

Multilayer Evolving Fuzzy Neural Networks

Supplementary Materials

A. Detailed Derivation of Consequent Parameter Matrix Updating

Given the derivative of the loss with respect to the output of the last layer (the L^{th}) of MEFNN, $\frac{\partial e}{\partial \mathbf{y}^L} = \mathbf{y}^L - \mathbf{r}$ as defined by Eq. (15), the derivative of the loss with respect to the consequent parameters of the n^{th} rule of the L^{th} ENFIS can be formulated as Eq. (S1) ($n = 1, 2, \dots, N^L$):

$$\begin{aligned} \frac{\partial e_w}{\partial a_{n,w,m}^L} &= \frac{\partial e_w}{\partial y_w^L} \cdot \frac{\partial y_w^L}{\partial a_{n,w,m}^L} = (y_w^L - r_w) \cdot \lambda_n^L \cdot \frac{\partial \sigma((a_{n,w}^L)^T \bar{\mathbf{x}}^L)}{\partial a_{n,w,m}^L} \\ &= \lambda_n^L \cdot (y_w^L - r_w) \cdot \sigma'((a_{n,w}^L)^T \bar{\mathbf{x}}^L) \cdot \bar{x}_m^L \end{aligned} \quad (S1)$$

where $w = 1, 2, \dots, W^L$; $m = 0, 1, 2, \dots, M^L$; \bar{x}_m^L is the m^{th} element of the $(M^L + 1) \times 1$ dimensional input vector, $\bar{\mathbf{x}}^L = [1, x_1^L, x_2^L, \dots, x_{M^L}^L]^T$; y_w^L is the w^{th} element of the $W^L \times 1$ dimensional output vector, \mathbf{y}^L ; r_w is the corresponding element of the reference vector, \mathbf{r} ; $\sigma'((a_{n,w}^L)^T \bar{\mathbf{x}}^L) = \sigma((a_{n,w}^L)^T \bar{\mathbf{x}}^L) \cdot (1 - \sigma((a_{n,w}^L)^T \bar{\mathbf{x}}^L))$, and; $e_w = y_w^L - r_w$.

Eq. (S1) can be rewritten in a more compact form as Eq. (S2) as follows.

$$\frac{\partial e}{\partial \mathbf{A}_n^L} = \frac{\partial e}{\partial \mathbf{y}^L} \cdot \frac{\partial \mathbf{y}^L}{\partial \mathbf{A}_n^L} = \lambda_n^L \cdot (\mathbf{d}^L \otimes \sigma'(\mathbf{A}_n^L \bar{\mathbf{x}}^L)) \cdot (\bar{\mathbf{x}}^L)^T \quad (S2)$$

where $\mathbf{d}^L = \frac{\partial e}{\partial \mathbf{y}^L} = (\mathbf{y}^L - \mathbf{r})$.

Next, the derivative of the loss with respect to the inputs of the L^{th} ENFIS can be formulated as Eq. (S3) ($m = 1, 2, \dots, M^L$):

$$\begin{aligned} \frac{\partial e}{\partial x_m^L} &= \sum_{w=1}^{W^L} \frac{\partial e_w}{\partial y_w^L} \cdot \frac{\partial y_w^L}{\partial x_m^L} = \sum_{w=1}^{W^L} \left((y_w^L - r_w) \cdot \sum_{n=1}^{N^L} \left(\frac{\partial \lambda_n^L}{\partial x_m^L} \cdot y_{n,w}^L + \lambda_n^L \cdot \frac{\partial y_{n,w}^L}{\partial x_m^L} \right) \right) \\ &= \sum_{w=1}^{W^L} \left((y_w^L - r_w) \cdot \sum_{n=1}^{N^L} \left(\frac{\partial \lambda_n^L}{\partial x_m^L} \cdot \sigma((a_{n,w}^L)^T \bar{\mathbf{x}}^L) + \lambda_n^L \cdot \sigma'((a_{n,w}^L)^T \bar{\mathbf{x}}^L) \cdot a_{n,w,m}^L \right) \right) \\ &= \sum_{n=1}^{N^L} \left(\frac{\partial \lambda_n^L}{\partial x_m^L} \cdot \sigma^T(\mathbf{A}_n^L \bar{\mathbf{x}}^L) \cdot \mathbf{d}^L + \lambda_n^L \cdot (\tilde{\mathbf{a}}_{n,m}^L)^T \cdot (\mathbf{d}^L \otimes \sigma'(\mathbf{A}_n^L \bar{\mathbf{x}}^L)) \right) \end{aligned} \quad (S3)$$

where $\tilde{\mathbf{a}}_{n,m}^L = [a_{n,1,m}^L, a_{n,2,m}^L, \dots, a_{n,W^L,m}^L]^T$ is $W^L \times 1$ dimensional consequent parameter vector; and there is:

$$\begin{aligned} \frac{\partial \lambda_n^L}{\partial x_m^L} &= \frac{D_n(x^L)}{\sum_{i=1}^{N^L} D_i(x^L)} \cdot \frac{2(p_{n,m}^L - x_m^L)}{(\tau_n^L)^2} - \frac{D_n(x^L)}{(\sum_{i=1}^{N^L} D_i(x^L))^2} \cdot \sum_{i=1}^{N^L} \left(D_i(x^L) \cdot \frac{2(p_{i,m}^L - x_m^L)}{(\tau_i^L)^2} \right) \\ &= \lambda_n^L \left(\frac{2(p_{n,m}^L - x_m^L)}{(\tau_n^L)^2} - \sum_{i=1}^{N^L} \left(\lambda_i^L \cdot \frac{2(p_{i,m}^L - x_m^L)}{(\tau_i^L)^2} \right) \right) \end{aligned} \quad (S4)$$

Eq. (S3) can be further converted to the following compact form as:

$$\begin{aligned} \mathbf{d}^{L-1} &= \frac{\partial e}{\partial \mathbf{y}^L} \cdot \frac{\partial \mathbf{y}^L}{\partial \mathbf{y}^{L-1}} = \mathbf{d}^L \cdot \frac{\partial \mathbf{y}^L}{\partial \mathbf{x}^L} \\ &= \sum_{n=1}^{N^L} \left(\frac{\partial \lambda_n^L}{\partial \mathbf{x}^L} \cdot \sigma^T(\mathbf{A}_n^L \bar{\mathbf{x}}^L) \cdot \mathbf{d}^L + \lambda_n^L \cdot (\tilde{\mathbf{A}}_n^L)^T \cdot (\mathbf{d}^L \otimes \sigma'(\mathbf{A}_n^L \bar{\mathbf{x}}^L)) \right) \end{aligned} \quad (S5)$$

$$\text{where } \frac{\partial \lambda_n^L}{\partial \mathbf{x}^L} = \lambda_n^L \left(\frac{2(p_n^L - \mathbf{x}^L)}{(\tau_n^L)^2} - \sum_{i=1}^{N^L} \left(\lambda_i^L \cdot \frac{2(p_i^L - \mathbf{x}^L)}{(\tau_i^L)^2} \right) \right) \quad (S6)$$

