

Neo-Fuzzy-Neuron Network with Genetic Programming Learning for Non Linear Regression Problems

Glender Brás¹, Alisson Marques Silva¹

¹Programa de Pós Graduação em Modelagem Matemática e Computacional
Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG
Av. Amazonas, 7675 - Nova Gameleira – 30510-000
Belo Horizonte – MG – Brasil

glenderbras@gmail.com, alisson@cefetmg.br

Abstract. *This paper proposes two new approaches to create the rule-base for the Neo-Fuzzy-Neuron Network. The approaches use Genetic Programming (GP) to generate the rules associated with each input, creating and adjusting the membership functions. A Gradient based-method is used to update parameters. The evaluation of the models is performed considering forecast and non-linear system identification problems. The results obtained are compared with alternative models of the state of the art. The results show that the proposed algorithms are efficient and competitive.*

Resumo. *Este artigo propõe dois algoritmos baseados em Programação Genética (PG) para aprendizado em redes Neo-Fuzzy Neuron (NFN). As abordagens utilizam a PG para gerar as regras associadas a cada entrada, criando e ajustando as funções de pertinência, e um método baseado no Gradiente para ajuste de pesos. Os modelos propostos são avaliados e comparados com modelos alternativos em problemas de previsão e identificação de sistemas não lineares. Os resultados obtidos mostram que os modelos propostos são promissores e competitivos com modelos alternativos do estado da arte.*

1. Introdução

Os Sistemas Neuro-Fuzzy combinam características da teoria de conjuntos Fuzzy e de Redes Neurais. Estes sistemas integram o tratamento de incerteza e a interpretabilidade dos Sistemas Fuzzy e a habilidade de aprendizado provida pelas Redes Neurais [Pedrycz 1991]. Os Sistemas Neuro-Fuzzy têm sido amplamente utilizados para resolução de problemas reais de previsão [Abdollahzade et al. 2015], identificação de sistemas não-lineares [Cervantes et al. 2017] e classificação [Badnjevic et al. 2015, KV and Pillai 2017]. Existe um crescente interesse no desenvolvimento de novos algoritmos de aprendizado que possibilitem a construção autônoma do conjunto de regras de Sistemas Neuro-Fuzzy e o ajuste dos seus parâmetros utilizando técnicas de aprendizado de máquina [Shihabudheen and Pillai 2018].

Dentre as abordagens utilizadas para aprendizado em Sistemas Neuro-Fuzzy têm-se as técnicas de Computação Evolucionária que se destacam pela sua capacidade de adaptação e aprendizado, baseado em busca global [Herrera 2008]. A Computação Evolucionária pode ser empregada tanto para a extração do conhecimento e construção

da base regras de Sistemas Neuro-*Fuzzy* [Koshiyama et al. 2015] quanto para o ajuste dos seus parâmetros [Kaur et al. 2014]. Existem duas principais abordagens para criação de regras com técnicas de Computação Evolucionária, que são a de *Michigan* [Holland and Reitman 1977] e a de *Pittsburg* [Smith 1980]. A abordagem de *Michigan* define que cada indivíduo representa uma regra e a base de regras final é um conjunto de indivíduos. Neste caso a modelagem do indivíduo é mais simples, porém sua avaliação se torna mais complexa devido à dificuldade em definir a qualidade de uma regra isolada. Já na abordagem de *Pittsburg* cada indivíduo codifica um conjunto de regras, facilitando assim sua avaliação, visto que toda a base de regras estará representada em um único indivíduo. Por outro lado, o custo computacional e tempo de treinamento torna-se maior.

Diversos trabalhos exploraram a utilização de Computação Evolucionária para aprendizado em Sistemas *Fuzzy* e Neuro-*Fuzzy*. Em [Abadeh et al. 2011] um Algoritmo Genético (AG) foi utilizado para criar a base de regras de um modelo *Fuzzy* de Takagi-Sugeno com funções de pertinência triangulares e conjuntos *fuzzy* homogeneamente particionados. [Kaur et al. 2014] apresenta um sistema Neuro-*Fuzzy* com aprendizado por *Backpropagation* para diagnóstico de risco de hipertensão. O AG neste caso foi utilizado para inicializar a rede Neuro-*Fuzzy* com pesos e vieses otimizados. Em [Elhag et al. 2015] utiliza-se um Sistema *Fuzzy* Genético dentro de uma estrutura de aprendizagem pareada para detecção de vulnerabilidades em sistemas de informação no intuito de evitar ataques cibernéticos. O AG é utilizado para seleção das melhores regras no modelo de Mamdani e eliminar redundâncias.

