Neural Comput & Applic (2013) 23:1987–1998
DOI 10.1007/s00521-012-1140-x

ISNN 2012

Monotonic type-2 fuzzy neural network and its application to thermal comfort prediction

Chengdong Li · Jianqiang Yi · Ming Wang · Guiqing Zhang

Received: 14 April 2012 / Accepted: 22 August 2012 / Published online: 14 September 2012

Abstract This paper studies the monotonic type-2 fuzzy neural network (T2FNN), which can be adopted in many identification and prediction problems where the monotonicity property between the inputs and outputs is required. Sufficient conditions on the parameters of the T2FNN are first presented to ensure the monotonicity between the inputs and outputs. Then, data-driven design model for the monotonic T2FNN is built. Also, under the monotonicity constraints, a hybrid algorithm is provided to optimize the parameters of the monotonic T2FNN. This hybrid algorithm utilizes the constrained least squares method and the penalty function-based gradient descent algorithm to realize reasonable parameter initialization and optimization. At last, an application to the thermal comfort index prediction is given to verify the effectiveness of the monotonic T2FNN. Comparisons with other methods are also made.

Keywords Type-2 fuzzy · Neural network · Monotonicity · Constrained least squares method · Gradient descent algorithm · Thermal comfort

1 Introduction

Fuzzy logic systems [1] that can incorporate human knowledge or experience into system design have been proved to be an effective approach to deal with imprecise information and to solve many engineering problems that cannot be solved by classical methods. However, the conventional ways for fuzzy logic systems design face the difficulties to transform human experience into the rule base and to tune the parameters of the membership functions (MFs) so as to maximize (minimize) the performance index. In order to overcome such drawbacks, several approaches have been developed, one of which is fuzzy neural network (FNN) [2], such as adaptive-network-based fuzzy inference system [3], dynamic FNN [4, 5], genetic dynamic FNN [6], and self-organizing FNN [7]. FNNs that can combine the merits of fuzzy logic systems and neural networks have been widely applied in many real-word applications.

Unfortunately, in real-world environments or applications, there exist high-level uncertainties, which cannot be modeled or handled by classic fuzzy sets (type-1 fuzzy sets: T1FS) or classic fuzzy logic systems (type-1 fuzzy logic systems: T1FLS). In order to tackle this problem, Zadeh [8] proposed the concept of type-2 fuzzy set (T2FS), which is an extension of T1FS. And a complete theory of type-2 fuzzy logic systems (T2FLS) [9–14] has been developed by researchers recently. T2FLS not only has the merits of T1FLS, but also can provide the capability to model high levels of uncertainties and produce more complex input–output mappings and better results [9–14], as T2FLS utilizes T2FSs which can provide additional degrees of freedom and more parameters. But, how to identify the structure of T2FLSs and to estimate the parameters of type-2 MFs are still the main issues associated with T2FLSs. One way to solve this problem is to combine T2FLSs with neural networks.

Type-2 fuzzy neural network (T2FNN), which has advantages of both T2FLS and neural network, has been

C. Li (✉) · M. Wang · G. Zhang
School of Information and Electrical Engineering, Shandong Jianzhu University, Jinan 250101, People's Republic of China
e-mail: lichengdong@sdjzu.edu.cn

J. Yi
Institute of Automation, Chinese Academy of Sciences, Beijing 100190, People's Republic of China

studied and applied in many real-world problems. The design of T2FNN is first described in [15]. The optimal and back-propagation training algorithms of T2FNN are given in [16] and [17]. In [18], a self-evolving T2FNN that can learn its structure and parameters online is presented. In [19], a recurrent T2FNN is explored for dynamic system modeling. In [20], the hierarchical type-2 neuro-fuzzy BSP model is studied. In [21], the authors adopt the fuzzy clustering and differential evolution optimization to construct T2FNN. T2FNN has also found lots of applications, for example, linear ultrasonic motor control [22], coupled-tank liquid level control [23], inverse control of cable-driven parallel mechanism [24], identification and control of time-varying plants [25], speech detection in noisy environments [26], control of permanent magnet linear synchronous motor drives [27], and channel equalization [28]. Most of such T2FNNs are constructed by data-driven methods.

