Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Local Linear Model Tree with Optimized Structure [⋆]

Xiaoyan Hu [*] **Yu Gong** [*] **Dezong Zhao** [**] **Wen Gu** [**]

[*] *Department of Wolfson School, Loughborough University, Loughborough, LE11 3TU, U.K. (e-mail: x.hu4@lboro.ac.uk, y.gong@lboro.ac.uk).*
[**] *Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, LE11 3TU, U.K. (e-mail: d.zhao@lboro.ac.uk, w.gu@lboro.ac.uk)*

Abstract: This paper investigates the local linear model tree (LOLIMOT) with optimized structure. The performance of the LOLIMOT model depends on how the neurons are constructed. In the typical LOLIMOT model, the number of neurons is initially set as one and starts to increase by repeatedly splitting an existing neuron into two equal ones until the required performance is achieved. Because the equal split of a neuron is not optimal, a large model size is often necessary for required performance, leading to high complexity and strong overfitting. In this paper, we propose a gradient-decent-search-based algorithm to optimally split an existing neuron into two new ones. Based on both numerical data and simulated engine data, through the evaluation of optimized structure, the effectiveness of proposed method has been verified.

Keywords: LOLIMOT, gradient descent search, optimized structure.

1. INTRODUCTION

The system modelling is crucial in many applications such as automation factory, vehicular engineering, aeroplane and telecommunications (Lee and Ouyang, 2003). While linear modelling is widely used due to its simplicity, most practical systems are non-linear (Kukolj and Levi, 2004). Various nonlinear modelling approaches have been proposed including the Volterra kernel (Kashiwagi and Rong, 2002), neural networks (Li et al., 2018) and Bayesian networks (Pan et al., 2015).

Of particular interest to this paper is the neural fuzzy modelling which brings the low-level learning and computational power of neural networks into fuzzy systems (Yeh and Su, 2017; Juang and Lin, 1998).The fuzzy set theory is originally used to deal with uncertain information (Takagi and Sugeno, 1985), which is achieved by formulating implicit knowledge of the underlying process into a set of linguistic variables and fuzzy rules or extracting the fuzzy rules from data(Jamab and Araabi, 2006). A number of neural fuzzy models have been proposed including the interval type-2 radial basis function neural network (Rubio-Solis and Panoutsos, 2015), four layers network featured Takagi-Sugeno-Kang architecture with Gaussian kernel (Pratama et al., 2013) and recurrent fuzzy neural network (Lin et al., 2013). In a typical neural fuzzy modelling, both the learning parameter and the modelling structure need to be identified. A non appropriately structured neural fuzzy model can have very large model size (Lee and Ouyang, 2003). In (Jamab and Araabi, 2006), the neural fuzzy model can be self structured through merging and splitting neurons.

In this paper, we focus on a popular neural fuzzy model, the local linear model tree (LOLIMOT) (Nelles and Isermann, 1996; Nelles, 2006). The LOLIMOT model consists of a number of neurons, where each neuron includes a Gaussian kernel and a local linear structure. The neurons are constructed in a growing manner that the model starts with a single neuron and repeatedly generates new neurons by splitting a previous node into two equal nodes until the performance is satisfied. While an equal split of an existing neuron provides a simple solution to generate new neurons, it may not match the input data statistics. As a result, many neurons are required to satisfy the performance. This leads to large model size, which results in overfitting and high complexity.

In this paper, we propose a new gradient decent search algorithm to optimally split an existing neuron. Compared to the traditional LOLIMOT algorithm, the proposed scheme is able to achieve superior performance with small model size, making it attractive in practice.

The rest of the paper is organized as follows: Section 2 introduces the LOLIMOT model; Section 3 proposes the gradient decent search algorithm to optimally split an neuron; Section 4 verifies the proposed algorithm with both numerical data and simulated engine data, and finally Section 5 concludes the paper.

2. LOCAL LINEAR MODEL TREES

The structure of LOLIMOT with K neurons is shown in Fig. 1.

[⋆] This work is supported by the Engineering and Physical Sciences Research Council of U.K.under the EPSRC-UKRI Innovation Fellowship scheme (EP/S001956/1).