No trabalho de [Soliman et al. 2018] é apresentado um Sistema Neuro-*Fuzzy* Híbrido chamado ANFIS-GA (*Adaptive Neuro-Fuzzy Inference System with Genetic Algorithm*). No ANFIS-GA o AG é utilizado para adaptar os parâmetros da rede ANFIS. Neste trabalho antecedentes e consequentes das regras *Fuzzy* são modelados como variáveis do AG. Utilizando Programação Genética (PG), [Koshiyama et al. 2015] apresenta um Sistema *Fuzzy* para problemas de classificação. O trabalho utiliza a Programação Genética Multi-Gene para criação e ajuste da base de regras de um Sistema *Fuzzy* Genético de Pittsburg. Já em [Mousavi et al. 2015] um modelo de PG multi-árvores é empregado para melhorar a acurácia de um Sistema *Fuzzy* de Takagi-Sugeno para mapeamento dinâmico de portfólios.

Neste contexto, este trabalho introduz dois algoritmos baseados em Programação Genética (PG) para construção da base de regras de uma rede Neo-*Fuzzy* Neuron (NFN) [Yamakawa et al. 1992]. A PG cria e ajusta as funções de pertinência que definem as regras e um método baseado no Gradiente ajusta os pesos da rede. Cada indivíduo representa uma base de regras completa para o modelo NFN.

As próximas seções estão organizadas da seguinte forma: a Seção 2 apresenta a estrutura da rede NFN e o seu processo de aprendizado; na Seção 3 são introduzidos os dois algoritmos propostos; a Seção 4 discorre sobre os experimentos computacionais e discute os resultados obtidos. Por fim, a Seção 5 traz as considerações finais, discutindo suas contribuições e perspectivas futuras.

2. Neo-*Fuzzy* Neuron

Esta seção detalha a rede Neo-*Fuzzy* Neuron (NFN) [Yamakawa et al. 1992] que será utilizada para implementação dos algoritmos apresentados na Seção 3. Neo-*Fuzzy* Neuron

é uma rede Neuro-Fuzzy composta por um conjunto de n modelos do tipo Takagi-Sugeno (TS) [Takagi and Sugeno 1985] de ordem zero, um para cada variável de entrada. Na NFN a saída de cada modelo individual y_{ti} é dada por um conjunto de m_j regras, e cada regra é representada por uma função de pertinência. A estrutura básica da rede NFN é ilustrada na Figura 1, na qual x_{ti} são as variáveis de entrada no instante t , q_{ij} são os pesos (parâmetros do consequentes) associados a cada função de pertinência (regra), y_{ti} é a saída individual de cada modelo e \hat{y}_t é a saída da rede.

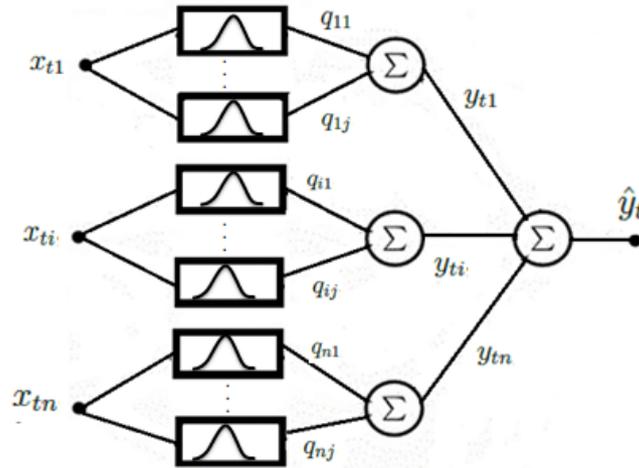


Figura 1. Estrutura da rede Neuro-Fuzzy Neo-Fuzzy-Neuron

Os modelos são desacoplados e as saídas individuais y_{ti} são obtidas pelo grau de ativação de cada regra $\mu_{A_{ij}}(x_i)$ multiplicado pelo seu respectivo peso q_{ij} , ponderados pelo somatório do grau de ativação de todas as regras, conforme apresentado na Equação (1):

$$y_{ti} = \frac{\sum_{j=1}^{m_i} \mu_{A_{ij}}(x_i) q_{ij}}{\sum_{j=1}^{m_i} \mu_{A_{ij}}(x_i)}, \quad (1)$$

onde i indexa as variáveis de entrada, j indexa as funções de pertinência e t é o instante de tempo. A saída do NFN é a soma dos modelos individuais e é obtida no instante t (\hat{y}_t) por:

$$\hat{y}_t = \sum_{i=1}^n y_{ti}. \quad (2)$$

2.1. Funções de Pertinência

As funções de pertinência utilizadas na NFN são Gaussianas. Outros tipos de funções de pertinência também podem ser utilizados, como funções Triangulares ou Trapezoidais [Bodyanskiy et al. 2003, Zaychenko and Gasanov 2012, Silva et al. 2014]. Uma função de pertinência Gaussiana é composta por dois parâmetros: centro (c) e espalhamento (s). A escolha do uso de funções de pertinência Gaussiana se justifica por ser o tipo de função mais adequada para representar dados incertos [Kreinovich et al. 1992]. A estrutura de uma função de pertinência Gaussiana é apresentada na Figura 2.

