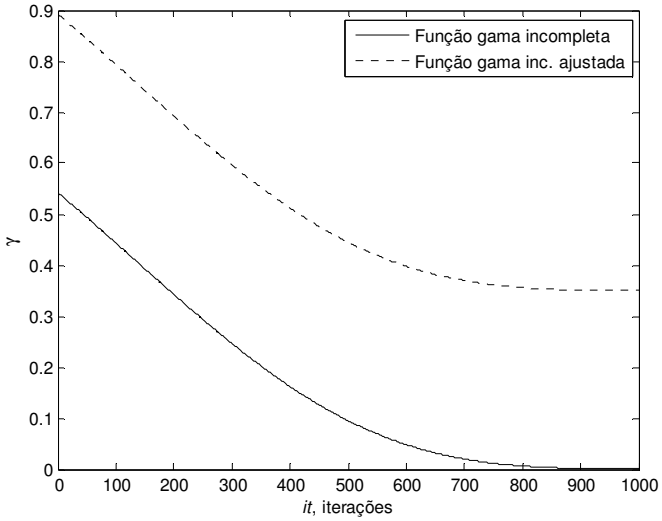

Figura 1. Curvas da função gama incompleta original e ajustada.

4. PREVISÃO DE SÉRIES TEMPORAIS E ESTUDOS DE CASO

As séries temporais podem ser definidas como um conjunto de variáveis estocásticas (probabilísticas) equiespaçadas e ordenadas no tempo (Morettin e Toloi, 2006), ou também como observações de uma variável de interesse realizadas numa sequência cronológica, que quando analisadas, podem ajudar a prever comportamentos futuros a partir de características passadas da série. Pode-se dizer que a característica mais importante de uma série temporal é que observações vizinhas são dependentes, onde o maior interesse é analisar esta dependência para se entender a dinâmica da série.

A seguir serão apresentadas as séries temporais que serão utilizadas como estudos de caso para validar o algoritmo de identificação e os algoritmos de otimização.

4.1. Sistema Box-Jenkins

Um problema de identificação de sistemas conhecido é a série de leituras de uma fornalha a gás, publicada em 1976 pelos autores Box e Jenkins (1976). A série consiste em 296 pares de amostras de entrada e saída de uma fornalha, com tempo de amostragem de nove segundos. A entrada *x(k)* é o fluxo de gás metano (pés^3/min), e a saída *y(k)* é a concentração de gás carbônico (CO_2), sendo *k* o índice da amostra. A série foi retirada do banco de dados da Universidade de York, disponível em "Banco de dados da Universidade de York".

As entradas selecionadas para o algoritmo de identificação foram escolhidas de acordo com Tzafestas e Zikidis (2001), e são *y*(*k*-1), *y*(*k*-2) e *y*(*k*-3), *x*(*k*-1), *x*(*k*-2) e *x*(*k*-3), a previsão será $\hat{y}(k)$. Todas as amostras da série foram normalizadas no intervalo [0 1], sendo que as primeiras 180 amostras foram selecionadas para o treinamento do algoritmo, e as 116 amostras restantes para validação do algoritmo.

4.2. Sistema caótico de Mackey-Glass

Um sistema caótico pode ser definido como um sistema sensível a pequenas variações nas condições iniciais e/ou parâmetros (Alligood *et al.,* 1996), fazendo-se necessário o uso de uma ferramenta que permita quantificar o caos do sistema estudado. Alligood *et al.* (1996), demonstra a aplicação do cálculo do expoente de Lyapunov, que indica se um sistema é caótico, quase-periódico ou periódico, e é capaz de quantificar o caos de um dado sistema.

O expoente de Lyapunov pode ser calculado discretamente a partir da Eq. (14).

$$\lambda = \frac{1}{\Delta n}\left(\ln \frac{d_n}{d_0}\right), \tag{14}$$

onde, Δn é o número de passos, d_0 é a distância inicial entre os pontos e d_n é a distância entre os pontos após Δn passos.

A interpretação dos valores do expoente de Lyapunov é que sistemas caóticos irão gerar valores positivos do expoente, sistemas quase-periódicos terão valores iguais a zero, e sistemas periódicos terão valores negativos do expoente de Lyapunov.