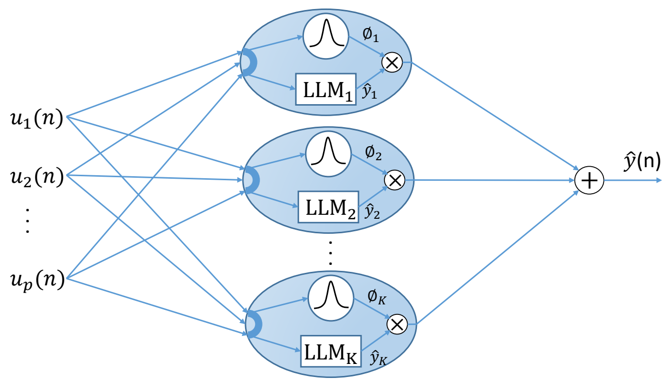

Fig. 1. The LOLIMOT structure with K neurons

Each neuron consists of a local linear model 'LLM_k' and a Gaussian kernel. The data input vector at time n is $\boldsymbol{u}(n) = \{u_1(n), \cdots, u_p(n)\}^{\mathrm{T}}$, where p is the dimension. The output of the 'k-th' Gaussian kernel, i.e. the validity function is

$$\phi_k(n) = \frac{\mu_k(\boldsymbol{u}(n))}{\sum_{j=1}^{K} \mu_j(\boldsymbol{u}(n))} \tag{1}$$

with

$$\begin{aligned}\mu_k(\boldsymbol{u}) =&\exp\left(-\frac{1}{2}\cdot\frac{(u_1(n)-c_{k1})^2}{\sigma_{k1}^2}\right)\cdot\ldots\\ &\cdot\exp\left(-\frac{1}{2}\cdot\frac{(u_p(n)-c_{kp})^2}{\sigma_{kp}^2}\right)\end{aligned} \tag{2}$$

where c_{kp} and σ_{kp} is the center and standard deviation of 'k-th' Gaussian kernel in the 'p-th' dimension respectively.

The output of the local linear model LLM is given by:

$$\hat{y_k}(n) = \omega_{k0} + \omega_{k1}\cdot u_1(n) + \omega_{k2}\cdot u_2(n) + \ldots + \omega_{kp}\cdot u_p(n) \tag{3}$$

The output of the overall LOLIMOT system is the sum from all neurons:

$$\hat{y}(n) = \sum_{k=1}^{K} \hat{y_k}(n)\cdot\phi_k(n) \tag{4}$$

Assuming there are N snapshots of the input, then $\boldsymbol{\omega}_k$ indicates the local linear parameters for the k-th node which is obtained by the weighted least square (WLS) as:

$$\boldsymbol{\omega}_k = (\boldsymbol{U}_k^T \boldsymbol{Q}_k \boldsymbol{U}_k)^{-1} \boldsymbol{U}_k^T \boldsymbol{Q}_k \boldsymbol{y} \tag{5}$$

where

$$\boldsymbol{U}_k = \begin{pmatrix} 1 & u_1(1) & u_2(1) & \ldots & u_p(1) \\ 1 & u_2(2) & u_2(2) & \ldots & u_p(2) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & u_1(N) & u_2(N) & \ldots & u_p(N) \end{pmatrix} \tag{6}$$

and

$$\boldsymbol{Q}_k = \begin{pmatrix} \phi_k(1) & 0 & \ldots & 0 \\ 0 & \phi_k(2) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \phi_k(N) \end{pmatrix} \tag{7}$$

and $\boldsymbol{y} = [y(1), y(2), ..., y(N)]^{\mathrm{T}}$ is the vector of measured model output.

The input space is split into multiple hyper rectangle partitions, and each partition corresponds to one neuron. The global loss function G is introduced to show the modelling performance of the overall system as:

$$G = \sum_{n=1}^{N} [y(n) - \hat{y}(n)]^2 \tag{8}$$

On the other hand, the local loss function e_k shows the modelling performance of the k-th neuron as :

$$e_k = \sum_{n=1}^{N} [y(n) - \hat{y}(n)]^2 \phi_k(\boldsymbol{u}(n)) \tag{9}$$

The LOLIMOT starts with a single neuron and grows the number of neurons by splitting an existing neuron into two parts until the global loss function G is below a certain threshold. The partition of a neuron is split according to the following rules:

(1) Choose the partition for the neuron with the largest local loss function e_k. This neuron is called the 'worst' neuron.
(2) Calculate the global loss function for every possible splitting dimension (along which the partition for the selected neuron is equally split), respectively.
(3) Split the chosen neuron into two neurons along the dimensions with the best would-be performance after the splitting.
(4) Repeat the above procedures until the maximum model size (i.e. the number of neurons) is reached or the global loss function G is below a certain threshold.