In system identification or prediction problems, it is usually hard to obtain exact physical structure knowledge of some systems. However, some kind of qualitative knowledge of these systems can be observed easily, such as monotonicity, bounded range, and symmetry. Such qualitative knowledge can partly reflect the characteristics of the unknown systems. Hence, we can make full use of the information from the qualitative knowledge to achieve better performance when data-driven methods are used for system identification or prediction problems. One particular but common qualitative knowledge is the monotonicity property between the inputs and outputs. For example, in the water heating system, the temperature of water will change with respect to the heat power monotonically. In recent years, several researches [29–36] have studied how to incorporate the monotonicity property into T1FLSs. However, for the monotonicity of T2FLSs, there has been only a few studies till now because of the complexity of the input–output mappings. In [37, 38], we have studied the parameter conditions for the monotonicity of T2FLS. But, we have only considered single-input T2FLSs or SIRMs connected T2FLSs, which essentially are linear combinations of single-input T2FLSs. To the best of the authors' knowledge, there is no work concerning the parameter conditions and design of monotonic FNNs including both the type-2 and type-1 cases. To be more practical, more works need to be done on FNNs.

In this paper, the theoretical analysis, design, and optimization issues of the monotonic FNNs are studied. As type-1 fuzzy neural networks (T1FNNs) are special cases of T2FNNs, in this study, we only take T2FNNs into account. Similar results can be extended to the type-1 case readily. First, sufficient conditions on the parameters of T2FNNs are derived to ensure the monotonicity between the inputs and outputs. And then, data-driven model for the design of monotonic T2FNNs is presented. In order to design satisfactory monotonic T2FNNs, a hybrid algorithm that is a combination of the constrained least squares method and the penalty function-based gradient descent algorithm is provided to optimize the parameters of the T2FNNs. Finally, to verify the effectiveness of the monotonic T2FNNs, we apply the monotonic T2FNNs to realize the prediction of the thermal comfort index. Also, comparisons with other methods are made. Simulation result and comparisons demonstrate that the monotonic T2FNN can achieve satisfactory performance and performs better than its counterpart—T1FNN.

The primary contributions of this work are summarized as follows:

1. Parameter conditions of T2FNNs are derived to ensure the monotonicity property between their inputs and outputs. Such conditions are useful for the design of monotonic T2FNNs. Also, the derived conditions are for multi-input T2FNNs. The parameter conditions for multi-input T2FNNs include but cannot be extended from existing single-input results, since the existing results are for the iterative Karnik–Mendel type-reduction method–based T2FLSs, the input–output mappings of which do not have closed-form expressions. For this reason, it is a hard task to derive monotonicity conditions for the multi-input T2FLSs that adopt the Karnik–Mendel type-reduction method. But the type-reduction method adopted in this study is the Begian–Melek–Mendel (BMM) method [12]. With the BMM method, the input–output mappings of multi-input T2FLSs or T2FNNs have closed-form expressions. This makes it possible to obtain the monotonicity conditions.
2. The mathematical model for the data-driven design of monotonic T2FNNs is built. As pointed by this study, the data-driven design of monotonic T2FNNs can be seen as the constrained nonlinear optimization problems, while the data-driven design of ordinary T2FNNs can be seen as the nonconstrained nonlinear optimization problems.
3. Optimization algorithm is given for parameter learning of monotonic T2FNNs. To the best of the authors' knowledge, most monotonicity studies are about parameter conditions, and there exists no work on the parameter optimization and design problems of monotonic FNNs.
4. Application of the monotonic T2FNNs to the prediction of the thermal comfort index is made. This application demonstrates the usefulness of the monotonicity property.

2 Type-2 fuzzy neural network

In this paper, we consider the general multi-input–single-output system, whose input variables are supposed to be $\boldsymbol{x} = (x_1, \ldots, x_p) \in X_1 \times X_2 \times \cdots \times X_p$.

By assigning the jth input variable N_j T2FSs, we can obtain the following fuzzy rule base with $\prod_{j=1}^{p} N_j$ fuzzy rules:

Rule$(i_1, i_2, \ldots, i_p)$: $x_1 = \widetilde{A}_1^{i_1}$, $x_2 = \widetilde{A}_2^{i_2}$, …, $x_p = \widetilde{A}_p^{i_p} \rightarrow y_o(\boldsymbol{x}) = [\underline{w}^{i_1 i_2 \cdots i_p}, \overline{w}^{i_1 i_2 \cdots i_p}]$, where $i_j = 1, 2 \ldots, N_j$, $[\underline{w}^{i_1 i_2 \cdots i_p}, \overline{w}^{i_1 i_2 \cdots i_p}]$s are the interval weights, $\widetilde{A}_j^{i_j}$s are T2FSs for the jth input variable.