A série temporal Mackey-Glass é gerada a partir de uma equação diferencial com comportamento caótico e é dada pela Eq. (15),

$$\frac{dx(t)}{dt} = \frac{0{,}2x(t-\tau)}{1+x^{10}(t-\tau)} - 0{,}1x(t), \tag{15}$$

onde τ é o atraso (Abe, 1999; Li *et al.,* 2012). De acordo com o cálculo do expoente de Lyapunov apresentado anteriormente, foi possível determinar o atraso e a condição inicial que resultariam no comportamento mais caótico possível, conforme a Tab. 1 exibe.

Tabela 1. Parâmetros da série Mackey-Glass e respectivos expoentes de Lyapunov.

τ	x_0	λ
17	0,24	$3{,}349 * 10^{-3}$
34	0,68	$3{,}533 * 10^{-3}$

Os atributos escolhidos para o algoritmo de identificação são *x*(*k*), *x*(*k*-6), *x*(*k*-12) e *x*(*k*-18), e a saída estimada é $\hat{x}(k+6)$, conforme (Abe, 1999; Li *et al.,* 2012) sugerem. Foram geradas 2500 amostras normalizadas no intervalo [0 1] através do método de Euler, sendo que as 500 amostras iniciais foram descartadas a fim de evitar o início transitório.

4.3. Sistema de Lorenz

O sistema de equações diferenciais de Lorenz foi desenvolvido com o objetivo de modelar o movimento de convecção do calor em um fluído como água ou ar (Li *et al.,* 2012). Após algumas simplificações, Lorenz obteve três equações diferenciais ordinárias,

$$\begin{cases} \frac{dx}{dt} = -\sigma x(t) + \sigma y(t), \\ \frac{dy}{dt} = rx(t) - y(t) - x(t)z(t), \\ \frac{dz}{dt} = x(t)y(t) - cz(t), \end{cases} \tag{16}$$

onde σ,*r* e *c* são parâmetros do sistema.

Lorenz (Li *et al.,* 2012) observou que ao utilizar σ =10 e *b* = 8/3, e o parâmetro *r*, também conhecido como número de Rayleigh (ou Reynolds), maior que aproximadamente $r \approx 24{,}74$, o sistema se comportava caoticamente, mostrando também uma sensibilidade às condições iniciais escolhidas, Alligood *et al.* (1996).

Sendo assim, método de Euler foi utilizado para resolução das equações diferenciais e geração de 3000 amostras. Os valores dos parâmetros foram escolhidos de acordo com Alligood *et al.* (1996) e Li *et al.* (2012), e são σ =10 e *b* = 8/3 e *r* = 28. As condições iniciais escolhidas foram *x*(0) = 12, *y*(0) = 2 e *z*(0) = 9. Através do cálculo do expoente de Lyapunov, presente na Tab. 2, foi possível identificar que a coordenada *x* possui o comportamento mais caótico entre as 3 dimensões.

Tabela 2. Expoentes de Lyapunov para cada dimensão da série de Lorenz.

λ_x	$3{,}082 * 10^{-3}$
λ_y	$3{,}074 * 10^{-3}$
λ_z	$3{,}062 * 10^{-3}$

5. ANÁLISE DE RESULTADOS

Este trabalho propõe o emprego do algoritmo PSO e suas variações apresentadas anteriormente, para a otimização das ponderações do algoritmo ELM e da função de ativação TAF. Será simulado uma rede contendo 30 neurônios ocultos, sendo que serão testadas configurações contendo 3, 15 e 27 neurônios não-lineares, representados pela Eq. (7), e o restante serão neurônios lineares, representados pela Eq. (6).

Os parâmetros escolhidos para os três algoritmos realizarem as otimizações são os seguintes, o PSO clássico com peso de inércia w = 0.4 a variação do PSO com peso de inércia dinâmico, DPSO, $w = 0{,}9 \rightarrow 0{,}4$, e também a variação proposta pelos autores empregando a função gama incompleta para gerar os valores do peso de inércia, GPSO.