3. GRADIENT DESCENT SEARCH ALGORITHM

3.1 Split an existing partition

Equally splitting the 'worst' existing neuron into two in the LOLIMOT is often not optimal, leads to a large model size (i.e. large number of neurons). We propose to optimally split the neuron using the gradient decent search approach.

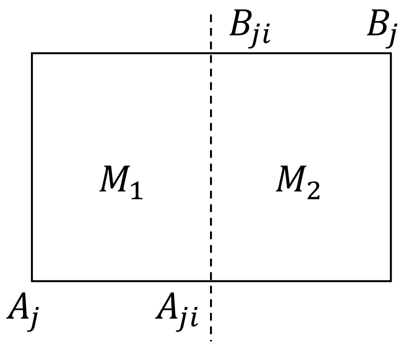

Fig. 2. On the j-th dimension of hyper rectangle with point coordinates

For illustration, Fig. 2 shows the the hyper rectangle partition of a neuron on j-th dimension, where A_j and B_j are the corner coordinates of the partition which are given by

$$\begin{aligned} A_j &= (a_1, ..., a_j, ..., a_p) \\ B_j &= (b_1, ..., b_j, ..., b_p) \end{aligned} \tag{10}$$

respectively. We assume that the split dimension is along the dotted line connecting A_{ji} and B_{ji} with coordinates as:

$$\begin{aligned} A_{ji} &= (a_1, ..., [(b_j - a_j)\cdot X + a_j], ..., a_p) \\ B_{ji} &= (b_1, ..., [(b_j - a_j)\cdot X + a_j], ..., b_p) \end{aligned} \tag{11}$$

respectively, where X is the split ratio parameter to control how the existing partitions is split into two parts. The original LOLIMOT simply sets $X = 0.5$ so that an existing partition is equally split into two.

Assume the new partitions after the split are M_1 and M_2, which corresponds to two new Gaussian kernels, respectively. The centers and standard deviations of the two new Gaussian kernels are given by:

$$\begin{aligned} c_1 &= (A_j + B_{ji})\cdot 0.5 \\ c_2 &= (A_{ji} + B_j)\cdot 0.5 \\ \sigma_1 &= (B_{ji} - A_j)\cdot \eta \\ \sigma_2 &= (B_j - A_{ji})\cdot \eta \end{aligned} \tag{12}$$

where the index '1' and '2' indicate the parameters for M_1 and M_2 respectively, η is the smoothness parameter. In the following, the procedure of optimizing X is given to achieve the best splitting performance.

3.2 The optimal partition with the gradient decent search

Based on the gradient descent search rule, the split ratio X can be updated as:

$$X_{t+1} = X_t - \beta\cdot\nabla J(X_t) \tag{13}$$

where X_t is the split ratio at time t, β is the step size parameter, and $\nabla J(X_t)$ is gradient of the model global loss function value with respective to X_t :

$$\begin{aligned} \nabla J(X_t) =& \frac{\partial(G)}{\partial X_t} \\ =& 2\cdot\sum_{n=1}^{N}\cdot(y(n) - \hat{y}(n))\cdot\frac{\partial[y(n) - \hat{y}(n)]}{\partial X_t} \end{aligned} \tag{14}$$

For better expression, the time index n is omitted in the rest of this subsection. First we have:

$$\begin{aligned} \frac{\partial(y-\hat{y})}{\partial X_t} =& \frac{\partial(y - \hat{y}_1 - \hat{y}_2 - ... - \hat{y}_{m_1} - \hat{y}_{m_2} - ... - \hat{y}_K)}{\partial X_t} \\ =& -\frac{\partial\hat{y}_{m_1}}{\partial X_t} - \frac{\partial\hat{y}_{m_2}}{\partial X_t} \\ =& -\frac{\partial\phi_{m_1}}{\partial X_t}\cdot \mathrm{LLM}_{m_1} - \frac{\partial\phi_{m_2}}{\partial X_t}\cdot \mathrm{LLM}_{m_2} \end{aligned} \tag{15}$$

