REDE NEURO-FUZZY EVOLUTIVA COM NEURÔNIOS BASEADOS EM UNINORMAS PARA PREVISÃO DE SÉRIES TEMPORAIS

Raul Rosa*, Rosangela Ballini†, Fernando Gomide*

*Departamento de Engenharia de Computação e Automação Industrial
Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas
Av. Albert Einstein, 400 - Campinas, SP, Brasil - CEP: 13083-852

†Instituto de Economia
Universidade Estadual de Campinas
Rua Pitágoras, 353 - Campinas, SP, Brasil - CEP: 13083-857

Emails: raulrosa@dca.fee.unicamp.br, ballini@eco.unicamp.br, gomide@dca.fee.unicamp.br

Abstract— This paper introduces an evolving hybrid neural network modeling approach using neurons based on uninorms and sigmoidal activation functions. The evolving neural network simultaneously adapts its structure and updates its weights using a stream of data. Currently, learning from data streams is a challenging and important issue because often traditional learning methods are impracticable in these circumstances. Uninorm-based neurons generalize the fuzzy neurons models based on triangular norms and conorms. Uninorms increase flexibility and generality of fuzzy neuron models because they can modify their processing capabilities by adjusting identity elements. In addition to structural plasticity, this feature adds processing plasticity in neural network structures. A recursive procedure to granulate the input space and uncover the neural network structure, and an extreme learning-based algorithm is developed to train the neural network. Computational results show that the evolving neural fuzzy network is competitive when compared with representative methods of the current state of the art of evolving modeling.

Keywords— hybrid neural fuzzy systems, unineurons, extreme learning, clouds, evolving systems

Resumo— Esse trabalho apresenta uma abordagem para modelagem de sistemas usando uma rede neural híbrida evolutiva. A rede neural é construída com neurônios baseados em uninormas e neurônios clássicos. Redes neurais evolutivas adaptam suas estruturas e parâmetros simultaneamente usando um fluxo de dados. Presentemente, aprendizagem em contextos que envolvem fluxo de dados é um desafio para os métodos de aprendizagem convencionais. Modelos de neurônios baseados em uninormas generalizam neurônios construídos com normas e conormas triangulares. Uninormas tornam os neurônios *fuzzy* mais flexíveis pois permitem adaptar o processamento neural ajustando o elemento identidade. Além da plasticidade estrutural, essa característica aumenta a plasticidade de redes neurais evolutivas quanto ao processamento neural. Um método recursivo é usado para granularizar o espaço de entrada e encontrar a estrutura da rede neural. Um algoritmo baseado em aprendizagem extrema atualiza os pesos da rede. Juntos, o procedimento de granularização e o de atualização de pesos compõem o algoritmo de aprendizagem da rede neural híbrida evolutiva. Resultados computacionais mostram que a abordagem proposta é competitiva quando comparada com métodos representativos do estado da arte em modelagem evolutiva.

Palavras-chave— sistemas *fuzzy* neurais híbridos, unineurônios, aprendizagem extrema, *clouds*, sistemas evolutivos

1 Introdução

Série temporal é uma sequencia de dados, tipicamente indexada por intervalos uniformes de tempo. Análise de séries temporais envolve métodos com o intuito de extrair estatísticas e outras características dos dados. Previsão de séries temporais trata do uso de modelos para prever valores futuros de uma série temporal utilizando valores observados previamente, isto é, valores passados da série. Previsão de séries temporais é um tema importante em várias áreas, entre elas processamento de sinais, sistemas de controle, comunicação, reconhecimento de padrões, econometria, etc.

Modelagem de séries temporais tradicionalmente utiliza métodos estatísticos (Overbey et al., 2007), mas usualmente esses métodos não são suficientes para capturar a natureza dinâmica e complexa das séries temporais, principalmente quando elas se originam de processos não lineares e não estacionários (Wang et al., 2012). Nos últimos anos, métodos clássicos de modelagem foram generalizados e novas abordagens desenvolvidas para tratar séries temporais complexas de forma adaptativa.