Similarly, Eqs. (S7) and (S8) can be derived following the chain rule ($l = 1, 2, \dots, L - 1$):

$$\begin{aligned} \mathbf{d}^l &= \frac{\partial e}{\partial \mathbf{y}^L} \cdot \frac{\partial \mathbf{y}^L}{\partial \mathbf{y}^{L-1}} \cdot \dots \cdot \frac{\partial \mathbf{y}^{l+2}}{\partial \mathbf{y}^{l+1}} \cdot \frac{\partial \mathbf{y}^{l+1}}{\partial \mathbf{y}^l} = \mathbf{d}^{l+1} \cdot \frac{\partial \mathbf{y}^{l+1}}{\partial \mathbf{x}^{l+1}} \\ &= \sum_{n=1}^{N^{l+1}} \left(\frac{\partial \lambda_n^{l+1}}{\partial \mathbf{x}^{l+1}} \cdot \sigma^T(\mathbf{A}_n^{l+1} \bar{\mathbf{x}}^{l+1}) \cdot \mathbf{d}^{l+1} + \lambda_n^{l+1} \cdot (\tilde{\mathbf{A}}_n^{l+1})^T \cdot (\mathbf{d}^{l+1} \otimes \sigma'(\mathbf{A}_n^{l+1} \bar{\mathbf{x}}^{l+1})) \right) \end{aligned} \quad (\text{S7})$$

$$\frac{\partial e_k}{\partial \mathbf{A}_n^l} = \frac{\partial e}{\partial \mathbf{y}^L} \cdot \frac{\partial \mathbf{y}^L}{\partial \mathbf{y}^{L-1}} \cdot \dots \cdot \frac{\partial \mathbf{y}^{l+1}}{\partial \mathbf{y}^l} \cdot \frac{\partial \mathbf{y}^l}{\partial \mathbf{A}_n^l} = \mathbf{d}^l \cdot \frac{\partial \mathbf{y}^l}{\partial \mathbf{A}_n^l} = \lambda_n^l \cdot (\mathbf{d}^l \otimes \sigma'(\mathbf{A}_n^l \bar{\mathbf{x}}^l)) \cdot (\bar{\mathbf{x}}^l)^T \quad (\text{S8})$$

B. Pseudo-code of MEFNN System Identification Process

Algorithm S1. MEFNN identification process

```

k ← 1;
while (xk is available) do
  xk1 ← xk;
  if (k = 1) then
    ### Stage 0 ###
    for l = 1 to L do
      Nl ← 1;
      initialise μl and Xl by (7);
      initialise RNll by (8);
      initialise CNll by (9);
      produce ykl by (5);
      if (l ≠ L) then
        xkl+1 ← ykl;
      else
        yk ← ykl;
      end if
    end for
  else
    ### Stage 1 ###
    for l = 1 to L do
      update μl and Xl by (10);
      for n = 1 to Nl do
        calculate Dn(xkl) by (11);
      end for
      if (Condition 1 is satisfied) then
        Nl ← Nl + 1;
        initialise RNll by (8);
        initialise CNll by (9);
      else
        n* = argmaxn=1,2,...,n* (Dn(xkl));
        update Cn*l by (13);
      end
      produce ykl by (5);
      if (l ≠ L) then
        xkl+1 ← ykl;
      else
        yk ← ykl;
      end if
    end for
    ### Stage 2 ###
    calculate ek by (14);
    calculate  $\frac{\partial e_k}{\partial \mathbf{y}_k}$  by (15);
  end if

```

```

 $\mathbf{d}_k^L \leftarrow \frac{\partial e_k}{\partial \mathbf{y}_k};$ 
for  $l = L$  to 1 do
  calculate  $\frac{\partial e_k}{\partial \mathbf{A}_n^l}$  by (17);
  update  $\mathbf{A}_n^l$  by (19);
  if ( $l \neq 1$ ) then
    calculate  $\mathbf{d}_k^{l-1}$  by (18);
  end if
end for
end if
 $k \leftarrow k + 1;$ 
end while

```

C. Key Information of Benchmark Datasets for Experimental Investigation

Table S1. Key information of numerical benchmark problems for classification

| Dataset | #(Samples) | #(Attributes) | #(Classes) |
|---------|------------|----------------------|------------|
| CA | 2126 | 21 inputs + 1 label | 3 |
| WF | 5456 | 24 inputs + 1 label | 4 |
| GP | 9901 | 18 inputs + 1 label | 5 |
| OR | Training | 64 inputs + 1 label | 10 |
| | Testing | | |
| AU | 690 | 14 inputs + 1 label | 2 |
| BA | 625 | 4 inputs + 1 label | 3 |
| LI | 345 | 6 inputs + 1 label | 2 |
| MG | 19020 | 10 inputs + 1 label | 2 |
| MO | 432 | 6 inputs + 1 label | 2 |
| PB | 5472 | 10 inputs + 1 label | 5 |
| PI | 768 | 8 inputs + 1 label | 2 |
| SE | 2584 | 15 inputs + 1 label | 2 |
| SO | 208 | 60 inputs + 1 label | 2 |
| SH | 267 | 44 inputs + 1 label | 2 |
| OD | Training | 5 inputs + 1 label | 2 |
| | Testing | | |
| MF | 2000 | 649 inputs + 1 label | 10 |
| PR | Training | 16 inputs + 1 label | 10 |
| | Testing | | |
| AB | 4177 | 8 inputs + 1 label | 3 |
| IS | Training | 19 inputs + 1 label | 7 |
| | Testing | | |
| PW | 11055 | 30 inputs + 1 label | 2 |
| SP | 4601 | 57 inputs + 1 label | 2 |
| MA | 11183 | 6 inputs + 1 label | 2 |
| TE | 5500 | 40 inputs + 1 label | 11 |
| SPF | 1941 | 27 inputs + 1 label | 7 |

Table S2. Key information of benchmark problems for image classification

| Dataset | #(Images) | #(Pixels) | #(Classes) |
|---------|-----------|-----------|------------|
| OPT | 1860 | 256×256 | 31 |
| WHU | 950 | 600×600 | 19 |
| UCM | 2100 | 256×256 | 21 |
| RSS | 2800 | 400×400 | 7 |

Table S3. Key information of benchmark problems for intrusion detection

| Dataset | | #(Samples) | #(Attributes) | #(Normal Samples) | #(Anomalies) |
|----------|----------|------------|--|-------------------|--------------|
| NSLKDD | Training | 125973 | 38 numerical inputs + 3 categorical inputs + 1 label | 67343 | 58630 |
| | Testing | 22544 | | 9711 | 12833 |
| UNSWNB15 | Training | 175341 | 40 numerical inputs + 3 categorical inputs + 1 label | 56000 | 119341 |
| | Testing | 82332 | | 37000 | 45332 |

Table S4. Key information of real-world regression problems

| Dataset | #(Attributes) | #(Training Samples) | #(Testing Samples) |
|--------------------|----------------------|---------------------|--------------------|
| Autos | 15 inputs + 1 output | 80 | 79 |
| Autompg | 6 inputs + 1 output | 196 | 196 |
| Delta ailerons | 5 inputs + 1 output | 3000 | 4129 |
| California housing | 8 inputs + 1 output | 10320 | 10320 |
| Mackey-Glass | 3 inputs + 1 output | 3000 | 500 |
| S&P500 | 5 inputs + 1 output | 14893 | 14893 |

Details of Mackey-Glass and S&P500 problems are as follows.

Mackey-Glass: the time series are generated from the following differential equation [1]:

$$\frac{dx_k}{dk} = \frac{0.2x_{k-\tau}}{1+x_{k-\tau}^{10}} - 0.1x_k. \quad (S9)$$

In this study, $\tau = 17$ and the initial state $x_0 = 1.2$. In the prediction process, the input vector, which consists of three past values and the current value $[x_{k-18}, x_{k-12}, x_{k-6}, x_k]^T$, is used to predict the value x_{k-85} . A total of 3000 training samples are extracted from the time period between $k = 201$ and $k = 3200$. 500 testing samples are extracted from the time period between $k = 5001$ and $k = 5500$.