where LLM_{m_1} and LLM_{m_2} are the local linear parts, ϕ_{m_1} and ϕ_{m_2} are normalized Gaussian kernals, $\hat{y}_{m_1}$ and $\hat{y}_{m_2}$ are the new nodes output for partition M_1 and M_2, respectively. From (12), we have

$$\begin{aligned} c_{m_1} &= (A_j + B_{ji})\cdot 0.5 \\ &= \frac{(b_j - a_j)\cdot X_t + 2a_j}{2} \\ \sigma_{m_1} &= (B_{ji} - A_j)\cdot\eta \\ &= (b_j - a_j)\cdot X_t\cdot\eta \\ c_{m_2} &= (A_{ji} + B_j)\cdot 0.5 \\ &= \frac{(b_j - a_j)\cdot X_t + a_j + b_j}{2} \\ \sigma_{m_2} &= (B_j - A_{ji})\cdot\eta \\ &= [b_j - (b_j - a_j)\cdot X_t - a_j]\cdot\eta \end{aligned} \tag{16}$$

Then we have:

$$\begin{aligned} F_{m_1} &= \frac{\partial\phi_{m_1}}{\partial X_t} = \frac{\partial\frac{\mu_{m_1}}{\mu_{\text{sum}}}}{\partial X_t} \\ &= \frac{\frac{\partial\mu_{m_1}}{\partial X_t}\cdot(\mu_{\text{sum}}) - \mu_{m_1}\cdot\frac{\partial(\mu_{m_1}+\mu_{m_2})}{\partial X_t}}{(\mu_{\text{sum}})^2} \\ &= \frac{\frac{\partial\mu_{m_1}}{\partial X_t}\cdot(\mu_{\text{sum}}) - \mu_{m_1}\cdot\frac{\partial(\mu_{m_1})}{\partial X_t} - \mu_{m_1}\cdot\frac{\partial(\mu_{m_2})}{\partial X_t}}{(\mu_{\text{sum}})^2} \end{aligned} \tag{17}$$

$$\begin{aligned} F_{m_2} &= \frac{\partial\phi_{m_2}}{\partial X_t} = \frac{\partial\frac{\mu_{m_2}}{\mu_{\text{sum}}}}{\partial X_t} \\ &= \frac{\frac{\partial\mu_{m_2}}{\partial X_t}\cdot(\mu_{\text{sum}}) - \mu_{m_2}\cdot\frac{\partial(\mu_{m_1}+\mu_{m_2})}{\partial X_t}}{(\mu_{\text{sum}})^2} \\ &= \frac{\frac{\partial\mu_{m_2}}{\partial X_t}\cdot(\mu_{\text{sum}}) - \mu_{m_2}\cdot\frac{\partial(\mu_{m_1})}{\partial X_t} - \mu_{m_2}\cdot\frac{\partial(\mu_{m_2})}{\partial X_t}}{(\mu_{\text{sum}})^2} \end{aligned} \tag{18}$$

$$\mu_{\text{sum}} = \mu_1 + \mu_2 + ... + \mu_{m_1} + \mu_{m_2} + ... + \mu_K \tag{19}$$

where μ_{m1} and μ_{m2} are the Gaussian kernel outputs for M_1 and M_2 respectively. From (2), we have

$$\begin{aligned} P_{m_1} &= \frac{\partial\mu_{m_1}}{\partial X_t} \\ &= \mu_{m_1}\cdot(1/\eta)^2\cdot \\ &\quad \frac{(a_j - u_j)\cdot[a_j + 0.5X_t\cdot(b_j - a_j) - u_j]}{(b_j - a_j)^2\cdot X_t^3} \\ P_{m_2} &= \frac{\partial\mu_{m_2}}{\partial X_t} \\ &= \frac{1}{2}\mu_{m_2}\cdot(1/\eta)^2\cdot \\ &\quad \frac{(a_j - b_j)(b_j - u_j)[a_j + b_j - 2u_j - (a_j - b_j)X_t]}{[b_j - a_j - (b_j - a_j)X_t]^3} \end{aligned} \tag{20}$$

The algorithm is listed in Algorithm 1, where G_{th} is the threshold for the global error, and K_{max} is the maximum model size.