Sistemas *neuro-fuzzy* combinam a capacidade de aprendizagem e aproximação das redes neurais com a aptidão de representação e processamento de informação imprecisa dos sistemas *fuzzy* (Lee and Lee, 1975). Redes *neuro-fuzzy* para a previsão de séries temporais é uma alternativa atrativa tanto em aplicações como no desenvolvimento de novos modelos de neurônios e redes neurais. Por exemplo, uninormas e nullnormas são operadores que generalizam t-normas e s-normas e modelos de neurônios *fuzzy* baseados nestes operadores foram propostos em Pedrycz (2006), Hell et al. (2008), Lemos et al. (2010), Hell et al. (2009) e Leite et al. (2012). Aplicação de redes *neuro-fuzzy* evolutivas

incluem previsão de preço de ações, modelo térmico de transformadores e estimação de estado, entre outras (Nguyen and Quek, 2010)(Souza et al., 2012)(Wang et al., 2012).

Este trabalho sugere uma abordagem baseada em redes neurais *fuzzy* evolutivas híbridas (eRFH) para modelar sistemas e séries temporais. Redes *neuro-fuzzy* evolutivas são redes capazes de adaptar continuamente sua estrutura e atualizar seus pesos processando um fluxo de dados (Angelov and Zhou, 2006). Devido à sua natureza adaptativa, modelos na forma de redes *neuro-fuzzy* evolutivas podem adequar-se às variações que ocorrem em um sistema.

A rede neural *fuzzy* evolutiva híbrida objeto deste trabalho tem uma estrutura do tipo *feed-forward* com duas camadas. Ela utiliza uninormas como operador de processamento sináptico dos neurônios *fuzzy*, neurônios estes que compõem a camada intermediária. A camada de saída é composta por neurônios clássicos, isto é neurônios com produto algébrico no processamento sináptico, soma algébrica na agregação e função de ativação sigmoidal. A estrutura da rede é determinada pelo número de neurônios *fuzzy* da camada intermediária e este número, por sua vez, é determinado por um algoritmo baseado no conceito de *cloud*. Uma *cloud* pode ser vista como um conjunto de pontos cuja densidade implicitamente define um grupo (Angelov and Yager, 2011). O uso deste conceito em modelagem e controle é sugerido em Sadeghi-Tehran et al. (2012). As *clouds* granularizam o espaço de entrada segundo a densidade de seus pontos. No contexto de sistemas evolutivos as densidades são calculadas de forma recursiva. A adaptação dos pesos da rede neural baseia-se na aprendizagem extrema (*ELM-Extreme Learnig Machine*) conforme Huang et al. (2004). Basicamente, a aprendizagem extrema atribui valores aleatórios aos pesos da camada intermediária da rede e ajusta os pesos da camada de saída através do algoritmo dos quadrados mínimos recursivo.

Para avaliar o desempenho da rede neural *fuzzy* evolutiva híbrida (eRFH) proposta neste trabalho, dados relativos a dois problemas clássicos de modelagem e de previsão de séries temporais são considerados, o forno a gás de Box and Jenkins e a série temporal de Mackey-Glass, respectivamente. Os resultados de simulação mostram que, sob o ponto de vista do critério do erro quadrático médio, a eRFH é competitiva quando comparada com métodos representativos do estado da arte em modelagem evolutiva na área, DENFIS (Kasabov and Song, 2002), eTS (Angelov and Zhou, 2006), ANYA (Angelov and Yager, 2011) e também com o ELM (Huang et al., 2004).

Este trabalho está organizado da seguinte forma. Após esta introdução, a próxima seção apresenta a fundamentação, descreve a eRFH e seu algoritmo de aprendizagem. A Seção 3 descreve os resultados de simulação. Finalmente, a Seção 4 resume as contribuições deste artigo e sugere temas para investigação futura.

2 Fundamentação teórica e algoritmos

Esta seção apresenta os conceitos necessários para a formulação e construção da rede neural *fuzzy* evolutiva híbrida (eRFH). Após rever o conceito de uninormas e a concepção dos unineurônios, descreve-se o algoritmo que granulariza o espaço de entrada e determina a estrutura da rede, isto é, o número de unineurônios que compõem a camada intermediária. Um algoritmo de aprendizagem extrema para determinar os pesos da camada intermediária e da camada de saída completa o processo de aprendizagem.