Para os algoritmos de otimização obterem soluções de melhor qualidade para o problema de minimização do *fitness*, a função objetivo escolhida para minimização é a média da soma dos erros ao quadrado, em inglês, *Mean sum of Squares Error* (*MSE*). Será incluído um termo que consiste na média da soma dos quadrados das ponderações, que permite aprimorar a generalização selecionando ponderações menores. Isto força a rede a responder mais suavemente e com sobressinais menores. Esta função é definida pelas Equações (17) e (18),

$$fitness = (\mathrm{K})MSE + (1-\mathrm{K})\frac{1}{\tilde{N}}\sum_{j=1}^{\tilde{N}} w_j^2 \ , \tag{17}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 \ , \tag{18}$$

onde e_i é o erro de previsão da i-ésima amostra distinta, N é o número de amostras distintas, w_j é a ponderação de saída do algoritmo ELM, $\tilde{N}$ é o número de neurônios ocultos, e K é uma ponderação na faixa entre [0 1].

5.1. Caso do Sistema Box-Jenkins

Para a série de *Box-Jenkins*, o algoritmo GPSO mostrou-se superior na simulação que empregava a maior proporção de neurônios não-lineares, portanto, contendo o maior número de variáveis. Os valores do *fitness* dos três algoritmos podem ser observados na Tab. 3.

Tabela 3. Resultado para a série Box-Jenkins

Neur. Ocultos	Neur. Não-Lineares	PSO	DPSO	GPSO
30	3	$7{,}236 * 10^{-5}$	$\mathbf{7{,}214 * 10^{-5}}$	$7{,}274 * 10^{-5}$
	15	$\mathbf{4{,}780 * 10^{-5}}$	$4{,}907 * 10^{-5}$	$4{,}791 * 10^{-5}$
	27	$4{,}363 * 10^{-5}$	$4{,}280 * 10^{-5}$	$\mathbf{4{,}130 * 10^{-5}}$

A Fig. 2 mostra os sinais de saída real, saída prevista e sinal de erro, para o conjunto de validação. Pode-se observar que o algoritmo ELM e função TAF otimizados pelo GPSO foram capazes de identificar as dinâmicas do sistema. O erro médio absoluto para o conjunto de validação foi de 2,75%.

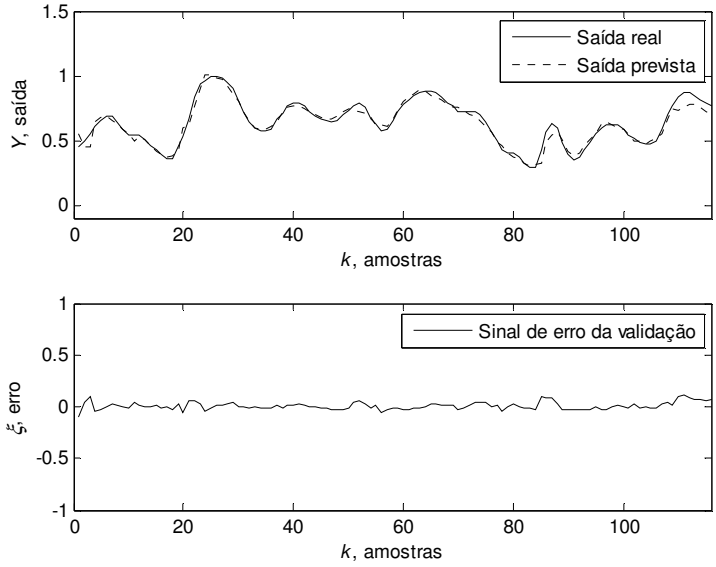

Figura 2. Previsão e sinal de erro para o conjunto de validação da série de Box-Jenkins.

5.2. Caso do Sistema Caótico Mackey-Glass

Na série caótica de Mackey-Glass, o algoritmo GPSO obteve o melhor resultado novamente na situação empregando o maior número de neurônios não-lineares, o que pode ser observado na Tab. 4.

Tabela 4. Resultado para a série de Mackey-Glass.