Algorithm 1 Gradient Descent Search Based on LOLIMOT Algorithm

```
Initialization;
Generate the first neuron;
Calculate the global error for 1st neuron: $G$
while $G > G_{th}$ or $K < K_{max}$ do
    Select the worst neuron with the largest $e_k$ as in (9);
    Obtain the hyper rectangle coordinates as in (10);
    for every input dimension $(j = 1 : p)$ do
        Set initial split ratio $X(0) = 0.5$;
        for every gradient search iteration $(i = 1 : Z)$ do
            Calculate new coordinates $A_{ji}$ and $B_{ji}$ as in (11);
            Calculate new Gaussian kernel parameters $c_{m1}$, $c_{m2}$, $\sigma_{m1}$, $\sigma_{m2}$ as in (16);
            Determine the validity for two new LLMs $\phi_{m1}$ and $\phi_{m2}$ as in (1) and (2);
            Calculate the new LLMs parameters $\omega_{m1}$ and $\omega_{m2}$ as in (5);
            Calculate $G_j(i)$, $e_{kj}(i)$ as in (8) and (9);
            Determine the gradient $\nabla J_j(i)$ of $G_j(i)$ using (14) - (20);
            Update the split ratio $X$ at $i = i+1$ using (13);
        end
    end
    Split the neuron along the dimension with the best would-be performance from above;
end
```

4. SIMULATION AND ANALYSIS

In this section, the proposed algorithm is verified with the numerical data and simulated engine data. The modelling performance is measured by the normalized root mean squared error (NRMSE) which is given by:

$$\text{NRMSE} = \frac{\| \boldsymbol{y} - \hat{\boldsymbol{y}} \|}{\| \boldsymbol{y} - \text{E}(\boldsymbol{y}) \|} \tag{21}$$

where $\boldsymbol{y} = [y(1), y(2), ..., y(N)]$, $\hat{\boldsymbol{y}} = [\hat{y}(1), \hat{y}(2), ..., \hat{y}(N)]$, $\| \,.\, \|$ is the 2-norm and E(.) is the expectation.

In the proposed gradient decent search scheme, the step size is 0.003 and the iteration time is 150. After trying different step size and iteration time, they are chosen with showing better result performance.

Case 1: Numerical data example

In this case, the output of an non-linear system is given by:

$$\begin{aligned} y(t) &= 0.72y(t-1) + \sin[y(t-2)u(t-2)] \\ &+ \cos[u(t-3)]^2 + 0.2\sin[u(t-2)]^3 \\ &+ \text{atan}[2u(t-1) + 3u(t-3)] \end{aligned} \tag{22}$$

where the system input $u(t)$ is the amplitude modulated pseudo random binary signal (AMPRBS). There are 1000 input data $x(t)$ are generated, among which the first 750 data are used for training and the rest are used for testing.