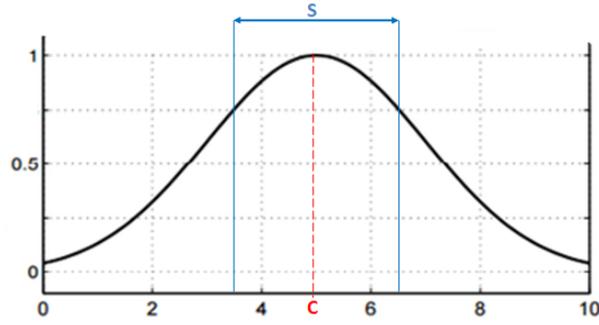


Figura 2. Representação de uma função de pertinência Gaussiana

2.2. Ajuste de Pesos da Rede NFN

O procedimento para adaptação dos pesos da rede NFN, isto é, o treinamento, é realizado por um método baseado no Gradiente. O aprendizado é supervisionado e tem o propósito de ajustar os parâmetros q_{ij} [Silva et al. 2014]. Este método atualiza os pesos a cada época de treinamento a partir de uma taxa de aprendizado (α) previamente definida. A fórmula para atualização dos pesos pode ser vista na Equação 3:

$$q_{ij} = q_{ij} - \alpha(e_t)(x_{ti})(d_{ij}), \quad (3)$$

onde α é a taxa de aprendizado, e_t é o erro de modelagem obtido por $y_t - \hat{y}_t$, x_{ti} é a amostra de dados no instante t para a variável i e d_{ij} é obtido por:

$$d_{ij} = \frac{\mu_{A_{ij}}}{\sum_{j=1}^{m_i} \mu_{A_{ij}}}, \quad (4)$$

onde m_i é o número de funções de pertinência para a variável i .

2.3. Algoritmo

O Algoritmo 1 sumariza a estimação da saída e o processo de atualização dos parâmetros da rede NFN.

3. GP-NFN - Genetic Programming-based Neo-Fuzzy Neuron

Esta seção introduz a abordagem proposta para construção da base de regras de uma rede Neo-Fuzzy Neuron. A metodologia de *Pittsburg* [Smith 1980], na qual um indivíduo representa uma base de regras completa, foi utilizada em conjunto com a Programação Genética Multi-Gene (PG-MG) [Ferreira 2001] para criação da base de regras da rede NFN e um método baseado no Gradiente para ajuste dos pesos da rede.

A PG-MG é uma variação da Programação Genética que considera que um indivíduo possui várias árvores em sua estrutura conectados por uma estrutura linear de genes. A saída final de cada indivíduo é a agregação das árvores ligadas a cada gene superior. No modelo apresentado neste trabalho, cada gene superior representa um neurônio (um modelo individual do NFN). Cada neurônio está associado a uma árvore de aridade m , sendo m o número de regras, ou seja, cada sub-árvore ligada ao nó central representa

Algoritmo 1: Algoritmo do NFN

```
Entrada  $x_t, y_t, n$ ;  
Saida  $\hat{y}_t$ ;  
for  $epoca = 1 : l$  do  
    for  $t = 1, 2, \dots$  do  
        Ler  $x_t, y_t$ ;  
        Calcular  $e_t$ ;  
        for  $j = 1 : m$  do  
            for  $i = 1 : n$  do  
                Calcular  $d_{ij}$ ;  
                Atualizar  $q_{ij}$ ;  
            end  
        end  
        Calcular  $\hat{y}_t$ ;  
    end  
end
```

uma regra (função de pertinência). O nó raiz é composto por uma função de agregação que retorna o resultado das m regras associadas àquele neurônio, conforme discutido na Seção 2 e mostrado na Figura 1.

O cruzamento na PG-MG pode ser realizado em alto nível ou baixo nível. O cruzamento em alto nível ocorre quando troca-se um gene superior de um pai com um gene superior do outro. Quando o cruzamento ocorre nas sub-árvores associadas ao gene superior, chama-se cruzamento de baixo nível. Neste trabalho foi utilizado cruzamento de baixo nível, combinando-se apenas as sub-árvores que representam as regras e não o neurônio inteiro. Informações complementares sobre cruzamento de alto e baixo nível são detalhadas no trabalho de [Koshiyama et al. 2015]. A reprodução é realizada gerando um filho exatamente igual ao pai (clone). Já a mutação altera apenas uma função (sub-árvore) de um gene sorteado, trocando-a por uma nova função aleatória. O método de seleção de pais é o método do torneio [Poli et al. 2008] e a geração da nova população é realizada substituindo-se toda a população atual pela nova população gerada.