Neur. Ocultos	Neur. Não-Lineares	PSO	DPSO	GPSO
30	3	$\mathbf{5,191 * 10^{-3}}$	$5{,}260 * 10^{-3}$	$5{,}234 * 10^{-3}$
	15	$\mathbf{3,332 * 10^{-3}}$	$3{,}539 * 10^{-3}$	$3{,}406 * 10^{-3}$
	27	$3{,}320 * 10^{-3}$	$3{,}133 * 10^{-3}$	$\mathbf{3,063 * 10^{-3}}$

A Fig. 3 mostra os sinais de saída real, saída prevista e sinal de erro, para o conjunto de validação, onde é possível notar que as dinâmicas caóticas do sistema foram identificadas pelo modelo, apesar de o sinal de erro possuir uma amplitude superior em relação ao estudo anterior. O erro médio absoluto para o conjunto de validação foi de 6,9%.

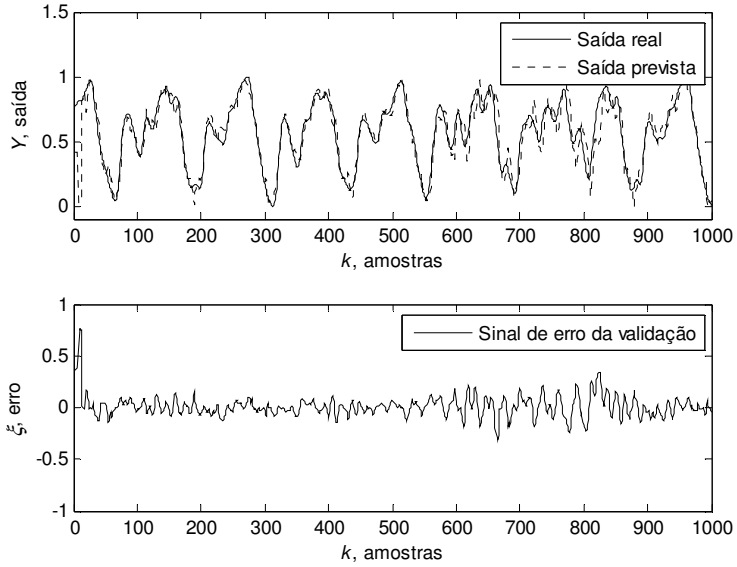

Figura 3. Previsão e sinal de erro para o conjunto de validação da série de Mackey-Glass.

5.3. Caso do Sistema de Lorenz

Na série caótica de Lorenz, o algoritmo GPSO obteve o melhor resultado novamente na situação empregando o maior número de neurônios não-lineares, e também na situação empregando a metade dos neurônios não-lineares, fatos que podem ser observados na Tab. 5.

A Figura 4 mostra os sinais de saída real, saída prevista e sinal de erro, para o conjunto de validação. É possível notar que o erro de validação é pequeno, sendo que o erro absoluto médio é de apenas 0,4%, o menor entre os três estudos.

Tabela 5. Resultado para a série de Lorenz.

Neur. Ocultos	Neur. Não-Lineares	PSO	DPSO	GPSO
30	3	$6{,}243 * 10^{-5}$	$\mathbf{5,681 * 10^{-5}}$	$5{,}987 * 10^{-5}$
	15	$1{,}728 * 10^{-5}$	$1{,}865 * 10^{-5}$	$\mathbf{1,565 * 10^{-5}}$
	27	$1{,}605 * 10^{-5}$	$1{,}324 * 10^{-5}$	$\mathbf{1,181 * 10^{-5}}$

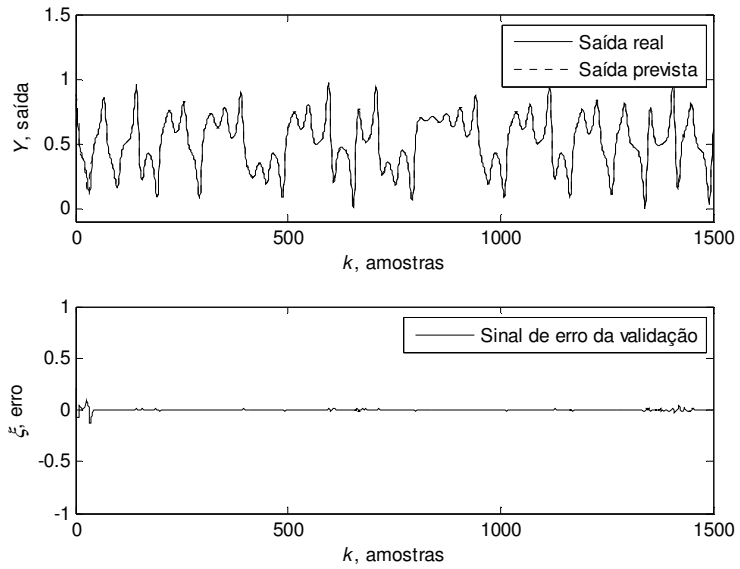

Figura 4. Previsão e sinal de erro para o conjunto de validação da série de Lorenz.