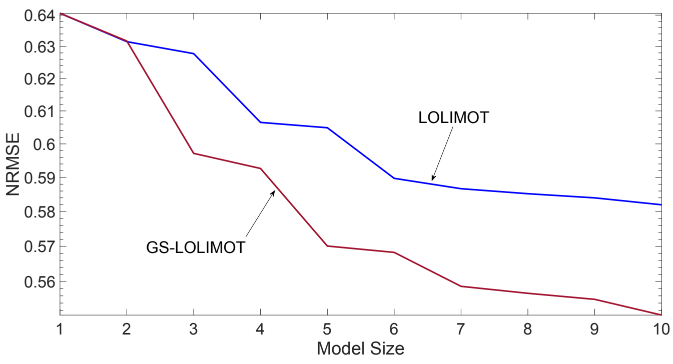

Fig. 3. Case 1 - Numerical data: The NRMSE performance for the training data

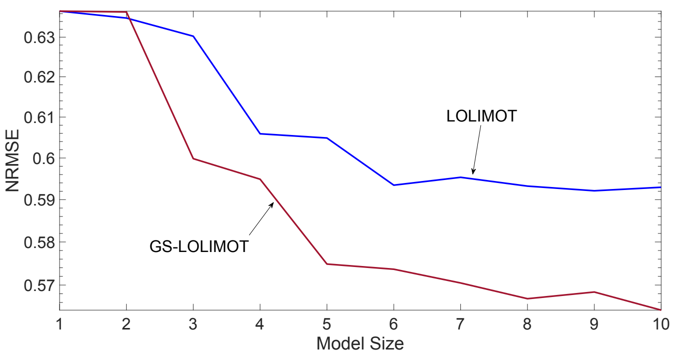

Fig. 4. Case 1 - Numerical data: The NRMSE performance for the testing data

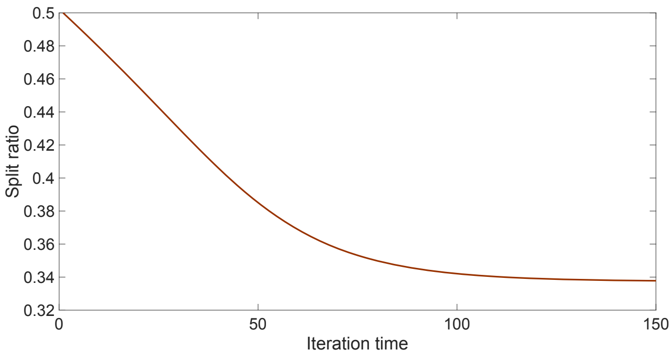

Fig. 5. Case 1 - Numerical data: Learning curve of the split ratio factor X

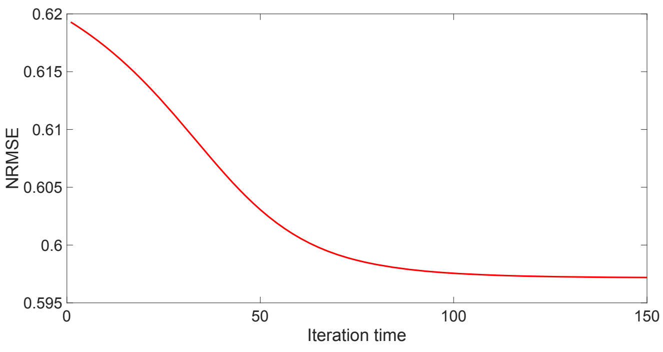

Fig. 6. Case 1 - Numerical data: Learning curve of the NRMSE

Fig. 3 compares the NRMSE performance based on the training data for the traditional LOLIMOT and proposed gradient decent search based LOLIMOT (denoted as GS-based LOLIMOT in the figure). The proposed scheme achieves a better modelling performance than the traditional LOLIMOT.

Fig. 4 shows the NRMSE performance for the testing data (i.e. the model coefficients are fixed at those obtained at the training stage), which also shows the performance improvement from the proposed scheme. Particularly, with the same model size, the proposed scheme has significantly lower NRMSE than the original LOLIMOT counterpart, making it more robust against over-fitting.

Fig. 5 and Fig. 6 show the learning curves of the split ratio factor X and the corresponding NRMSE for one neuron partition splitting process, respectively. It shows that with the split ratio factor converged from the initial value of 0.5 to about 0.34, the NRMSE converges to the optimum. This also verifies that the split ratio of 0.5 used in the LOLIMOT is not optimal.

Case 2: Physical engine model example

In the second case, the high fidelity engine data generated from the Ford 3-cylinder 1.4L gasoline engine model developed in the WAVERT software are used. The system model has three inputs: fuel injection duration, waste gate diameter and best-fit Wiebe burn duration. The system output is the engine torque. We use 4000 data for training the model and 2000 data for validation.

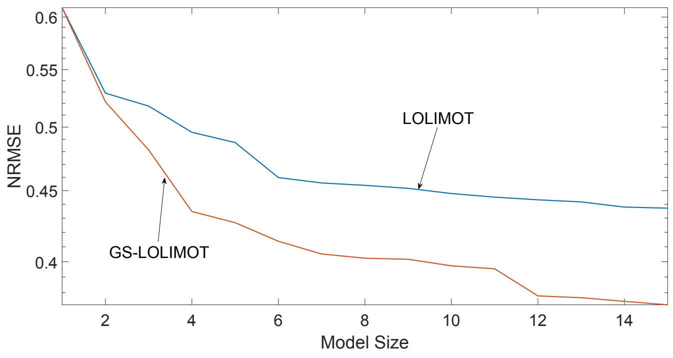

Fig. 7. Case 2 - simulated engine data: The NRMSE performance for the training data