Corresponding to this type-2 fuzzy rule base, the structure of the T2FNN can be constructed as shown in Fig. 1. The T2FNN works as follows in each layer.

Layer 1-fuzzification layer: There are p nodes in this layer. For analysis simplicity, singleton fuzzifier is adopted in this layer.

Layer 2-type-2 MF layer: In this layer, each node performs a T2FS, and there are $\sum_{j=1}^{p} N_j$ nodes in this layer. With the choice of Gaussian T2FS (see Fig. 2), the T2FS $\widetilde{A}_j^{i_j}$ can be represented as an interval bound by its lower MF (LMF) $\underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j)$ and upper MF (UMF) $\overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j)$:

$$\underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j) = \exp\left[-\frac{1}{2}\frac{(x_j - c_j^{i_j})^2}{(\underline{\delta}_j^{i_j})^2}\right], \tag{1}$$

$$\overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j) = \exp\left[-\frac{1}{2}\frac{(x_j - c_j^{i_j})^2}{(\overline{\delta}_j^{i_j})^2}\right], \tag{2}$$

where $c_j^{i_j}$ and $[(\underline{\delta}_j^{i_j})^2, (\overline{\delta}_j^{i_j})^2]$ are, respectively, the center and uncertain widths of $\widetilde{A}_j^{i_j}$, and $0 < (\underline{\delta}_j^{i_j})^2 \le (\overline{\delta}_j^{i_j})^2$.

The output of each node can be represented as an interval $\left[\underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j), \overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j)\right]$.

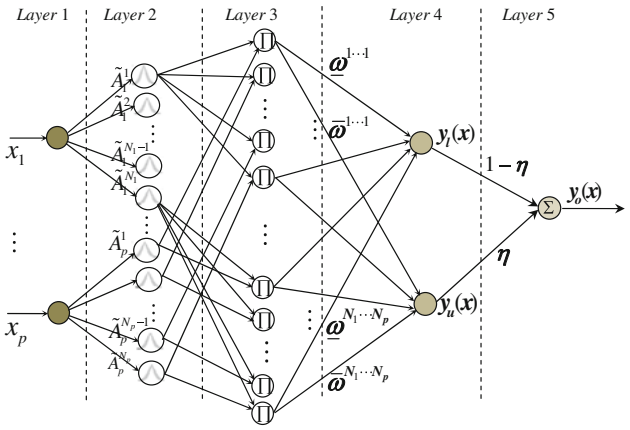

Fig. 1 Structure of type-2 fuzzy neural network

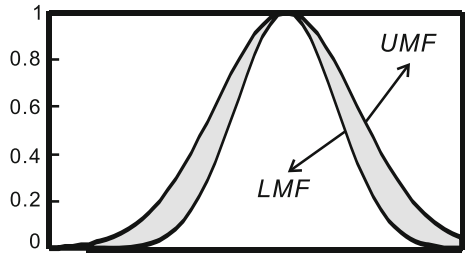

Fig. 2 A Gaussian T2FS with uncertain widths

Layer 3- Rule Layer: Each node in this layer represents one fuzzy logic rule and performs precondition matching of a rule. So, there are $\prod_{j=1}^{p} N_j$ nodes in this layer. The output of a rule node represents the firing strength of this fuzzy rule. For the node $(i_1, i_2, \ldots, i_p)$ corresponding to Rule $(i_1, i_2, \ldots, i_p)$, its firing strength is calculated by the product operation as follows:

$$F^{i_1 i_2 \cdots i_p}(\boldsymbol{x}) = \left[\prod_{j=1}^{p} \underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j), \prod_{j=1}^{p} \overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j)\right]. \tag{3}$$

Layer 4-type-reduction layer: This layer is used to achieve the type-reduction. In this layer, different type-reducers may give different results. For simplicity, we adopt Begian–Melek–Mendel (BMM) method proposed in [12] to realize the type-reduction. With this type-reduction method, the input–output mappings of T2FNNs have closed-form expressions, which makes it convenient to do theoretical analysis. Using the BMM method, the output of the two nodes in the fourth layer can be computed as:

$$y_l(\boldsymbol{x}) = \frac{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} \underline{w}^{i_1 i_2 \cdots i_p} \prod_{j=1}^{p} \underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j)}{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} \prod_{j=1}^{p} \underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j)}, \tag{4}$$

$$y_u(\boldsymbol{x}) = \frac{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} \overline{w}^{i_1 i_2 \cdots i_p} \prod_{j=1}^{p} \overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j)}{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} \prod_{j=1}^{p} \overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j)}. \tag{5}$$

Layer 5-output layer: This layer performs the defuzzification. Here, we use the linear combination of $y_l(\boldsymbol{x})$ and $y_u(\boldsymbol{x})$ to generate the crisp output, that is,

$$y_o(\boldsymbol{x}) = (1 - \eta) y_l(\boldsymbol{x}) + \eta y_u(\boldsymbol{x}), \tag{6}$$

where η is the defuzzification coefficient, and $0 \le \eta \le 1$.