S&P500: The daily closing prices of S&P500 are collected from the Yahoo! Finance website ranging from 03/01/1950 to 12/03/2009, 14893 samples in total. Following the common practice, the value range of data has been normalised to $[0,1]$ in advance. This dataset is then expanded with its flipped time series, and thus, a total of 29786 samples after augmentation. The first half of the augmented dataset, namely, the original time series, are used for training and the flipped time series are used for online testing under the prequential test-then-train protocol [2]. The regression model is defined as follows.

$$x_{k+1} = f(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1}, x_k). \quad (S10)$$

D. Sensitivity Analysis

In this section, the influences of the three externally controlled parameters, namely, L , W_o and δ_o on the prediction performance of MEFNN is studied using the following four benchmark problems, namely, 1) CA; 2) WF; 3) GP, and; 4) OR. In running the experiments, for CA, WF and GP datasets, 50% of the data samples are randomly selected to build the training sets and the remaining 50% data samples are used for testing. The original training-testing split of OR dataset is kept.

Firstly, the influences of L and W_o on the prediction performance of MEFNN are investigated. In this example, δ_o is fixed as $\delta_o = e^{-3}$, the value of L is varied from 1 to 3 and the value of W_o is varied from W , $2W$, $3W$ and M . Not that MEFNN is reduced to a single ENFIS model if $L = 1$ and the value of W_o has no influence on a single layer MEFNN. The average classification accuracy curves of MEFNN with different settings of L and W_o over the four datasets are depicted in Fig. S1. The obtained results after 200 training epochs in terms of classification accuracy and number of rules per layer are tabulated in Table S5.

Table S5. Classification performance of MEFNN with different settings of L and W_o

| L | W_o | Meas. | CA | WF | GP | OR |
|-----|-------|-------------------|--------------------|-------------------|-------------------|-------------------|
| 1 | | Acc | 0.8919 | 0.8450 | 0.6198 | 0.9603 |
| | | (N^1) | (14.2) | (7.7) | (4.1) | (8.0) |
| 2 | W | Acc | 0.9015 | 0.8792 | 0.6434 | 0.9613 |
| | | (N^1, N^2) | (14.2, 19.0) | (7.7, 21.1) | (4.1, 13.4) | (8.0, 27.6) |
| | $2W$ | Acc | 0.9058 | 0.8988 | 0.6635 | 0.9651 |
| | | (N^1, N^2) | (14.2, 25.0) | (7.7, 23.6) | (4.1, 17.6) | (8.0, 24.0) |
| | $3W$ | Acc | 0.9046 | 0.9015 | 0.6707 | 0.9672 |
| | | (N^1, N^2) | (14.2, 24.1) | (7.7, 26.3) | (4.1, 16.5) | (8.0, 19.0) |
| | M | Acc | 0.9025 | 0.9001 | 0.6689 | 0.9667 |
| | | (N^1, N^2) | (14.2, 26.8) | (7.7, 24.5) | (4.1, 20.1) | (8.0, 21.5) |
| 3 | W | Acc | 0.8988 | 0.8738 | 0.6272 | 0.9459 |
| | | (N^1, N^2, N^3) | (14.2, 16.3, 15.8) | (7.7, 17.1, 20.1) | (4.1, 12.1, 24.2) | (8.0, 22.2, 30.1) |
| | $2W$ | Acc | 0.9041 | 0.8965 | 0.6468 | 0.9553 |
| | | (N^1, N^2, N^3) | (14.2, 23.3, 22.0) | (7.7, 22.7, 19.8) | (4.1, 10.9, 28.8) | (8.0, 19.8, 24.4) |
| | $3W$ | Acc | 0.9045 | 0.9046 | 0.6610 | 0.9526 |
| | | (N^1, N^2, N^3) | (14.2, 24.6, 18.7) | (7.7, 25.1, 23.5) | (4.1, 12.2, 26.0) | (8.0, 19.6, 23.3) |
| | M | Acc | 0.9030 | 0.9046 | 0.6694 | 0.9618 |
| | | (N^1, N^2, N^3) | (14.2, 30.6, 17.9) | (7.7, 23.1, 20.5) | (4.1, 14.5, 27.7) | (8.0, 19.3, 26.4) |

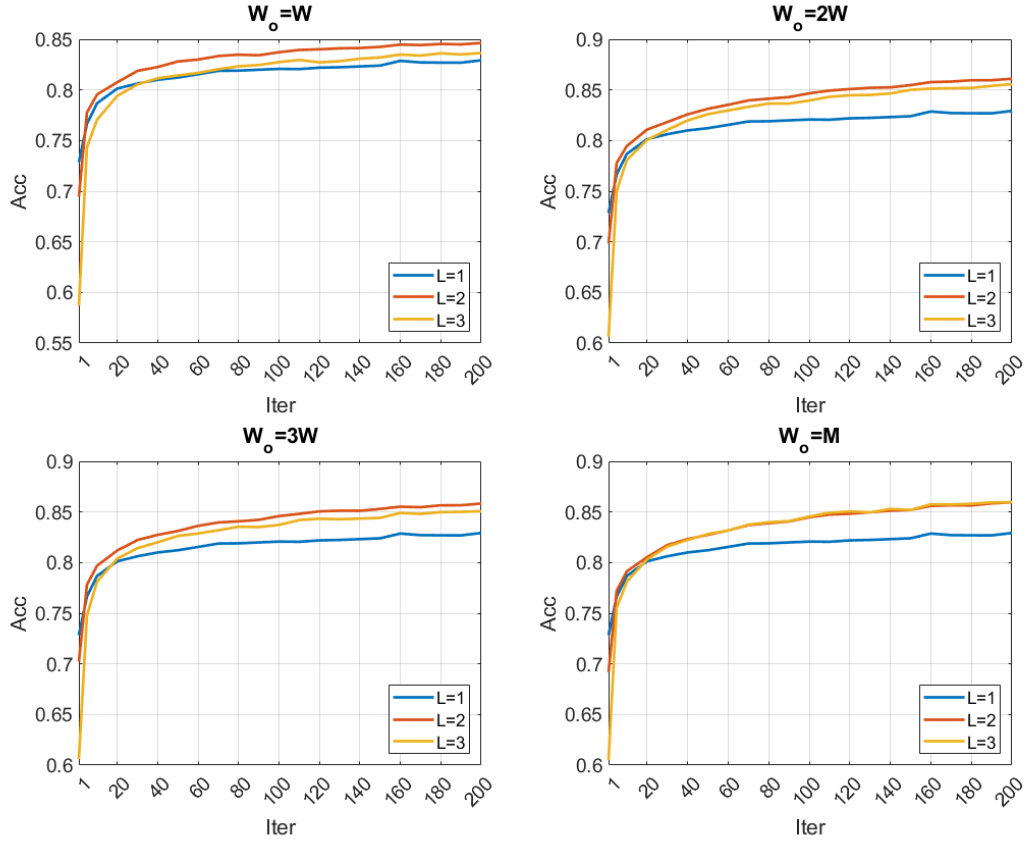


Fig. S1. Average prediction accuracy of MEFNN with different parameter settings over the training process

One can see from Table S5 and Fig. S1 that, in general, the prediction accuracy of MEFNN increases with its ensemble structure going deeper and more base models stacked in layers, namely, a larger L . This is thanks to its stronger multi-level nonlinear distributed representation ability, benefiting from the deeper structure and a larger amount of parameters learned from data. The output size, W_o also influences the prediction accuracy of the

proposed ensemble system to a certain degree. With a greater W_o , more information is allowed to travel between different layers of MEFNN, allowing the ensemble system to learn more descriptive representations from data and achieve better prediction performance. However, as the dimensionality of consequent parameter matrices of the individual base models within the stacking ensemble system becomes higher, the amount of trainable parameters of MEFNN also increases, which may greatly increase the computational complexity for parameter updating and structure evolving.