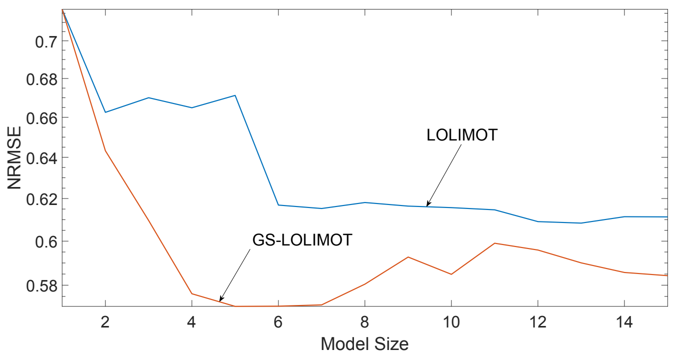

Fig. 8. Case 2: Simulated engine data: The NRMSE performance for the testing data

Fig. 7 and Fig. 8 show the NRMSE performance for the training and testing data, respectively. The comparison

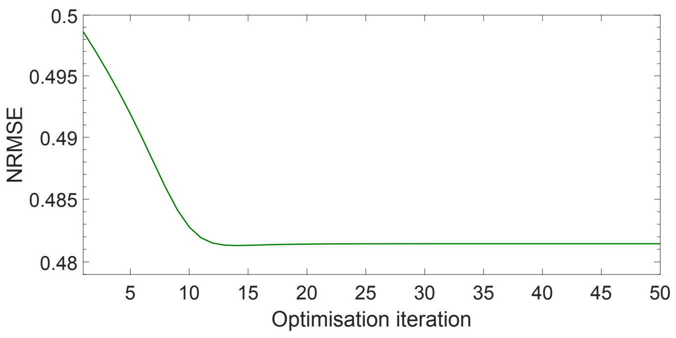

Fig. 9. Case 2: Simulated engine data: Learning curve of the split ratio factor X

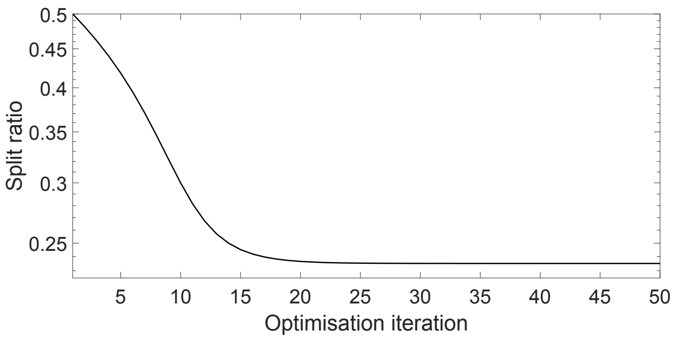

Fig. 10. Case 2: Learning curve of the NRMSE

of the traditional LOLIMOT and the proposed scheme is similar to that in Fig. 3 and Fig. 4. In both Fig. 7 and Fig. 8, the proposed scheme is well superior to the traditional LOLIMOT. Particularly, we can observe in Fig. 8 that when the model size = 4, the proposed scheme has significantly lower NRMSE than the traditional LOLIMOT, which is also lower than the LOLIMOT with model size at 14. This shows that the proposed can achieve better performance with smaller model size.

Fig. 9 and Fig. 10 show the learning curves of the split ratio factor X and the corresponding NRMSE for a splitting processes, respectively. Similar to that in Case 1 using the numerical data, both the split ratio factor and NRMSE can converge to the optimal values.

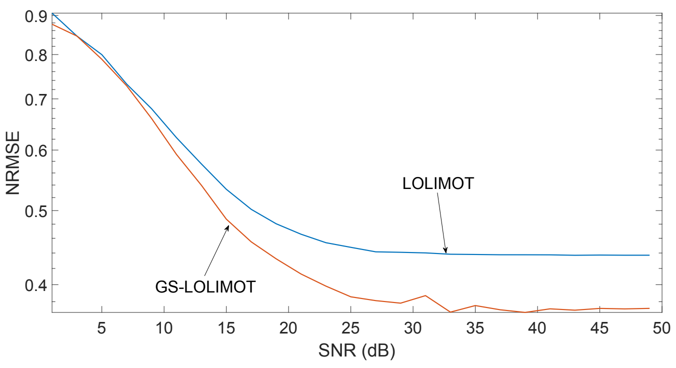

Fig. 11. Training results with different SNR at the last iteration step

Fig. 11 and Fig. 12 show the NRMSE vs SNR performance for the training and testing data when the model outputs are corrupted with noise, respectively. In both figures, the LOLIMOT and GS-LOLIMOT have 15 neurons. It is