A T2FNN can be seen as a multivariable function $y_o(\boldsymbol{x})$. When all sources of uncertainty disappear, the T2FSs $\widetilde{A}_j^{i_j}$s in Layer 2 becomes T1FSs $A_j^{i_j}$s, and the interval weights $[\underline{w}^{i_1 i_2 \cdots i_p}, \overline{w}^{i_1 i_2 \cdots i_p}]$s between Layer 3 and Layer 4 becomes crisp weights $w^{i_1 i_2 \cdots i_p}$s. Hence, the T2FNN turns to a T1FNN.

3 Monotonicity of type-2 fuzzy neural network

In this section, we will derive the conditions on the parameters of T2FNN to ensure the monotonicity property between its input and output. To begin, let us give the definition of monotonicity for this special multivariable function first.

Definition 1 (*Monotonic T2FNN*) A T2FNN is said to be monotonically increasing with respect to (w.r.t) x_k, if $\forall x_k^1 \le x_k^2 \in X_k$ implies that $y_o(x_1,\ldots,x_k^1,\ldots,x_p) \le y_o(x_1,\ldots,x_k^2,\ldots,x_p)$ for all combinations of $(x_1,\ldots,x_{k-1},x_{k+1},\ldots,x_p)$. And, a T2FNN is said to be monotonically decreasing with respect to (w.r.t) x_k, if $\forall x_k^1 \le x_k^2 \in X_k$ implies that $y_o(x_1,\ldots,x_k^1,\ldots,x_p) \ge y_o(x_1,\ldots,x_k^2,\ldots,x_p)$ for all combinations of $(x_1,\ldots,x_{k-1},x_{k+1},\ldots,x_p)$.

In this study, we just take the increasing monotonicity into account. Similar results can be obtained for decreasing monotonicity. For the increasing monotonicity, we have the following results:

Theorem 1 *The T2FNN is monotonically increasing w.r.t x_k if the following conditions can be satisfied*:

1. *the T2FSs in layer 2 satisfy that* $\frac{\partial \underline{\mu}_{A_k^l}(x_k)}{\partial x_k}\cdot\underline{\mu}_{A_k^m}(x_k) \le \frac{\partial \underline{\mu}_{A_k^m}(x_k)}{\partial x_k}\cdot\underline{\mu}_{A_k^l}(x_k)$ *and* $\frac{\partial \overline{\mu}_{A_k^l}(x_k)}{\partial x_k}\cdot\overline{\mu}_{A_k^m}(x_k) \le \frac{\partial \overline{\mu}_{A_k^m}(x_k)}{\partial x_k}\cdot\overline{\mu}_{A_k^l}(x_k)$, *where* $1 \le l < m \le N_k$;
2. *the weighting factors between Layer 3 and Layer 4 satisfy that* $\underline{w}^{i_1\ldots i_{k-1}i_k i_{k+1}\ldots i_p} \le \underline{w}^{i_1\ldots i_{k-1}(i_k+1)i_{k+1}\ldots i_p}$, $\overline{w}^{i_1\ldots i_{k-1}i_k i_{k+1}\ldots i_p} \le \overline{w}^{i_1\ldots i_{k-1}(i_k+1)i_{k+1}\ldots i_p}$ *for all combinations of* $(i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_p)$, *where* $i_k = 1,\ldots,N_k-1$.

Proof Suppose that $\boldsymbol{x}^1 = (x_1,\ldots,x_k^1,\ldots,x_p) \le \boldsymbol{x}^2 = (x_1,\ldots,x_k^2,\ldots,x_p)$.