Next, the influence of δ_o on the performance of the proposed ensemble system is studied. In running the experiments, the value of W_o is fixed as $3W$ and the value of δ_o is varied from e^{-2} , e^{-3} , e^{-4} , and e^{-5} . The results are reported in Table S6.

One can see from Table S6 that, a greater δ_o enables MEFNN to become more sensitive towards these input samples that are spatially distant from existing prototypes. As these samples are more likely to represent new local data patterns unseen in the historical data, MEFNN will use them to expand its knowledge base and each ensemble component within MEFNN tends to build a larger rule base. In general, having larger rule bases enables MEFNN to approximate the problem more precisely, but also this leads to a greater risk of overfitting. On the other hand, a smaller δ_o forces MEFNN to focus on major data patterns and identify less (but more important) prototypes from data. In such cases, MEFNN will have a more compact knowledge base and its computational efficiency will be improved greatly due to less rules in the rule bases of its base models. However, if the existing prototypes are insufficient to represent the underlying data patterns and fail to preserve the structure of data, the performance of MEFNN will be impacted negatively.

Table S6. Classification performance of MEFNN with different settings of L and δ_o

| L | δ_o | Meas. | CA | WF | GP | OR |
|-----|------------|---------------------|--------------------|--------------------|-------------------|--------------------|
| 1 | e^{-2} | Acc | 0.8916 | 0.8522 | 0.6284 | 0.9622 |
| | | (N^1) | (31.9) | (38.9) | (6.3) | (37.7) |
| | e^{-3} | Acc | 0.8919 | 0.8450 | 0.6198 | 0.9603 |
| | | (N^1) | (14.2) | (7.7) | (4.1) | (8.0) |
| | e^{-4} | Acc | 0.8968 | 0.8357 | 0.6140 | 0.9554 |
| | | (N^1) | (6.5) | (4.3) | (3.5) | (3.8) |
| | e^{-5} | Acc | 0.8971 | 0.8193 | 0.6120 | 0.9520 |
| | | (N^1) | (4.4) | (2.6) | (3.1) | (2.5) |
| | e^{-2} | Acc | 0.9018 | 0.8899 | 0.6636 | 0.9622 |
| | | (N^1, N^2) | (31.9, 54.9) | (38.9, 75.9) | (6.3, 46.9) | (37.7, 66.6) |
| | | Acc | 0.9046 | 0.9015 | 0.6707 | 0.9672 |
| | | (N^1, N^2) | (14.2, 24.1) | (7.7, 26.3) | (4.1, 16.5) | (8.0, 19.0) |
| | | Acc | 0.9014 | 0.9027 | 0.6727 | 0.9656 |
| | | (N^1, N^2) | (6.5, 14.4) | (4.3, 12.7) | (3.5, 9.7) | (3.8, 10.8) |
| | e^{-5} | Acc | 0.9044 | 0.9045 | 0.6685 | 0.9615 |
| | | (N^1, N^2) | (4.4, 11.6) | (2.6, 9.3) | (3.1, 7.6) | (2.5, 8.0) |
| 3 | e^{-2} | Acc | 0.9054 | 0.8998 | 0.6648 | 0.9593 |
| | | (N^1, N^2, N^3) | (31.9, 54.2, 29.5) | (38.9, 59.9, 53.7) | (6.3, 41.3, 69.9) | (37.7, 46.9, 67.9) |
| | e^{-3} | Acc | 0.9045 | 0.9046 | 0.6610 | 0.9526 |
| | | (N^1, N^2, N^3) | (14.2, 24.6, 18.7) | (7.7, 25.1, 23.5) | (4.1, 12.2, 26.0) | (8.0, 19.6, 23.3) |
| | e^{-4} | Acc | 0.9032 | 0.8932 | 0.6652 | 0.9534 |
| | | (N^1, N^2, N^3) | (6.5, 17.3, 12.6) | (4.3, 12.1, 11.4) | (3.5, 9.9, 15.8) | (3.8, 11.6, 13.9) |
| | e^{-5} | Acc | 0.9028 | 0.8943 | 0.6736 | 0.9528 |
| | | (N^1, N^2, N^3) | (4.4, 10.9, 11.0) | (2.6, 9.5, 9.6) | (3.1, 6.1, 10.3) | (2.5, 9.9, 9.4) |

E. Ablation Analysis

In this section, an ablation analysis is performed to demonstrate the influence of the extra nonlinearity introduced by the sigmoid function to MEFNN on its prediction performance. In this example, an alternative version of MEFNN utilising the classical linear functions in the consequent part of its IF-THEN rules is implemented and named as MEFNN_L to differentiate it from the original MEFNN that uses sigmoid function. In running the experiments, the externally controlled parameters of MEFNN and MEFNN_L are set as $L = 2$; $W_o = 3W$, and $\delta_o =$

e^{-3} , following the recommended setting. The learning rate of MEFNN_L is set as $\gamma_o = 10^{-3}$ to avoid gradient explosion. The same four benchmark problems used before, namely, 1) CA; 2) WF; 3) GP, and; 4) OR are employed, and the same training-testing splits are adopted. The performance of MEFNN and MEFNN_L are reported in Table S7 after 200 training epochs.

From Table S7 one can see that MEFNN outperforms MEFNN_L under the same experimental setting on all four benchmark problems by offering greater prediction accuracy with less IF-THEN rules (in the second layer). This numerical example justifies the use of sigmoid function in MEFNN and demonstrates clearly the performance improvement thanks to the extra nonlinearity brought by the sigmoid function.

Table S7. Classification performance comparison between MEFNN and MEFNN_L

| Algorithm | Meas. | CA | WF | GP | OR |
|--------------------|--------------|--------------|-------------|-------------|-------------|
| MEFNN | <i>Acc</i> | 0.9046 | 0.9015 | 0.6707 | 0.9672 |
| | (N^1, N^2) | (14.2, 24.1) | (7.7, 26.3) | (4.1, 16.5) | (8.0, 19.0) |
| MEFNN _L | <i>Acc</i> | 0.8978 | 0.8325 | 0.5910 | 0.9418 |
| | (N^1, N^2) | (14.2, 41.5) | (7.7, 78.6) | (4.1, 37.0) | (8.0, 99.3) |

F. Parameter settings for Comparative Approaches

In this study, SVM uses Gaussian kernel. The kernel width is determined through a heuristic procedure automatically and the box constraint is set as 1. $k = 5$ is used for both KNN and SDKNN. SEQ follows the same parameter setting as [3]. The maximum number of neurons of ELM is set as 200. MLP has three hidden layers, each hidden layer has 20 neutrons. LSTM has a single hidden LSTM layer and the number of neurons is set as 100. Both MLP and LSTM are trained for 200 epochs. The standard deviation of PNN is set as $\frac{\delta_x}{\sqrt{2}}$ (δ_x is the standard deviation of the training data). EIG considers the first five eigenvalues for classification. The parameter setting of ESAFIS is as follows: $\gamma = 0.999$; $\varepsilon_{min} = 0.1$; $\varepsilon_{max} = 1.2$; $K = 1.5$; $e_g = 0.01$; $e_p = 0.01$, and; $M = 20$. The first-order PALM uses local updating strategy and its parameters are set as: $a = 0.1$; $b_1 = 0.002$; $b_2 = 0.01$; $c_1 = 0.01$, and; $c_2 = 0.01$. The externally controlled parameters of SEFIS are determined as: $K = 0.5$; $\delta_1 = 0.5$; $\delta_2 = 0.5$, and $p_0 = 2$. SPA, SAFL, MOOSOFIS, eClass0 and eClass1 follow the same experimental settings as [4]–[7], respectively. Since SAFL, SEFIS, ESAFIS and PALM are first-order EFSs designed for regression tasks, they employ the “one-versus-all” strategy for classification.