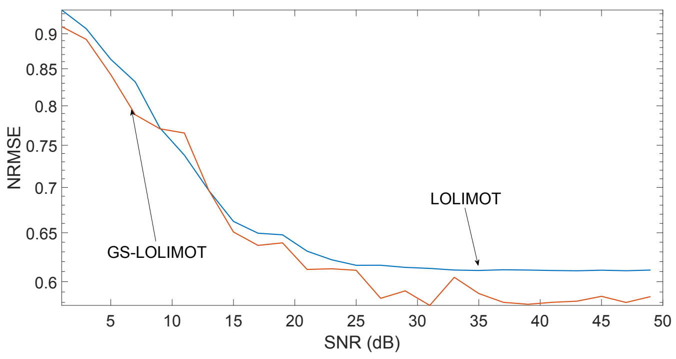

Fig. 12. Testing results of SNR on 15-th split

shown that in Fig. 11 and 12, the proposed GS-LOLIMOT has better modelling performance than the traditional LOLIMOT, indicating its superiority in the robustness against noise.

5. CONCLUSION

In this paper, we described a new LOLIMOT with optimized split ratio to divide the input partitions. The proposed scheme is able to achieve better modelling performance than the traditional LOLIMOT. Particularly the proposed scheme only uses a smaller model size to achieve acceptable performance, which is not only more robust to overfitting but also leads to fast implementation. Because the gradient decent search is used to find the optimum split ratio, the involved complexity is also low. Based on the numerical and simulated engine data, the effectiveness of the proposed scheme has been verified.

REFERENCES

Jamab, A.S. and Araabi, B.N. (2006). A learning algorithm for local linear neuro-fuzzy models with self-construction through merge & split. *2006 IEEE Conference on Cybernetics and Intelligent Systems*, 1–6.

Juang, C.F. and Lin, C.T. (1998). An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Transactions on Fuzzy Systems*, 6(1), 12–32. doi:10.1109/91.660805.

Kashiwagi, H. and Rong, L. (2002). Identification of volterra kernels of nonlinear van de vusse reactor. *International Journal of Control Automation and Systems*, 4(2), 109–113.

Kukolj, D. and Levi, E. (2004). Identification of Complex Systems Based on Neural and Takagi-Sugeno Fuzzy Model. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(1), 272–282.

Lee, S.J. and Ouyang, C.S. (2003). A neuro-fuzzy system modeling with self-constructing rule generation and hybrid SVD-based learning. *IEEE Transactions on Fuzzy Systems*, 11(3), 341–353.

Li, D.P., Liu, Y.J., Tong, S., Chen, C.P., and Li, D.J. (2018). Neural networks-based adaptive control for nonlinear state constrained systems with input delay. *IEEE transactions on cybernetics*, 49(4), 1249–1258.

Lin, Y.Y., Chang, J.Y., and Lin, C.T. (2013). Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 24(2), 310–321.

Nelles, O. and Isermann, R. (1996). Basis function networks for interpolation of local linear models. 470–475.

Nelles, O. (2006). Axes-oblique partitioning strategies for local model networks. In *IEEE International Symposium on Intelligent Control - Proceedings*, 2378–2383. Munich.

Pan, W., Yuan, Y., Gonçalves, J., and Stan, G.B. (2015). A sparse bayesian approach to the identification of nonlinear state-space systems. *IEEE Transactions on Automatic Control*, 61(1), 182–187.

Pratama, M., Er, M.J., Li, X., Oentaryo, R.J., Lughofer, E., and Arifin, I. (2013). Data driven modeling based on dynamic parsimonious fuzzy neural network. *Neurocomputing*, 110, 18–28. URL `http://dx.doi.org/10.1016/j.neucom.2012.11.013`.

Rubio-Solis, A. and Panoutsos, G. (2015). Interval Type-2 Radial Basis Function Neural Network: A Modeling Framework. *IEEE Transactions on Fuzzy Systems*, 23(2), 457–473.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man, and cybernetics*, (1), 116–132.

Yeh, J.W. and Su, S.F. (2017). Efficient Approach for RLS Type Learning in TSK Neural Fuzzy Systems. *IEEE Transactions on Cybernetics*, 47(9), 2343–2352.