2.1 Uninorma e Unineurônios

Uma uninorma é um operador binário que generaliza t-normas e s-normas (Yager and Rybalov, 1996). Formalmente, uma uninorma é um mapeamento $u : [0,1] \times [0,1] \rightarrow [0,1]$ que possui as seguinte propriedades

1. comutatividade: $a\ u\ b = b\ u\ a$
2. associatividade: $a\ u\ (b\ u\ c) = (a\ u\ b)\ u\ c$
3. monotonicidade: se $b \leq c$, então $a\ u\ b \leq a\ u\ c$
4. elemento identidade: $a\ u\ e = a, \forall\ a, e \in [0,1]$

Quando $e = 1$ a uninorma é uma t-norma e quando $e = 0$ a uninorma é uma s-norma.

Neurônios *fuzzy* do tipo unineurônios são a unidade de processamento básica da rede neural *fuzzy* evolutiva híbrida. Neurônios *fuzzy* baseados em normas triangulares são os neurônios do tipo *and* e *or* (Pedrycz and Rocha, 1993). O unineurônio, portanto, tem os os neurônios *and* e *or* como casos particulares.

O modelo do unineurônio tem a seguinte forma

$$U_U(\lambda, \mathbf{w}) = U_{i=1}^{N_l} \lambda_i\ u\ w_i, \tag{1}$$

onde λ_i é a i-ésima entrada e w_i é o peso correspondente. Em particular, se $N_l = 1$, então a saída z_l do unineurônio l será

$$z_l = U_U = \lambda_l\ u\ w_l,$$

onde $l = 1, \ldots, L^t$ para uma rede neural *fuzzy* com L^t unineurônios na camada intermediária.

Uma uninorma u é definida por seu construtor conforme o seguinte (Yager and Rybalov, 1996):

$$a\ u\ w = \begin{cases} e + (1-e)\left(\frac{(a-e)}{(1-e)} s \frac{(w-e)}{(1-e)}\right) & \text{se } a, w \in (e,1] \\ e(\frac{a}{e} t \frac{w}{e}) & \text{caso contrário} \end{cases}, \tag{2}$$

onde t é uma t-norma e s uma s-norma. Este trabalho considera que a t-norma é o produto algébrico ($a\ t\ b = ab$) e a s-norma é soma algébrica ($a\ s\ b = a+b-ab$). Estas escolhas resultam em superfície suaves para a função do unineurônio *fuzzy*.

2.2 Granularização do espaço de entrada

A granularização do espaço de entrada utiliza o conceito de *clouds* proposto em Angelov and Yager (2011). O propósito das *clouds* é criar uma partição do espaço de entrada sem formar explicitamente grupos e, com isto, formar uma estrutura de grupo (*cluster structure*). Uma *cloud* é caracterizada por uma densidade que representa uma distribuição de dados no espaço de entrada. Uma coleção de *clouds* tem papel análogo ao de uma estrutura de grupos. No contexto de sistemas evolutivos é necessário determinar as *clouds* de maneira recursiva, pois os dados devem ser processados de acordo com seu fluxo.

A densidade local γ_i^t da entrada $\mathbf{x}^t = [x_1, \ldots, x_n]$ em t, para a i-ésima *cloud*, é calculada recursivamente como

$$\gamma_i^t = \frac{1}{1 + ||\mathbf{x}^t - \mu_i^t||^2 + \Sigma_i^t - ||\mathbf{x}^t||^2}, \qquad (3)$$

onde $\mu_i^t = ((M_i^t - 1)/(M_i^t))\mu_i^{t-1} + \mathbf{x}^t/(M_i^t)$, $\mu_1^1 = \mathbf{x}^1$ é o valor modal local, M_i^t é o número de dados de entrada associados à i-ésima *cloud*, i é o número de *clouds*, $i = 1, ..., L^t$. A dispersão Σ_i^t é calculada usando

$$\Sigma_i^t = \frac{M_i^t - 1}{M_i^t}\Sigma_i^{t-1} + \frac{1}{M_i^t}||\mathbf{x}^t||^2, \quad \Sigma_1^1 = ||\mathbf{x}^1||^2.$$