Os indivíduos utilizam o mesmo conjunto de pesos (parâmetros do consequente) e estes são ajustados por um método baseado no Gradiente, similar ao descrito na Seção 2. Os indivíduos são avaliados pelo Erro Quadrático Médio (EQM) obtido por:

$$EQM = \frac{\sum_{t=1}^k (e_t^2)}{k}, \quad (5)$$

onde e_t é o erro (diferença entre a saída esperada y_t e a saída do modelo \hat{y}_t) no instante t e k é o número de amostras.

Foram propostas duas versões do algoritmo, chamadas respectivamente de GP-NFN-I e GP-NFN-II. A Seção 3.1 detalha o algoritmo do GP-NFN-I e a Seção 3.2 o do GP-NFN-II. Nos dois algoritmos tanto as funções quanto os pesos e parâmetros iniciais são gerados aleatoriamente.

3.1. GP-NFN-I

No GP-NFN-I a cada geração da PG são executados l épocas de treinamento para ajuste dos pesos. O número de épocas de treinamento é um parâmetro de entrada do algoritmo e experimentos iniciais mostram que esse parâmetro pode ser definido entre 1 e 10. Foram observados resultados melhores com $l = 5$, onde a Programação Genética apresenta convergência mais lenta, mas obtém uma menor taxa de erro. Utilizando $l = 10$ observou-se uma convergência prematura da Programação Genética em um número considerável de testes. O algoritmo pode ser resumido nas seguintes etapas:

1. Gera parâmetros aleatórios (pesos da rede).
2. Gera população inicial (indivíduos contendo funções aleatórias).
3. Avalia a população calculando o erro de modelagem de cada indivíduo.
4. Seleciona o melhor indivíduo (indivíduo com menor erro).
5. Aplica o melhor indivíduo ao NFN, atribuindo as funções do melhor indivíduo ao modelo.
6. Executa l passos do Gradiente para ajuste dos pesos.
7. Aplica cruzamento, mutação e reprodução para geração da nova população.
8. Substitui a população atual pela nova população.
9. Volta ao passo 3

Estas etapas são repetidas até atingir o número de gerações definido (critério de parada). O fluxo de execução deste algoritmo pode ser visto na Figura 3.

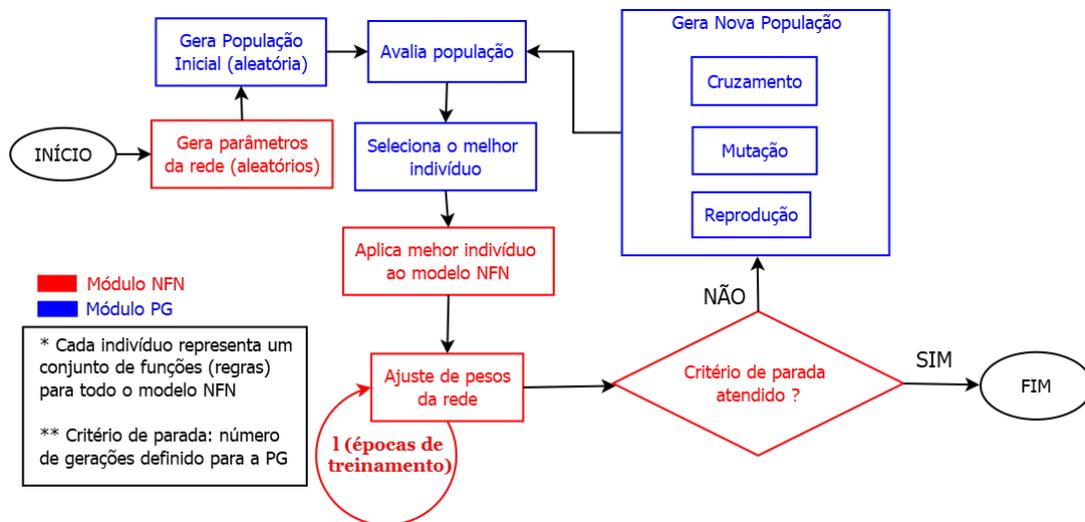


Figura 3. Fluxograma do algoritmo GP-NFN-I

3.2. GP-NFN-II

Na segunda versão (GP-NFN-II), assim como na primeira (GP-NFN-I), os pesos e parâmetros são gerados aleatoriamente como primeiro passo do algoritmo. No entanto, no GP-NFN-II os pesos e parâmetros somente são alterados após o término de todas as gerações da PG. Em outras palavras, os pesos e parâmetros do consequente das regras serão otimizados pelo método do Gradiente somente para o melhor indivíduo definido pela Programação Genética após a última geração. O algoritmo segue as seguintes etapas:

1. Gera parâmetros aleatórios (pesos da rede).
2. Gera população inicial (indivíduos contendo funções aleatórias).
3. Avalia a população calculando o erro de modelagem de cada indivíduo.
4. Aplica cruzamento, mutação e reprodução para geração da nova população.
5. Substitui a população atual pela nova população.
6. Repete passos 3 a 5 até atingir o número de gerações estabelecido (critério de parada).
7. Seleciona o melhor indivíduo (indivíduo com menor erro).
8. Aplica o melhor indivíduo ao NFN, atribuindo as funções do melhor indivíduo ao modelo.
9. Executa l épocas de treinamento para ajuste dos pesos.

Este algoritmo é ilustrado na Figura 4.

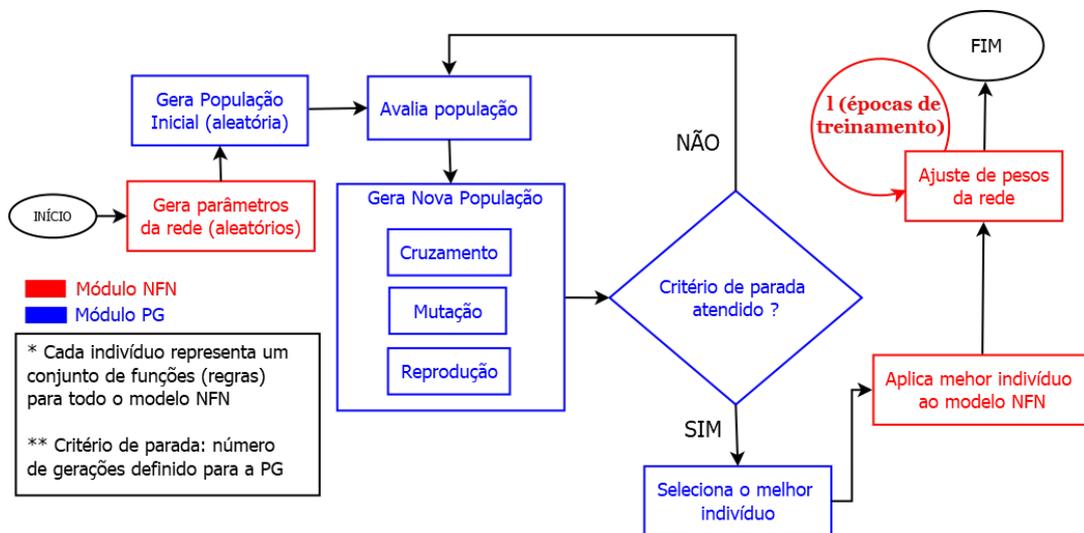


Figura 4. Fluxograma do algoritmo GP-NFN-II

4. Experimentos Computacionais

Os algoritmos propostos (GP-NFN-I e GP-NFN-II) foram avaliados e comparados com modelos alternativos em problemas de previsão e identificação de sistemas não-lineares. Os modelos alternativos utilizados nos experimentos foram o ANFIS (*Adaptive Neuro-Fuzzy Inference System*) com aprendizado por Mínimos Quadrados e Gradiente Descendente com *Backpropagation*, uma Rede Neural MLP (*Multilayer Perceptron*) com aprendizado pelo método do Gradiente Descendente com *Backpropagation* e uma rede Neo-Fuzzy-Neuron tradicional com aprendizado pelo método do Gradiente.

Para cada experimento o conjunto de dados foi dividido em três subconjuntos, um para treinamento com 60% das amostras, um para validação com 20% e outro para teste com 20% das amostras. Cada experimento foi executado 10 vezes e o melhor resultado foi utilizado para comparação. O desempenho dos sistemas foi avaliado pelo *RMSE* (*Root Mean Square Error*) (6):

$$RMSE = \frac{1}{k} \left(\sum_{t=1}^k (y_t - \hat{y}_t) \right)^{\frac{1}{2}}, \quad (6)$$

onde k é o número de amostra do conjunto de teste, \hat{y}_t é a saída estimada, y_t é a saída desejada. Os parâmetros dos modelos foram definidos como descrito a seguir para os dois experimentos realizados:

- ANFIS: épocas de treinamento = 500; funções de pertinência inicial = 2; tipo das funções pertinência = Gaussiana; $\alpha = 0.01$.
- GP-NFN-I: tamanho da população = 50; gerações = 300; épocas de treinamento = 5; taxa de cruzamento = 0.9; taxa de mutação = 0.08; $\alpha = 0.01$; funções de pertinência inicial para cada indivíduo = 5; tipo das funções de pertinência = Gaussiana.
- GP-NFN-II: tamanho da população = 50; gerações = 300; épocas de treinamento = 500; taxa de cruzamento = 0.9; taxa de mutação = 0.08; $\alpha = 0.01$; funções de pertinência inicial para cada indivíduo = 5; tipo das funções de pertinência = Gaussiana.
- NFN: épocas de treinamento = 500; $\alpha = 0.01$; funções de pertinência = 5; tipo das funções de pertinência = Gaussiana.
- RNA MLP: camadas ocultas = 2; neurônios por camada = 10; épocas de treinamento = 500; $\alpha = 0.01$; função de ativação tangente hiperbólica.

4.1. Previsão de Vazão Semanal

Nesta seção, avaliam-se os modelos na previsão da vazão média semanal de uma usina hidrelétrica de grande porte (Usina de Sobradinho). O objetivo do modelo é prever a vazão da semana seguinte baseado nas semanas anteriores, conforme Equação (7) [Lemos et al. 2011, Silva et al. 2014].

$$y_t = f(y_{t-1}, y_{t-2}, y_{t-3}). \quad (7)$$

O conjunto de dados se refere ao período de 1.931 a 2.000 e é composto por 3.707 amostras com 3 variáveis de entrada e 1 de saída. Destas, 2.224 foram utilizadas para treinar os modelos, 741 para validar e 742 para avaliar o desempenho dos modelos pelo RMSE. Assim como em [Lemos et al. 2011, Silva et al. 2014] os dados disponibilizados estão normalizados entre 0 e 1 para manter a privacidade.

Como pode ser visto na Tabela 1 o GP-NFN-I obteve o melhor desempenho, seguido pelo NFN e pelo GP-NFN-II. Os resultados obtidos pelo NFN e GP-NFN-II são comparáveis e superam os dos demais modelos. A Figura 5 ilustra os valores desejados e os estimados pelo GP-NFN-I para os dados de teste.

4.2. Identificação de Sistemas Não-Lineares

Este experimento visa analisar o comportamento dos algoritmos propostos na identificação de processo não linear descrito por (8):

$$y_t = \frac{y_{t-1}y_{t-2}y_{t-3}(y_{t-3} - y_{t-1})\beta_{t-1} + \beta_t}{1 + y_{t-3}^2 + y_{t-2}^2}, \quad (8)$$

onde $y_t = 0$ e $\beta_t = 0$ para $t \leq 3$, β_t é definido pela Equação (9) para $3 < t \leq 500$ e pela Equação (10) para $t > 500$.