For the ensemble models used for performance comparison, the maximum split of DT used by RF, SAMMED and XGBoost, is set as 20 and the number of base classifiers is set as 50. $\eta = 0.3$ is used for XGBoost. In SAMMEK, $k = 5$ is used for KNN. FWAdaBoost uses self-organising fuzzy inference system (SOFIS) as its base classifier, and the same experimental setting in [8] is followed. eEnsemble is composed of 10 eClass0 classifiers, and other parameters follow the same setting used in [9].

G. Supplementary Numerical Results

Table S8. Performance comparison on 10 benchmark classification problems from KEEL

| Dataset | Measure | MEFNN | MEFNN ₃ | EIG | SPA | SAFL | MOOSOFIS | SEFIS | PALM | CPBLS | CFBLS | HIDFC |
|---------|---------|---------------|--------------------|--------|--------|--------|----------|--------|---------------|---------------|---------------|--------|
| AU | Max | 0.8986 | 0.8824 | 0.7971 | 0.8841 | 0.8857 | 0.8714 | 0.8696 | 0.9143 | 0.8826 | 0.8870 | 0.8647 |
| | Mean | 0.8524 | 0.8380 | 0.7582 | 0.8464 | 0.8466 | 0.8290 | 0.7403 | 0.8580 | 0.8736 | 0.8750 | 0.8337 |
| | Std | 0.0340 | 0.0476 | 0.0382 | 0.0236 | 0.0399 | 0.1160 | 0.0945 | 0.0436 | 0.0027 | 0.0036 | 0.0187 |
| BA | Max | 1.0000 | 1.0000 | 0.9206 | 0.9048 | 0.9206 | 0.8889 | 0.8689 | 0.9048 | 0.9920 | 0.9119 | 0.8594 |
| | Mean | 0.9583 | 0.9647 | 0.9055 | 0.8704 | 0.9009 | 0.7951 | 0.6473 | 0.8753 | 0.9894 | 0.9066 | 0.8416 |
| | Std | 0.0246 | 0.0262 | 0.0181 | 0.0285 | 0.0159 | 0.0994 | 0.1967 | 0.0201 | 0.0014 | 0.0035 | 0.0155 |

| | | | | | | | | | | | | |
|----|------|---------------|---------------|---------------|---------------|---------------|---------------|--------|--------|---------------|---------------|--------|
| LI | Max | 0.8000 | 0.7714 | 0.7647 | 0.7714 | 0.7941 | 0.8000 | 0.6286 | 0.7353 | 0.7361 | 0.7568 | 0.6935 |
| | Mean | 0.6753 | 0.6637 | 0.6666 | 0.5997 | 0.6956 | 0.6314 | 0.5360 | 0.6610 | 0.7252 | 0.7464 | 0.6563 |
| | Std | 0.0752 | 0.0475 | 0.0607 | 0.1031 | 0.0586 | 0.0854 | 0.0432 | 0.0647 | 0.0066 | 0.0062 | 0.0212 |
| MG | Max | 0.8759 | 0.8754 | 0.8686 | 0.8448 | 0.8654 | 0.8128 | 0.7940 | 0.6768 | 0.8563 | 0.8557 | 0.8241 |
| | Mean | 0.8641 | 0.8664 | 0.8621 | 0.8357 | 0.8583 | 0.7993 | 0.7090 | 0.6657 | 0.8548 | 0.8533 | 0.8160 |
| | Std | 0.0074 | 0.0060 | 0.0063 | 0.0096 | 0.0068 | 0.0808 | 0.0895 | 0.0044 | 0.0005 | 0.0016 | 0.0051 |
| MO | Max | 1.0000 | 1.0000 | 0.9286 | 0.9318 | 1.0000 | 0.8409 | 0.7907 | 0.8864 | 0.9261 | 0.9353 | 0.6599 |
| | Mean | 0.9863 | 0.9977 | 0.8063 | 0.8798 | 0.9354 | 0.7798 | 0.6323 | 0.8220 | 0.9180 | 0.9066 | 0.6494 |
| | Std | 0.0245 | 0.0072 | 0.0586 | 0.0350 | 0.0343 | 0.0835 | 0.0983 | 0.0548 | 0.0040 | 0.0149 | 0.0120 |
| PB | Max | 0.9670 | 0.9651 | 0.9800 | 0.9689 | 0.9652 | 0.9653 | 0.9246 | 0.9156 | 0.9596 | 0.9567 | 0.9137 |
| | Mean | 0.9574 | 0.9600 | 0.9713 | 0.9488 | 0.9572 | 0.9563 | 0.8406 | 0.9065 | 0.9590 | 0.9550 | 0.9067 |
| | Std | 0.0072 | 0.0030 | 0.0069 | 0.0110 | 0.0052 | 0.1261 | 0.1725 | 0.0077 | 0.0003 | 0.0011 | 0.0040 |
| PI | Max | 0.8571 | 0.8026 | 0.8026 | 0.8158 | 0.8052 | 0.7662 | 0.7143 | 0.8312 | 0.7395 | 0.7890 | 0.7424 |
| | Mean | 0.7761 | 0.7526 | 0.7084 | 0.7475 | 0.7539 | 0.7187 | 0.6274 | 0.7553 | 0.7316 | 0.7764 | 0.7182 |
| | Std | 0.0413 | 0.0308 | 0.0502 | 0.0379 | 0.0474 | 0.1233 | 0.0695 | 0.0419 | 0.0034 | 0.0038 | 0.0156 |
| SE | Max | 0.9612 | 0.9612 | 0.9651 | 0.9535 | 0.9574 | 0.9419 | 0.9341 | 0.9612 | 0.9354 | 0.9333 | 0.9168 |
| | Mean | 0.9276 | 0.9253 | 0.9303 | 0.9237 | 0.9276 | 0.9168 | 0.8820 | 0.9330 | 0.9348 | 0.9321 | 0.9023 |
| | Std | 0.0176 | 0.0189 | 0.0181 | 0.0189 | 0.0170 | 0.1049 | 0.0505 | 0.0157 | 0.0003 | 0.0009 | 0.0011 |
| SO | Max | 1.0000 | 0.9524 | 0.8571 | 0.9524 | 0.9524 | 0.9524 | 0.6818 | 0.8571 | 0.7986 | 0.8169 | 0.7280 |
| | Mean | 0.8665 | 0.8561 | 0.7628 | 0.8749 | 0.8568 | 0.8656 | 0.6006 | 0.7689 | 0.7783 | 0.7591 | 0.6585 |
| | Std | 0.0909 | 0.0894 | 0.1086 | 0.0695 | 0.0786 | 0.1179 | 0.0474 | 0.0601 | 0.0108 | 0.0162 | 0.0351 |
| SH | Max | 0.8846 | 0.8889 | 0.8846 | 0.8571 | 0.8846 | 0.8214 | 0.8077 | 0.8462 | 0.8153 | 0.8470 | 0.7875 |
| | Mean | 0.8205 | 0.7718 | 0.7794 | 0.7864 | 0.8093 | 0.7594 | 0.7759 | 0.7983 | 0.7929 | 0.8335 | 0.7520 |
| | Std | 0.0404 | 0.0660 | 0.0741 | 0.0398 | 0.0468 | 0.0918 | 0.0417 | 0.0406 | 0.0104 | 0.0055 | 0.0344 |