Outra medida importante é a da densidade global, calculada similarmente, porém considerando todos os dados de entrada. Assim, tanto o valor modal quanto a dispersão modificam. A densidade global Γ^t em t é calculada recursivamente da seguinte forma

$$\Gamma^t = \frac{1}{1 + ||\mathbf{x}^t - \mu_G^t||^2 + \Sigma_G^t - ||\mathbf{x}^t||^2}, \qquad (4)$$

onde $\mu_G^t = ((t-1)/t)\mu_G^{t-1} + \mathbf{x}^t/t$, $\mu_G^1 = \mathbf{x}^1$ é o valor modal global de todos os dados em t, e a dispersão global

$$\Sigma_G^t = \frac{t-1}{t}\Sigma_G^t - 1 + \frac{1}{t}||\mathbf{x}^t||^2, \quad \Sigma_G^1 = ||\mathbf{x}^1||^2.$$

Usualmente a primeira amostra de dados define a primeira *cloud* e seu respectivo ponto focal, isto é, $\mathbf{X}_1^f \leftarrow \mathbf{x}^1$. O ponto focal $\mathbf{X}_i^f$ é uma informação importante pois ele corresponde ao dado de entrada com maior densidade local e global da da i-ésima *cloud*. Uma *cloud* é criada se a densidade global de um dado de entrada é maior do que a densidade global estimada de cada *cloud* no seu ponto focal, isto é $\Gamma^t > \Gamma_i^f \ \ \forall i, i = 1, ..., L^t$, onde Γ_i^f é a densidade global do ponto focal da *cloud* i. Caso o dado candidato não satisfaça a condição de densidade global, atualiza-se a *cloud* cuja densidade local do dado de entrada é a maior, ou seja, a *cloud* que satisfaz

$$Cloud_i = \arg\max_i(\gamma_i^t). \qquad (5)$$

A atualização do ponto focal de uma *cloud* i depende de dois parâmetros, a densidade global do seu ponto focal, Γ_i^f, e a densidade local do ponto focal, γ_i^f. Para que um dado seja considerado o novo ponto focal de uma *cloud* i, tanto sua densidade global, como sua densidade local devem ser maiores que Γ_i^f e γ_i^f, respectivamente.

O grau de compatibilidade de um dado de entrada $\mathbf{x}^t$ às *clouds* existentes é calculado da seguinte forma

$$\lambda_i^t = \frac{\gamma_i^t}{\sum_{j=1}^{L^t}\gamma_j^t}, \ i = 1, ..., L^t, \qquad (6)$$

onde γ_i^t é a densidade local da entrada $\mathbf{x}^t$ na i-ésima *cloud*, conforme expressão 3.

O seguinte algoritmo resume o processo de granularização do espaço de entrada e o cálculo do grau de compatibilidade.