Modelo	RMSE
GP-NFN-I	0,0235450
NFN	0,0252245
GP-NFN-II	0,0256137
ANFIS	0,0349183
MLP	0,1375477

Tabela 1. Desempenho na previsão de vazão semanal

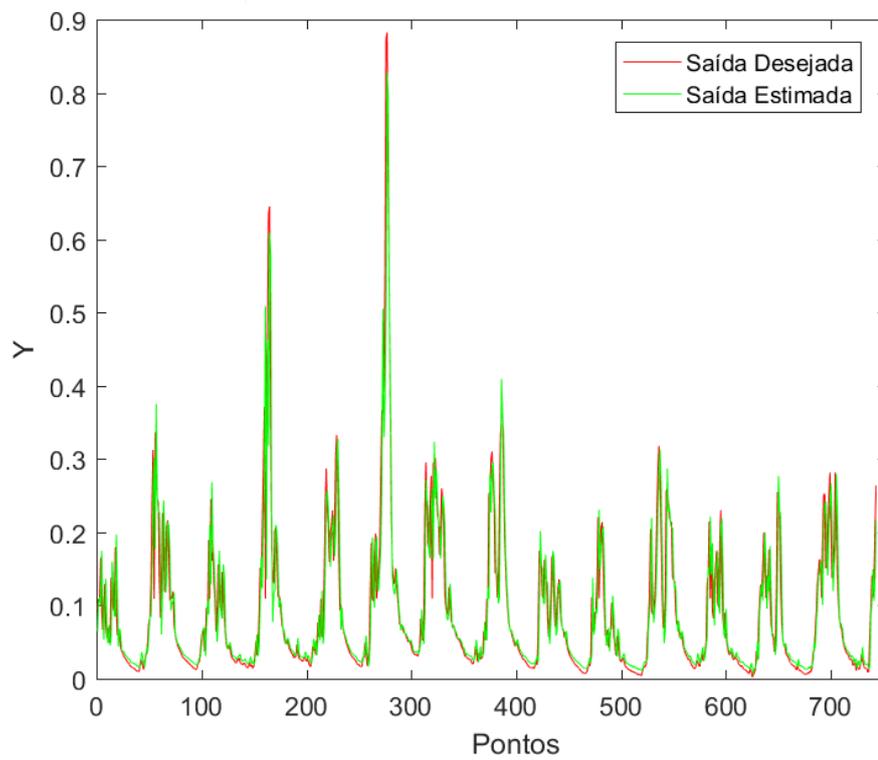


Figura 5. Previsão de vazão semanal pelo GP-NFN-I

$$\beta_t = \sin\left(\frac{2\pi t}{250}\right), \quad (9)$$

$$\beta_t = \sin\left(\frac{2\pi t}{250}\right) + 0.4 \sin\left(\frac{2\pi t}{25}\right). \quad (10)$$

O objetivo é prever a saída corrente utilizando-se a entrada com atraso e as saídas. O modelo para este conjunto de dados é definido por:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-5}), \quad (11)$$

onde \hat{y}_t é a saída. Para o experimento foram criadas 10.000 amostras, sendo 4.000 utilizadas para treinar os modelos, 2.000 para validar e 2.000 para avaliar o seu desempenho pelo RMSE.

Os resultados obtidos pelos modelos na identificação do sistema não-linear são apresentados na Tabela 2. Observa-se que os melhores resultados foram obtidos pela rede ANFIS. Dentre os algoritmos baseados no NFN o GP-NFN-I obteve o melhor resultado, seguido pelo NFN tradicional. O modelo GP-NFN-II se mostrou inferior aos demais baseados no NFN. A Figura 6 ilustra os valores desejados e os estimados pelo GP-NFN-I para os dados de avaliação.

Modelo	RMSE
ANFIS	0,00000845
GP-NFN-I	0,00055080
NFN	0,00067363
GP-NFN-II	0,00077090
MLP	0,03366441

Tabela 2. Desempenho em identificação de sistemas não-lineares

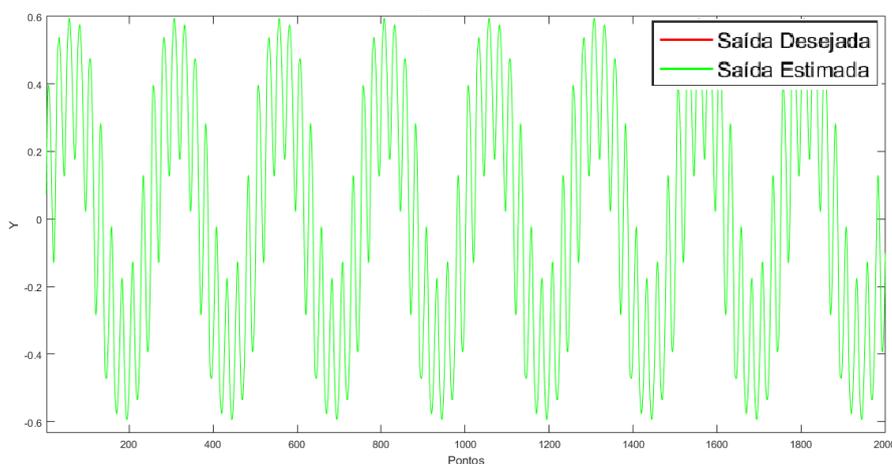


Figura 6. Identificação de sistemas não-lineares pelo GP-NFN-I

5. Conclusão

Este trabalho introduziu dois algoritmos baseados em Programação Genética Multi-Gene para construção da base de regras em redes Neo-Fuzzy Neuron. As abordagens utilizam a PG para criação e ajuste da base de regras e um método baseado no Gradiente para o ajuste de pesos.

Experimentos computacionais em problemas de previsão e identificação de sistemas sugerem que a abordagem seja promissora. Os modelos propostos se demonstram eficientes quando comparados com modelos alternativos, em especial o GP-NFN-I. Isso

demonstra que fazer o ajuste de pesos e funções simultaneamente pode trazer ganhos ao algoritmo, além de, no caso da abordagem apresentada, permitir que a PG e o método do Gradiente cooperem entre si. Mesmo nos casos em que o modelo NFN se demonstra inferior a outro modelo, o GP-NFN-I ainda se demonstra superior ao modelo NFN tradicional.