Table S9. Classification accuracy comparison on 10 benchmark classification problems from UCI

| Algorithm | OD | MF | PR | AB | IS | PW | SP | MA | TE | SPF |
|--------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| MEFNN | 0.9472 | 0.9715 | 0.9769 | 0.5537 | 0.9082 | 0.9509 | 0.9299 | 0.9834 | 0.9890 | 0.7175 |
| MEFNN ₃ | 0.9171 | 0.9654 | 0.9759 | 0.5536 | 0.9022 | 0.9503 | 0.9265 | 0.9844 | 0.9813 | 0.7039 |
| SVM | 0.9167 | 0.9749 | 0.9772 | 0.5381 | 0.9214 | 0.9514 | 0.9098 | 0.9789 | 0.9856 | 0.7205 |
| KNN | 0.9280 | 0.9761 | 0.9760 | 0.5259 | 0.9067 | 0.9298 | 0.8844 | 0.9852 | 0.9778 | 0.6832 |
| SEQ | 0.5189 | 0.9760 | 0.9511 | 0.4521 | 0.8824 | 0.9470 | 0.8853 | 0.6259 | 0.9814 | 0.6823 |
| SDKNN | 0.7351 | 0.9748 | 0.9531 | 0.4506 | 0.8381 | 0.9094 | 0.8865 | 0.5203 | 0.9728 | 0.6890 |
| ELM | 0.9899 | 0.2115 | 0.9483 | 0.3903 | 0.2280 | 0.9077 | 0.8758 | 0.9858 | 0.9811 | 0.6521 |
| MLP | 0.9102 | 0.8399 | 0.9478 | 0.5512 | 0.8976 | 0.9400 | 0.8238 | 0.9832 | 0.9757 | 0.6845 |
| LSTM | 0.9165 | 0.9825 | 0.9690 | 0.5492 | 0.8646 | 0.9414 | 0.9272 | 0.9837 | 0.9875 | 0.7109 |
| PNN | 0.8767 | 0.9478 | 0.9620 | 0.5271 | 0.8033 | 0.9478 | 0.8898 | 0.8598 | 0.9795 | 0.6836 |
| EIG | 0.9543 | 0.8935 | 0.9148 | 0.5379 | 0.8629 | 0.8354 | 0.8573 | 0.8952 | 0.9763 | 0.6858 |
| SAFL | 0.9242 | 0.9846 | 0.9648 | 0.5620 | 0.9017 | 0.9343 | 0.9164 | 0.9831 | 0.9946 | 0.7160 |
| MOOSOFIS | 0.9179 | 0.9727 | 0.9767 | 0.5099 | 0.9030 | 0.9358 | 0.8816 | 0.9830 | 0.9752 | 0.6756 |
| SEFIS | 0.8656 | 0.7264 | 0.7675 | 0.3983 | 0.6617 | 0.8654 | 0.6210 | 0.8715 | 0.5384 | 0.3956 |
| ESAFIS | 0.9597 | 0.9700 | 0.9154 | 0.5505 | 0.8954 | 0.9395 | 0.9128 | 0.9828 | 0.9928 | 0.7092 |
| PALM | 0.9824 | 0.9769 | 0.7897 | 0.5419 | 0.8595 | 0.9200 | 0.8888 | 0.9778 | 0.9835 | 0.6592 |
| eClass0 | 0.7795 | 0.8940 | 0.8004 | 0.5018 | 0.8094 | 0.7782 | 0.7780 | 0.7287 | 0.7900 | 0.4905 |
| eClass1 | 0.9743 | 0.9248 | 0.9151 | 0.5569 | 0.8876 | 0.8161 | 0.8988 | 0.9759 | 0.9744 | 0.6959 |
| RF | 0.9576 | 0.9767 | 0.8565 | 0.5509 | 0.9541 | 0.9299 | 0.9222 | 0.9851 | 0.8610 | 0.6855 |
| FWAdaBoost | 0.9335 | 0.9758 | 0.9761 | 0.5263 | 0.8997 | 0.9425 | 0.8729 | 0.9755 | 0.9855 | 0.6952 |
| SAMMED | 0.9240 | 0.9221 | 0.7773 | 0.5470 | 0.8986 | 0.9284 | 0.9042 | 0.9835 | 0.8425 | 0.6738 |
| SAMMEK | 0.9280 | 0.9761 | 0.9760 | 0.5266 | 0.8896 | 0.9298 | 0.8844 | 0.9852 | 0.9778 | 0.6805 |
| XGBoost | 0.9515 | 0.9734 | 0.9620 | 0.5329 | 0.9385 | 0.9658 | 0.9487 | 0.9862 | 0.9724 | 0.7700 |

Table S10. Balanced classification accuracy comparison on 10 benchmark classification problems from UCI