ler a primeira amostra de dados
inicializar a densidade global $\mu_G^1 \leftarrow \mathbf{x}^1$;
$\Sigma_G^1 \leftarrow ||\mathbf{x}^1||^2$; $\Gamma^1 \leftarrow 1$
inicializar a primeira *cloud* $\mu_1^1 \leftarrow \mathbf{x}^1$; $M_1^1 \leftarrow 1$;
$\Sigma_1^1 \leftarrow ||\mathbf{x}^1||^2$; $\gamma_1^1 \leftarrow 1$; $\mathbf{X}_1^f \leftarrow \mathbf{x}^1$; $\gamma_1^f \leftarrow \gamma_1^1$;
$\Gamma_1^f \leftarrow \Gamma^1$
while existirem dados de entrada **do**
  ler o vetor entrada $\mathbf{x}^t$
  atualizar a densidade global (4)
  calcular a densidade local para cada *cloud* (3)
  if $\Gamma^t > \Gamma_i^f \ \forall i | i = [1, L^t]$ **then**
    criar uma nova *cloud* $i \leftarrow i + 1$; $\mu_i^t \leftarrow \mathbf{x}^t$;
    $M_i^t \leftarrow 1$; $\Sigma_i^t \leftarrow ||\mathbf{x}^t||^2$; $\gamma_i^t \leftarrow 1$; $\mathbf{X}_i^f \leftarrow \mathbf{x}^t$;
    $\gamma_i^f \leftarrow \gamma_i^t$; $\Gamma_i^f \leftarrow \Gamma^t$
  else
    encontrar $Cloud_i$ que satisfaça (5)
    atualizar a $Cloud_i$ $M_i^t \leftarrow M_i^t + 1$;
    if $\gamma_i^t > \gamma_i^f$ **and** $\Gamma^t > \Gamma_i^f$ **then**
      atualizar o ponto focal $\mathbf{X}_i^f \leftarrow \mathbf{x}^t$; $\gamma_i^f \leftarrow \gamma_i^t$; $\Gamma_i^f \leftarrow \Gamma^t$
    end if
  end if
  calcular o grau de compatibilidade segundo (6)
end while

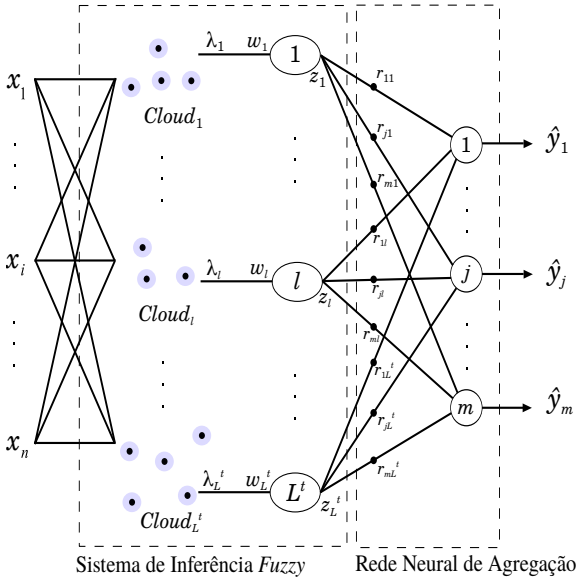

Figura 1: Estrutura da rede neural *fuzzy* híbrida.

2.3 Estrutura da rede neural fuzzy

A rede neural considerada neste trabalho é uma eRFH *feedforward* com três camadas. A primeira camada é composta por L^t *clouds* e a segunda por L^t unineurônios. Juntas, estas duas camadas compõem um sistema de inferência *fuzzy*. A camada de saída é composta por neurônios clássicos com função de ativação sigmoidal. Esta representa uma função de agregação e completa o modelo neural *fuzzy*. Os graus de compatibilidade dos dados de entrada λ_i^t correspondem aos valores normalizados das densidades locais para cada *cloud*. O dado de entrada $\mathbf{x}^t$ é enviado a todas as *clouds*. O número de *clouds* em t é L^t e este também corresponde ao número de unineurônios na segunda camada ou, equivalentemente, ao número de regras *fuzzy* que representa o modelo em t.

Os unineurônios da segunda camada agregam os valores da saída da camada de entrada, ponderados pelos pesos sinápticos w_l, isto é

$$z_l = \lambda_l \; u \; w_l, \tag{7}$$

onde z_l, $l = 1, \ldots, L^t$ é a saída do l-ésimo unineurônio.

Como a camada de saída contém neurônios com função de ativação sigmoidal $f(\cdot)$, a j-ésima saída é

$$\hat{y}_j = f\left(\sum_{i=1}^{L^t} r_{ji} z_i\right), \tag{8}$$

onde $j = 1, \ldots, m$ e r_{ji} é o peso sináptico entre o j-ésimo neurônioda camada de saída e o i-ésimo unineurônio da segunda camada. A figura 1 mostra a estrutura da rede.