Trabalhos futuros devem explorar a utilização da Programação Genética também para ajuste dos pesos, modelando as funções com seus pesos como indivíduos da PG. Além disso pode-se utilizar métodos de Programação Genética Semântica, que visa mitigar alguns problemas que podem ser encontrados pela PG tradicional. Também pode-se considerar utilizar funções de pertinência triangulares na rede NFN.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

- Abadeh, M. S., Mohamadi, H., and Habibi, J. (2011). Design and analysis of genetic fuzzy systems for intrusion detection in computer networks. *Expert Systems with Applications*, 38(6):7067–7075.
- Abdollahzade, M., Miranian, A., Hassani, H., and Iranmanesh, H. (2015). A new hybrid enhanced local linear neuro-fuzzy model based on the optimized singular spectrum analysis and its application for nonlinear and chaotic time series forecasting. *Inf. Sci.*, 295(C):107–125.
- Badnjevic, A., Cifrek, M., Koruga, D., and Osmankovic, D. (2015). Neuro-fuzzy classification of asthma and chronic obstructive pulmonary disease. *BMC Medical Informatics and Decision Making*, 15(3):S1.
- Bodyanskiy, Y., Kokshenev, I., and Kolodyazhniy, V. (2003). An adaptive learning algorithm for a neo fuzzy neuron. In *EUSFLAT Conf.*, pages 375–379. Citeseer.
- Cervantes, J., Yu, W., Salazar, S., and Chairez, I. (2017). Takagi–sugeno dynamic neuro-fuzzy controller of uncertain nonlinear systems. *IEEE Transactions on Fuzzy Systems*, 25(6):1601–1615.
- Elhag, S., Fernández, A., Bawakid, A., Alshomrani, S., and Herrera, F. (2015). On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. *Expert Syst. Appl.*, 42(1):193–202.
- Ferreira, C. (2001). Gene expression programming: a new adaptive algorithm for solving problems. *Complex Systems*, 13(2):87–129. cite arxiv:cs/0102027Comment: 22 pages, 17 figures.
- Herrera, F. (2008). Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evolutionary Intelligence*, 1(1):27–46.
- Holland, J. H. and Reitman, J. S. (1977). Cognitive systems based on adaptive algorithms. Number 63, pages 49–49. ACM, New York, NY, USA.

- Kaur, A., Bhardwaj, A., and Been, U. A. H. (2014). Genetic neuro fuzzy system for hypertension diagnosis. *International Journal of Computer Science and Information Technologies*, 5(4):4986–4989.
- Koshiyama, A. S., Vellasco, M. M., and Tanscheit, R. (2015). Gpfis-class: A genetic fuzzy system based on genetic programming for classification problems. *Appl. Soft Comput.*, 37(C):561–571.
- Kreinovich, V., Quintana, C., and Reznik, L. (1992). Gaussian membership functions are most adequate in representing uncertainty in measurements. In *Proceedings of NAFIPS*, pages 15–17.
- KV, S. and Pillai, G. (2017). Regularized extreme learning adaptive neuro-fuzzy algorithm for regression and classification. *Know.-Based Syst.*, 127(C):100–113.
- Lemos, A., Caminhas, W., and Gomide, F. (2011). Fuzzy evolving linear regression trees. *Evolving Systems*, 2(1):1–14.
- Mousavi, S., Esfahanipour, A., and Zarandi, M. H. F. (2015). Mgp-intactsky: Multi-tree genetic programming-based learning of interpretable and accurate tsk systems for dynamic portfolio trading. *Applied Soft Computing*, 34:449 – 462.
- Pedrycz, W. (1991). Neurocomputations in relational systems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(3):289–297.
- Poli, R., Langdon, W. B., and McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd.
- Shihabudheen, K. and Pillai, G. (2018). Recent advances in neuro-fuzzy system. *Know.-Based Syst.*, 152(C):136–162.
- Silva, A. M., Caminhas, W., Lemos, A., and Gomide, F. (2014). A fast learning algorithm for evolving neo-fuzzy neuron. *Appl. Soft Comput.*, 14:194–209.
- Smith, S. F. (1980). *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, Pittsburgh, PA, USA. AAI8112638.
- Soliman, M. A., Hasanien, H. M., Azazi, H. Z., El-kholy, E. E., and Mahmoud, S. A. (2018). Hybrid anfis-ga-based control scheme for performance enhancement of a grid-connected wind generator. *IET Renewable Power Generation*, 12(7):832–843.
- Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(1):116–132.
- Yamakawa, T., Uchino, E., Miki, T., and Kusabagi, H. (1992). A neo fuzzy neuron and its applications to system identification and predictions to system behavior. In *Proceedings of the International Conference on Fuzzy Logic and Neural Networks*, pages 477–484. IEEE.
- Zaychenko, Y. and Gasanov, A. (2012). Investigations of cascade neo-fuzzy neural networks in the problem of forecasting at the stock exchange. In *2012 IV International Conference "Problems of Cybernetics and Informatics"(PCI)*, pages 1–3.