| Algorithm | OD | MF | PR | AB | IS | PW | SP | MA | TE | SPF |
|--------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| MEFNN | 0.9343 | 0.9715 | 0.9773 | 0.5510 | 0.9082 | 0.9497 | 0.9257 | 0.7712 | 0.9890 | 0.7083 |
| MEFNN ₃ | 0.8919 | 0.9657 | 0.9760 | 0.5523 | 0.9009 | 0.9522 | 0.9177 | 0.7407 | 0.9862 | 0.6591 |
| SVM | 0.8530 | 0.9654 | 0.9763 | 0.5485 | 0.9022 | 0.9491 | 0.9224 | 0.7598 | 0.9813 | 0.6667 |
| KNN | 0.9148 | 0.9761 | 0.9762 | 0.5279 | 0.9067 | 0.9286 | 0.8739 | 0.7616 | 0.9778 | 0.7264 |
| SEQ | 0.6164 | 0.9760 | 0.9516 | 0.4551 | 0.8824 | 0.9446 | 0.8868 | 0.6107 | 0.9814 | 0.7280 |
| SDKNN | 0.6802 | 0.9748 | 0.9538 | 0.4504 | 0.8381 | 0.9083 | 0.8787 | 0.6901 | 0.9728 | 0.7333 |
| ELM | 0.9916 | 0.2115 | 0.9480 | 0.3970 | 0.2280 | 0.9023 | 0.8594 | 0.7692 | 0.9811 | 0.7198 |
| MLP | 0.8904 | 0.8399 | 0.9487 | 0.5514 | 0.8976 | 0.9383 | 0.8245 | 0.7135 | 0.9757 | 0.6170 |
| LSTM | 0.8999 | 0.9825 | 0.9694 | 0.5464 | 0.8646 | 0.9413 | 0.9204 | 0.7273 | 0.9875 | 0.7194 |
| PNN | 0.9177 | 0.9478 | 0.9627 | 0.5411 | 0.8033 | 0.9474 | 0.8822 | 0.8663 | 0.9795 | 0.7683 |
| EIG | 0.9447 | 0.8935 | 0.9151 | 0.5404 | 0.8629 | 0.8483 | 0.8247 | 0.6535 | 0.9763 | 0.6902 |
| SAFL | 0.9122 | 0.9846 | 0.9655 | 0.5626 | 0.9017 | 0.9319 | 0.9092 | 0.6830 | 0.9946 | 0.7161 |
| MOOSOFIS | 0.8977 | 0.9727 | 0.9771 | 0.5127 | 0.9030 | 0.9355 | 0.8728 | 0.7682 | 0.9752 | 0.7108 |
| SEFIS | 0.7856 | 0.7264 | 0.7685 | 0.3955 | 0.6617 | 0.8647 | 0.6156 | 0.5563 | 0.5384 | 0.3104 |
| ESAFIS | 0.9485 | 0.9700 | 0.9172 | 0.5549 | 0.8954 | 0.9375 | 0.9019 | 0.6830 | 0.9928 | 0.7001 |
| PALM | 0.9873 | 0.9769 | 0.7907 | 0.5419 | 0.8595 | 0.9172 | 0.8720 | 0.5488 | 0.9835 | 0.5751 |
| eClass0 | 0.8362 | 0.8940 | 0.8022 | 0.5108 | 0.8094 | 0.7826 | 0.7772 | 0.8071 | 0.7900 | 0.5945 |
| eClass1 | 0.9689 | 0.9248 | 0.9162 | 0.5572 | 0.8876 | 0.7974 | 0.8884 | 0.5106 | 0.9744 | 0.6860 |
| RF | 0.9360 | 0.9767 | 0.8570 | 0.5497 | 0.9541 | 0.9263 | 0.9102 | 0.7290 | 0.8610 | 0.5988 |
| FWAdaBoost | 0.9294 | 0.9758 | 0.9765 | 0.5284 | 0.8997 | 0.9410 | 0.8662 | 0.5000 | 0.9855 | 0.7092 |
| SAMMED | 0.8817 | 0.9221 | 0.7769 | 0.5486 | 0.8986 | 0.9273 | 0.8923 | 0.7448 | 0.8425 | 0.6351 |
| SAMMEK | 0.9148 | 0.9761 | 0.9762 | 0.5288 | 0.8896 | 0.9286 | 0.8739 | 0.7616 | 0.9778 | 0.7227 |
| XGBoost | 0.9223 | 0.9734 | 0.9624 | 0.5351 | 0.9386 | 0.9648 | 0.9451 | 0.7820 | 0.9724 | 0.7771 |

Table S11. p -values returned from pairwise Wilcoxon rank tests

| MEFNN vs | OD | MF | PR | AB | IS | PW | SP | MA | TE | SPF |
|------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| SVM | 0.0000 | 0.0003 | 0.1122 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.4971 | 0.0000 |
| KNN | 0.0000 | 0.0058 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.1203 | 0.0000 |
| SEQ | 0.0000 | 0.2404 | 0.0000 | 0.0418 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0007 | 0.0000 |
| SDKNN | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| ELM | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| MLP | 0.0000 | 0.0046 | 0.0026 | 0.0000 | 0.0070 | 0.9092 | 0.0000 | 0.0000 | 0.0892 | 0.0000 |
| LSTM | 0.0000 | 0.1183 | 0.0000 | 0.0000 | 0.1976 | 0.0000 | 0.0000 | 0.0000 | 0.2030 | 0.0015 |
| PNN | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.4891 | 0.0000 |
| EIG | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.5289 | 0.0092 |
| SAFL | 0.0000 | 0.1956 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.1901 | 0.0028 |
| MOOSOFIS | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| SEFIS | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0028 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| ESAFIS | 0.0000 | 0.0001 | 0.0000 | 0.0000 | 0.0000 | 0.0309 | 0.0000 | 0.0000 | 0.5123 | 0.1340 |
| PALM | 0.0000 | 0.0001 | 0.0844 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| eClass0 | 0.0000 | 0.1624 | 0.0000 | 0.0000 | 0.0003 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| eClass1 | 0.0000 | 0.0000 | 0.0000 | 0.0006 | 0.0072 | 0.0000 | 0.0000 | 0.0000 | 0.3178 | 0.1079 |
| RF | 0.0000 | 0.4766 | 0.0000 | 0.0000 | 0.6824 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| FWAdaBoost | 0.0000 | 0.3093 | 0.0045 | 0.0000 | 0.9388 | 0.0272 | 0.4560 | 0.0000 | 0.0007 | 0.0000 |
| SAMMED | 0.0000 | 0.9470 | 0.0000 | 0.0000 | 0.0119 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| SAMMEK | 0.0000 | 0.0058 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.1203 | 0.0000 |
| XGBoost | 0.0000 | 0.0018 | 0.0000 | 0.0000 | 0.1394 | 0.0000 | 0.0737 | 0.0193 | 0.0004 | 0.7060 |

Table S12. Performance comparison on four real-world regression problems

| Dataset | Algorithm | <i>RMSE</i> | <i> #(Rules)</i> |
|-----------------------|--------------|---------------|------------------|
| Auto | MEFNN | 0.0575 | (7.3, 17.0) |
| | PSO-ALMMo | 0.0425 | 8 |
| | SB-ALMMo | 0.0457 | 4 |
| | CEFNS | 0.0666 | 2 |
| | DENFIS | 0.4516 | 8 |
| | eTS | 0.0535 | 3 |
| | ESAFIS | 0.0604 | 3 |
| | SAFIS | 0.1184 | 5 |
| | RMCEFS | 0.0409 | 2 |
| | OS-Fuzzy-ELM | 0.0595 | 2 |
| Automgp | MEFNN | 0.0654 | (3.5, 11.9) |
| | PSO-ALMMo | 0.0694 | 7 |
| | SB-ALMMo | 0.0934 | 3 |
| | CEFNS | 0.0750 | 2 |
| | DENFIS | 0.1458 | 7 |
| | eTS | 0.0864 | 6 |
| | ESAFIS | 0.0731 | 33 |
| | SAFIS | 0.0993 | 2 |
| | RMCEFS | 0.0672 | 2 |
| | OS-Fuzzy-ELM | 0.2037 | 5 |
| Delta ailerons | MEFNN | 0.0501 | (18.2, 45.8) |
| | PSO-ALMMo | 0.0497 | 10 |
| | SB-ALMMo | 0.0506 | 5 |
| | CEFNS | 0.0502 | 3 |
| | DENFIS | 0.0497 | 11 |
| | eTS | 0.0513 | 4 |
| | ESAFIS | 0.0506 | 13 |
| | SAFIS | 0.0549 | 14 |
| | RMCEFS | 0.0498 | 2 |
| | OS-Fuzzy-ELM | 0.0506 | 3 |
| California housing | MEFNN | 0.0656 | (11.0, 68.0) |
| | PSO-ALMMo | 0.0711 | 11 |
| | SB-ALMMo | 0.0771 | 5 |
| | CEFNS | 0.0878 | 2 |
| | DENFIS | 0.0715 | 14 |
| | eTS | 0.0772 | 3 |
| | ESAFIS | 0.0892 | 6 |
| | SAFIS | 0.0988 | 12 |
| | OS-Fuzzy-ELM | 0.1320 | 5 |