2.4 Aprendizagem Extrema

Aprendizagem extrema (ELM) é um método para treinamento de redes neurais *feedforward* com uma camada intermediária (Huang et al., 2004). Neste método, os pesos da camada intermediária são escolhidos aleatoriamente e o algoritmo de quadrados mínimos é utilizado para determinar os da camada de saída. Similarmente, na rede neural *fuzzy* evolutiva objeto deste trabalho, os pesos e os elementos identidade dos unineurônios da camada intermediária são escolhidos aleatoriamente em (0,1) e mantidos fixos para todo t. Em seguida utiliza-se o algoritmo de mínimos quadrados recursivo com fator de esquecimento (ψ) para atualizar os pesos da camada de saída $\mathbf{R}^t = [r_{jl}^t]$ conforme o seguinte

$$\mathbf{p}^t = \mathbf{Q}^{t-1}\mathbf{z}^t\{\psi + (\mathbf{z}^t)^T\mathbf{Q}^{t-1}(\mathbf{z}^t)\}^{-1}, \tag{9}$$

$$\mathbf{Q}^t = (\mathbf{I}_{L^t} - \mathbf{p}^t(\mathbf{z}^t)^T)\psi^{-1}\mathbf{Q}^{t-1}, \tag{10}$$

$$\mathbf{R}^t = \mathbf{R}^{t-1} + (\mathbf{p}^t)^T(\mathbf{f}^{-1}(\mathbf{y}^t) - \mathbf{R}^{t-1}\mathbf{z}^t), \tag{11}$$

onde $\mathbf{f}^{-1}(\mathbf{y}^t) = \log(\mathbf{y}^t) - \log(1 - \mathbf{y}^t)$ e $\mathbf{y}^t = [y_1, \ldots, y_m]$. A matriz $\mathbf{Q}$ é inicializada com $\omega\mathbf{I}_{L^t}$, onde $\omega = 1000$ e $\mathbf{I}_{L^t}$ é a matriz de identidade $L^t \times L^t$.

O algoritmo de treinamento da eRFH é resumido a seguir.

ler a primeira amostra de dados $\mathbf{x}^1$
inicializar a primeira *cloud* segundo $\mathbf{x}^1$
inicializar a estrutura e os parâmetros da rede
while existirem dados de entrada **do**
  ler o vetor entrada $\mathbf{x}^t$
  estimar a saída $\hat{\mathbf{y}}^t$
  ler o vetor saída $\mathbf{y}^t$
  calcular potenciais locais e globais
  atualizar a estrutura da rede de acordo com o número de *clouds*
  atualizar os pesos $\mathbf{R}^t$ segundo (9) - (11)
end while

3 Resultados Computacionais

Esta seção apresenta os resultados obtidos pela rede neural *fuzzy* híbrida evolutiva eRFH e compara seu desempenho com abordagens representativas do estado da arte em modelagem evolutiva. Para isso são considerados dois exemplos clássicos, a identificação do forno a gás de *Box-Jenkins* (*gas furnace data*) e a série temporal caótica de *Mackey-Glass*.

3.1 Forno a gás de Box-Jenkins

O dados do forno a gás de Box-Jenkins representam a concentração de CO_2 (saída y^t) em função da vazão de gás (entrada u^t), resultado do processo de combustão de uma mistura metano-ar. O objetivo é prever y^t a partir de y^{t-1} e u^{t-4}. O conjunto de dados contém 290 pares de entrada

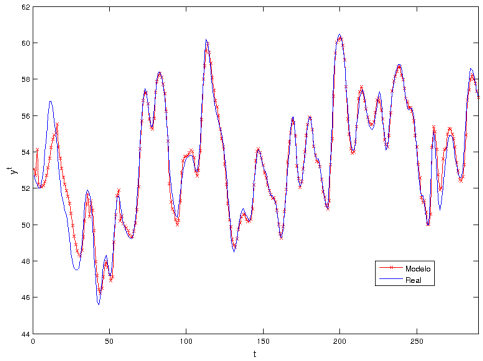

Figura 2: Saída da eRFH: Box-Jenkins.