6. CONCLUSÃO

As redes neurais utilizando o método de treinamento denominado Máquina de Aprendizagem Extrema, foram capazes de identificar as três séries temporais escolhidas como estudo de caso neste trabalho.

O algoritmo ELM se mostrou uma ferramenta poderosa e eficiente para a aprendizagem de redes neurais. Com um tempo de treinamento inferior a 0,1s, a sua aplicação em conjunto com algoritmos de otimização, para a sintonia das ponderações, supera tanto em tempo, quanto em qualidade, o seu concorrente mais popular, o método de retropropagação do erro.

Sobre o algoritmo de otimização, conclui-se que o algoritmo de enxame de partículas foi capaz de otimizar os parâmetros da função de ativação variável, além das ponderações do algoritmo ELM com sucesso, com todas variações do algoritmo PSO original convergindo para valores próximos da melhor solução em todos os experimentos.

A utilização da função de ativação variável se mostrou útil, obtendo os melhores resultados da função *fitness* quando foi empregada uma proporção igual ou superior ao número de neurônios lineares utilizados.

O algoritmo de otimização GPSO obteve resultados promissores, tendo um melhor desempenho que o PSO clássico e o DPSO nas situações com um número de variáveis superior. Assim, este trabalho abre as portas para pesquisas na aplicação da função gama incompleta no projeto de metaheurísticas de otimização.

Também ressalta o potencial do algoritmo ELM em aplicações conjuntas com algoritmos de otimização, tanto mono-objetivos, quanto multiobjetivos, sendo possível a sua aplicação em sistemas reais, graças ao seu tempo de processamento baixo.

7. REFERÊNCIAS

Abe, S., 1999, "Fuzzy Function Approximators with Ellipsoidal Regions". IEEE Transactions on Systems, Man and Cybernetics, Vol. 29, No. 4, pp. 654-661.

Alligood, K.T., Sauer, T.D., Yorke, J.A., 1996, "Chaos: An Introduction To Dynamical Systems". 1st Ed. Springer, US.

Banco de dados da Universidade de York. Disponível em: <http://www.york.ac.uk/depts/maths/data/ts/>. Acesso em: 10/05/2015.

Box, G.E.P., Jenkins, G.M., 1976, "Time Series Analysis: Forecasting and Control", 2nd ed. San Francisco, CA: Holden-Day.

De Castro, L.N., 2006, "Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications", Chapman & Hall/CRC, Boca Raton, FL, USA.

Engelbrecht, A.P., 2005, "*Fundamentals of Computational Swarm Intelligence*". 1st. ed. John Wiley & Sons.

Haykin, S., 2008, "Neural Networks and Learning Machines", 3rd Edition, Prentice Hall, USA.

Huang, G.B., Zhu, Q.Y., Siew, C.K., 2004, "Extreme Learning Machine: A new Learning Scheme of Feedforward Neural Networks". In: Proceedings of International Joint Conference on Neural Networks, Budapest, Hungary, pp. 985-990.

Huang, G.B., Zhu, Q.Y., Siew, C.K., 2006, "Extreme Learning Machine: Theory and Applications. Neurocomputing". v. 70, pp. 489-501.

Karapetyan, N.V., 2009, "Dividend Optimization under the Gamma Distribution of Claims". Moscow University Mathematics Bulletin, Vol. 64, No. 5, pp. 219-221.

Kennedy, J., Eberhart, R.C., 1995, "Particle Swarm Optimization". In: Proceeding of the IEEE Conference on Neural Networks, Perth, Australia, Vol. 4, pp. 1942-1948.