Table S13. Performance comparison on S&P500 problem

| Algorithm | <i>NDEI</i> | <i> #(Rules)</i> |
|-----------|---------------|------------------|
| MEFNN* | 0.0162 | (3.3, 9.6) |
| MEFNN | 0.0200 | (3.3, 9.6) |
| SAFL | 0.0121 | 19 |
| PANFIS | 0.09 | 4 |
| GENEFIS | 0.07 | 2 |
| EFS-SLAT | 0.0156 | 23 |
| SEFS | 0.0182 | 2 |
| LEOA | 0.1229 | 52 |

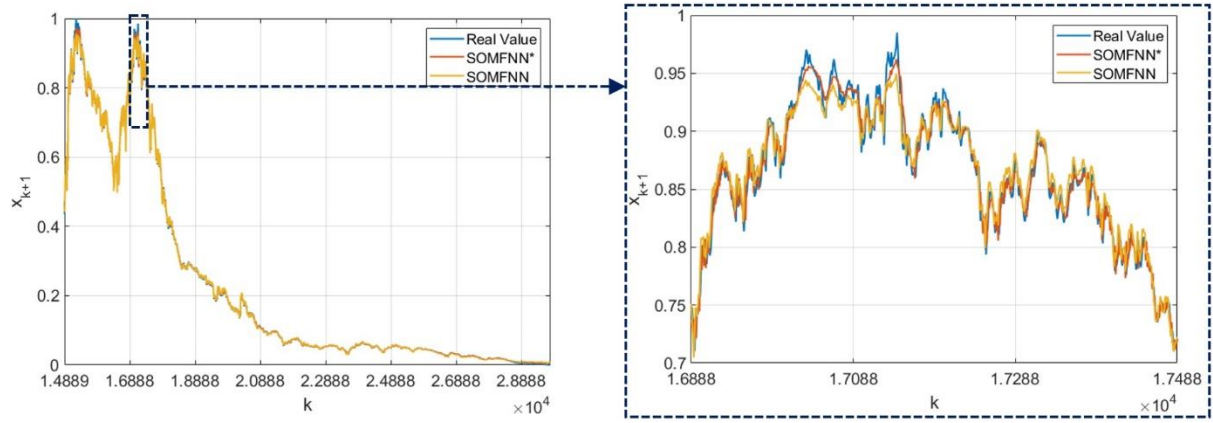


Fig. S2. Prediction results produced by MEFNN on S&P500 problem following different evaluation protocols

Table S14. IF-THEN rules learned by MEFNN from S&P500 problem

| # Layer | # Rule | Detailed Expression |
|---------|---------|--|
| 1 | R_1^1 | $IF (x^1 \sim [0.2910, 0.2905, 0.2888, 0.2883, 0.2887]^T)$ $THEN \left(y_1^1 = \sigma \left(\begin{bmatrix} -1.9004, -0.2939, -0.2887, -0.0710, -0.1356, 0.2327 \\ -0.3612, 0.2766, 0.2714, 0.4005, 0.7297, 1.4117 \\ 0.6712, 0.0155, 0.0037, 0.2279, 0.4030, 0.6291 \end{bmatrix} x^1 \right) \right)$ |
| | R_2^1 | $IF (x^1 \sim [0.0277, 0.0277, 0.0275, 0.0271, 0.0272]^T)$ $THEN \left(y_2^1 = \sigma \left(\begin{bmatrix} -1.9004, -0.2939, -0.2887, -0.0710, -0.1356, 0.2327 \\ -1.7863, 1.9052, 1.9258, 1.9780, 2.0602, 2.0168 \\ -0.5730, 1.5731, 1.4138, 1.4328, 1.4639, 1.5109 \end{bmatrix} x^1 \right) \right)$ |
| | R_3^1 | $IF (x^1 \sim [0.7373, 0.7413, 0.7421, 0.7398, 0.7300]^T)$ $THEN \left(y_3^1 = \sigma \left(\begin{bmatrix} 1.1219, -0.5487, -0.4763, -0.4715, -0.1555, 0.6738 \\ 1.1955, 0.1949, 0.3170, 0.4520, 0.6752, 1.9385 \\ -3.2190, -0.0589, 0.0630, 0.0284, 0.6653, 3.5812 \end{bmatrix} x^1 \right) \right)$ |
| 2 | R_1^2 | $IF (x^2 \sim [0.5240, 0.5600, 0.5240]^T)$ $THEN (y_1^2 = \sigma([-4.6487, -2.2163, -1.4434, -2.0031]x^2))$ |
| | R_2^2 | $IF (x^2 \sim [0.5023, 0.5057, 0.5023]^T)$ $THEN (y_2^2 = \sigma([-5.5448, -2.8085, -1.6564, -2.3125]x^2))$ |
| | R_3^2 | $IF (x^2 \sim [0.5611, 0.6492, 0.5614]^T)$ $THEN (y_3^2 = \sigma([-3.2960, -1.2372, 3.8157, 0.9415]x^2))$ |
| | R_4^2 | $IF (x^2 \sim [0.6431, 0.6816, 0.5487]^T)$ $THEN (y_4^2 = \sigma([-0.4160, 0.7301, 1.1419, 0.7789]x^2))$ |
| | R_5^2 | $IF (x^2 \sim [0.5795, 0.9306, 0.8066]^T)$ $THEN (y_5^2 = \sigma([2.5176, 1.6590, 2.3924, 1.7640]x^2))$ |
| | R_6^2 | $IF (x^2 \sim [0.8898, 0.8796, 0.7991]^T)$ $THEN (y_6^2 = \sigma([1.2757, 0.6794, 1.2152, 0.8184]x^2))$ |

References

- [1] H. J. Rong, N. Sundararajan, G. Bin Huang, and P. Saratchandran, "Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction," *Fuzzy Sets Syst.*, vol. 157, no. 9, pp. 1260–1275, 2006.
- [2] D. Ge and X. J. Zeng, "Learning data streams online - an evolving fuzzy system approach with self-

- learning/adaptive thresholds,” *Inf. Sci. (Ny)*., vol. 507, pp. 172–184, 2020.
- [3] R. N. Patro, S. Subudhi, P. K. Biswal, and F. Dell’Acqua, “Dictionary-based classifiers for exploiting feature sequence information and their application to hyperspectral remotely sensed data,” *Int. J. Remote Sens.*, vol. 40, no. 13, pp. 4996–5024, 2019.
 - [4] D. Li and D. B. Dunson, “Classification via local manifold approximation,” *Biometrika*, vol. 107, no. 4, pp. 1013–1020, 2020.
 - [5] X. Gu and Q. Shen, “A self-adaptive fuzzy learning system for streaming data prediction,” *Inf. Sci. (Ny)*., vol. 579, pp. 623–647, 2021.
 - [6] X. Gu *et al.*, “Multi-objective evolutionary optimisation for prototype-based fuzzy classifiers,” *IEEE Trans. Fuzzy Syst.*, DOI: 10.1109/TFUZZ.2022.3214241, 2022.
 - [7] P. Angelov and X. Zhou, “Evolving fuzzy-rule based classifiers from data streams,” *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1462–1474, 2008.
 - [8] X. Gu and P. Angelov, “Multi-class fuzzily weighted adaptive boosting-based self-organising fuzzy inference ensemble systems for classification,” *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 9, pp. 3722–3735, 2022.
 - [9] J. A. Iglesias, A. Ledezma, and A. Sanchis, “Ensemble method based on individual evolving classifiers,” in *IEEE Conference on Evolving and Adaptive Intelligent Systems*, 2013, pp. 56–61.