Modelo	Regras	melhor RMSE	RMSE médio
DENFIS	12	0,0190	0,0190
eTS	12	0,0796	0,0796
ANYA	7	0,0393	0,0393
ELM	20	0,0232	0,0931
eRFH	7	0,0245	0,0398

Tabela 1: Desempenho da rede eRFH: Box-Jenkins.

e saída. Durante a simulação, todos estes dados foram utilizados para teste e validação assumindo por hipótese que os dados são processados supondo um fluxo de dados.

Diferentes estudos e.g. Kasabov and Song (2002) e Angelov and Yager (2011) sugerem um modelo do forno com duas entradas e uma saída conforme o seguinte

$$y^t = f_{BJ}(y^{t-1}, u^{t-4}).$$

Os dados foram normalizados para obter valores no intervalo [0,1, 0,9]. O desempenho, o erro quadrático médio (RMSE), foi calculado para os dados normalizados. O valor do fator de esquecimento é $\psi = 0,9$.

Os resultados, resumidos na tabela 1, consideram o melhor valor do RMSE, o valor médio do RMSE e, para os modelos evolutivos neurais *fuzzy*, o número de regras *fuzzy* equivalentes. Conforme se observa, a eRFH apresenta o terceiro menor melhor RMSE e RMSE médio quando comparada com métodos alternativos de modelagem evolutiva e o ELM. O desempenho é competitivo. A figura 2 ilustra um resultado típico da eRFH, considerando todos os (290) dados disponíveis.

3.2 Mackey-Glass

A série temporal Mackey-Glass é construída utilizando a seguinte expressão

$$\frac{dx}{dt} = \frac{Ax^{t-\tau}}{1 + (x^{t-\tau})^C} - Bx^t, A, B, C > 0.$$

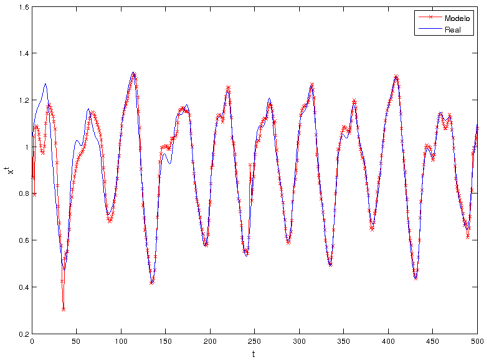

Figura 3: Previsão da rede eRFH: série Mackey-Glass.

Modelo	Regras	melhor RMSE	RMSE médio
DENFIS	25	0,0730	0,0730
eTS	24	0,0779	0,0779
ANYA	13	0,1081	0,1081
ELM	20	0,1413	0,2096
eRFH	13	0,0188	0,0435

Tabela 2: Desempenho da rede eRFH: série Mackey-Glass.

Como em outros trabalhos, Kasabov and Song (2002) e Angelov and Zhou (2006) por exemplo, adotou-se $A = 0,2$, $B = 0,1$, $C = 10$, $\tau = 17$ e passo de integração igual a $0,1$.

O objetivo é prever o valor de x^t no instante $t+85$. O modelo de previsão tem a seguinte forma

$$x^{t+85} = f_{MG}\left(x^t, x^{t-6}, x^{t-12}, x^{t-18}\right).$$

A simulação considerou 3200 dados e aqueles correspondentes a $t = 200 \ldots 3200$ foram utilizados simultaneamente para teste e validação, conforme hipótese de que os dados de entrada são processados como um fluxo de dados. Como no exemplo anterior, a tabela 2 mostra o melhor valor do RMSE, o valor médio do RMSE, e o número de regras *fuzzy* equivalentes. Neste caso o desempenho da eRFH supera os métodos evolutivos representativos do estado da arte e o ELM. É interessante notar que a eRFH também produz um menor número de regras do que os modelos DENFIS e eTS. A figura 3 ilustra um exemplo típico de previsão da série produzida pela eRFH.