Krishnamoorthy, K., Mathew, T., Mukherjee, S., 2008, "Normal-Based Methods for a Gamma Distribution: Prediction and Tolerance Intervals and Stress-Strenght Reliability". Technometrics. Vol. 51, pp. 69-78.

Kulaif, A.C.P., 2014, "Técnicas de Regularização para Máquinas de Aprendizado Extremo", Dissertação de mestrado, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, SP.

Li, D., Han, M., Wang, J., 2012, "Chaotic Time Series Prediction Based on a Novel Robust Echo State Network". IEEE Transactions on Neural Networks and Learning Systems. Vol0. 23, No. 5, pp. 787-799.

Mello, M.H.A., Arruda, H.V., Ortolani, A.A., 1994, "Probabilidade de Ocorrência de Totais Pluviais Máximos Horários, em Campinas – São Paulo". Revista do Instituo Geológico. Vol. 15, No. 1/ 2, pp. 59-67.

Morettin, P.A.; Toloi, C.M., 2006, "Análise de Séries Temporais". 2ª Ed. Edgard Blücher, São Paulo.

Silva, D.N.G., Pacifico, L.D.S., Ludermir, T.B., 2011, "An Evolutionary Extreme Learning Machine Based on Group Search Optimization". In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 574-580.

Shi, Y., Eberhart, R.C., 1999, "Empirical Study of Particle Swarm Optimization". In: Proceedings of the Evolutionary Computation, Washington D.C., USA, Vol. 3, pp. 1945-1950.

Temme, N.M., 1992, "Asymptotic Inversion of Incomplete Gamma Functions". Mathematics of Computation. Vol. 58, No. 198, pp. 755-764.

Tzafestas, S.G., Zikidis, K.C., 2001, "NeuroFAST: On-Line Neuro-Fuzzy ART-Based Structure and Parameter Learning TSK Model". IEEE Transactions on Systems, Man and Cybernetics. Vol. 31, No. 5, pp. 797-802.

Wang, G.T., Li, P., Cao, J.T., 2012, "Variable Activation Function Extreme Learning Machine Based on Residual Prediction Compensation. Soft Computing". Vol. 16, No. 9, pp. 1477-1484.

EXTREME LEARNING MACHINE WITH PARTICLE SWARM OPTIMIZATION APPLIED TO TIME SERIES FORECASTING

Rafael Bartnik Grebogi, rafagrebogi@gmail.com[1]
Deivid Zancheta de Lima, dd_lages@hotmail.com[2]
Eduardo Massashi Yamao, massashi_yamao@yahoo.com.br[3]
Marco Antonio Reichert Boaretto, marco.boaretto@hotmail.com[3]
Leandro dos Santos Coelho, leandro.coelho@pucpr.br[4]

[1]Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), Universidade Federal do Paraná (UFPR), Curitiba, PR, Brasil, e Departamento de Processos Industriais, Instituto Federal de Santa Catarina (IFSC), Lages, SC, Brasil
[2]Curso Técnico de Eletromecânica, Departamento de Processos Industriais, Instituto Federal de Santa Catarina (IFSC), Lages, SC, Brasil
[3]Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), Universidade Federal do Paraná (UFPR), Curitiba, PR, Brasil
[4]Programa de Pós-Graduação em Engenharia de Produção e Sistemas (PPGEPS), Escola Politécnica, Pontifícia Universidade Católica do Paraná (PUCPR) e Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), Universidade Federal do Paraná (UFPR), Curitiba, PR, Brasil

Abstract. *The purpose of this paper is to optimize the application of extreme learning machines on time series prediction. The extreme learning machine algorithm, in this context, works as an alternative due to the limitations regarding the computational cost and training time of the multi-layer perceptron artificial neural networks. It is applied the classical particle swarm optimization algorithm along with more two of its variations as an alternative of a linear optimization method of the extreme learning machine algorithm weights. To prove the effectiveness of this optimization approach will be compared the applications on three case studies of time series forecasting. The predictions performed by the extreme learning machine algorithm along with the particle swarm optimization technique has revealed very promising results, proving that the combination between both techniques to be an effective approach for time series forecasting and nonlinear system identification.*

Keywords: *Extreme learning machine; optimization; particle swarm optimization; system identification; time series forecasting.*