4 Conclusão

Este trabalho apresentou uma rede neural evolutiva *fuzzy* híbrida com unineurônios e uma abordagem recursiva baseada em potencial para granularizar o espaço de entrada e apredizagem extrema para determinar seus pesos. O desempenho da rede foi analisado considerando problemas

de modelagem não linear e previsão de série temporal caótica. Os resultados de simulação sugerem que a rede proposta é competitiva com alternativas de modelagem evolutivas representativas do estado da arte na área. Apesar de seu desempenho promissor, ainda é necessário investigar aspectos importantes de redes neurais evolutivas como, por exemplo, a possibilidade de não só criar neurônios na camada intermediária, mas também excluir neurônios desta camada quando for oportuno. Além disso o papel do unineurônio na rede neural e sua relevância para aproximação funcional é um item a ser investigado. Outras estruturas de neurônios como aquelas baseadas em nulnormas e aprendizagem de redes neurais com nulneurônios também são assuntos para desenvolvimento futuro.

Agradecimentos

O primeiro autor agradece à CAPES pelo apoio e o terceiro ao CNPq, processo 304596/2009-4.

Referências

Angelov, P. and Yager, R. (2011). A new type of simplified fuzzy rule-based system, *International Journal of General Systems* **41**(2): 163–185.

Angelov, P. and Zhou, X. (2006). Evolving fuzzy systems from data streams in real-time, *2006 International Symposium on Evolving Fuzzy Systems*, Lake District, Reino Unido, pp. 29 –35.

Hell, M., Gomide, F., Ballini, R. and Costa, P. (2009). Uninetworks in time series forecasting, *Annual Meeting of the North American Fuzzy Information Processing Society* pp. 1–6.

Hell, M., Jr, P. C., Gomide, F. and Costa, P. (2008). Hybrid neurofuzzy computing with nullneurons, *IEEE International Joint Conference on Neural Networks* pp. 3653–3659.

Huang, G., Zhu, Q. and Siew, C. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks, *IEEE International Joint Conference on Neural Networks* **2**: 985–990.

Kasabov, N. and Song, Q. (2002). DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, *IEEE Trans. on Fuzzy Systems* **10**(2): 144 – 154.

Lee, S. and Lee, E. (1975). Fuzzy neural networks, *Mathematical Biosciences* **23**(1-2): 151–177.

Leite, D., Costa, P. and Gomide, F. (2012). Evolving granular neural network for fuzzy time series forecasting, *The 2012 International Joint Conference on Neural Networks* pp. 1–8.

Lemos, A., Caminhas, W. and Gomide, F. (2010). New uninorm-based neuron model and fuzzy neural networks, *Annual Meeting of the North American Fuzzy Information Processing Society* pp. 1–6.

Nguyen, N. N. and Quek, C. (2010). Stock price prediction using Generic Self-Evolving Takagi-Sugeno-Kang (GSETSK) fuzzy neural network, *International Joint Conference on Neural Networks*, IEEE, pp. 1–8.

Overbey, L. A., Olson, C. C. and Todd, M. D. (2007). A parametric investigation of state-space-based prediction error methods with stochastic excitation for structural health monitoring, *Smart Materials and Structures* **16**(5): 1621.

Pedrycz, W. (2006). Logic-based fuzzy neurocomputing with unineurons, *IEEE Trans. on Fuzzy Systems* **14**(6): 860 –873.

Pedrycz, W. and Rocha, A. (1993). Fuzzy-set based models of neurons and knowledge-based networks, *IEEE Trans. on Fuzzy Systems* **1**(4): 254 –266.

Sadeghi-Tehran, P., Cara, A., Angelov, P., Pomares, H., Rojas, I. and Prieto, A. (2012). Self-evolving parameter-free rule-based controller, *IEEE International Conference on Fuzzy Systems*, pp. 1–8.

Souza, L., Lemos, A. P., Caminhas, W. M. and Boaventura, W. C. (2012). Thermal modeling of power transformers using evolving fuzzy systems, *Engineering Applications of Artificial Intelligence* **25**(5): 980–988.

Wang, W., Li, D. Z. and Vrbanek, J. (2012). An evolving neuro-fuzzy technique for system state forecasting, *Neurocomputing* **87**(0): 111–119.

Yager, R. and Rybalov, A. (1996). Uninorm aggregation operators, *Fuzzy Sets and Systems* **80**(1): 111–120.