Note that

$$
\begin{aligned}
y_l(\boldsymbol{x}) &= \frac{\sum_{i_1=1}^{N_1}\cdots\sum_{i_p=1}^{N_p}\underline{w}^{i_1 i_2\cdots i_p}\prod_{j=1}^{p}\underline{\mu}_{A_j^{i_j}}(x_j)}{\sum_{i_1=1}^{N_1}\cdots\sum_{i_p=1}^{N_p}\prod_{j=1}^{p}\underline{\mu}_{A_j^{i_j}}(x_j)} \\
&= \frac{\sum_{i_1=1}^{N_1}\cdots\sum_{i_{k-1}=1}^{N_{k-1}}\sum_{i_{k+1}=1}^{N_{k+1}}\cdots\sum_{i_p=1}^{N_p}[\prod_{j=1,j\neq k}^{p}\underline{\mu}_{A_j^{i_j}}(x_j)}{\sum_{i_1=1}^{N_1}\cdots\sum_{i_{k-1}=1}^{N_{k-1}}\sum_{i_{k+1}=1}^{N_{k+1}}\cdots\sum_{i_p=1}^{N_p}[\prod_{j=1,j\neq k}^{p}\underline{\mu}_{A_j^{i_j}}(x_j)} \\
&\quad \frac{\sum_{i_k=1}^{N_k}\underline{w}^{i_1 i_2\cdots i_p}\underline{\mu}_{A_k^{i_k}}(x_k)]}{\sum_{i_k=1}^{N_k}\underline{\mu}_{A_k^{i_k}}(x_k)]} \\
&= \frac{\sum_{v=1}^{M}\alpha^v\sum_{i_k=1}^{N_k}\underline{w}^{vi_k}\underline{\mu}_{A_k^{i_k}}(x_k)}{\sum_{v=1}^{M}\alpha^v\sum_{i_k=1}^{N_k}\underline{\mu}_{A_k^{i_k}}(x_k)},
\end{aligned}
\tag{7}
$$

where $M = \prod_{j=1,j\neq k}^{p} N_k$, $v = v(i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_p)$ is a combination of $i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_p$, and $\alpha^v = \alpha^{v(i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_p)} = \prod_{j=1,j\neq k}^{p}\underline{\mu}_{A_j^{i_j}}(x_j)$.

Then, we have

$$
\begin{aligned}
y_l(\boldsymbol{x}^2) - y_l(\boldsymbol{x}^1) &= \frac{\sum_{s=1}^{M}\alpha^s\sum_{l=1}^{N_k}\underline{w}^{sl}\underline{\mu}_{A_k^l}(x_k^2)}{\sum_{s=1}^{M}\alpha^s\sum_{l=1}^{N_k}\underline{\mu}_{A_k^l}(x_k^2)} \\
&\quad - \frac{\sum_{t=1}^{M}\alpha^t\sum_{m=1}^{N_k}\underline{w}^{tm}\underline{\mu}_{A_k^m}(x_k^1)}{\sum_{t=1}^{M}\alpha^t\sum_{m=1}^{N_k}\underline{\mu}_{A_k^m}(x_k^1)} \\
&= \frac{1}{\varepsilon}\left[\sum_{s=1}^{M}\sum_{t=1}^{M}\alpha^s\alpha^t\sum_{l=1}^{N_k}\sum_{m=1}^{N_k}\underline{w}^{sl}\underline{\mu}_{A_k^l}(x_k^2)\underline{\mu}_{A_k^m}(x_k^1)\right. \\
&\quad \left. - \sum_{s=1}^{M}\sum_{t=1}^{M}\alpha^s\alpha^t\sum_{l=1}^{N_k}\sum_{m=1}^{N_k}\underline{w}^{tm}\underline{\mu}_{A_k^l}(x_k^2)\underline{\mu}_{A_k^m}(x_k^1)\right]
\end{aligned}
\tag{8}
$$

where

$$
\varepsilon = \left[\sum_{s=1}^{M}\alpha^s\sum_{l=1}^{N_k}\underline{\mu}_{A_k^l}(x_k^2)\right]\left[\sum_{t=1}^{M}\alpha^t\sum_{m=1}^{N_k}\underline{\mu}_{A_k^m}(x_k^1)\right] > 0.
$$

Consider the following fact

$$
\begin{aligned}
&\sum_{s=1}^{M}\sum_{t=1}^{M}\alpha^s\alpha^t\sum_{l=1}^{N_k}\sum_{m=1}^{N_k}\underline{w}^{tm}\underline{\mu}_{A_k^l}(x_k^2)\underline{\mu}_{A_k^m}(x_k^1) \\
&\quad = \sum_{s=1}^{M}\sum_{t=1}^{M}\alpha^s\alpha^t\sum_{l=1}^{N_k}\sum_{m=1}^{N_k}\underline{w}^{sm}\underline{\mu}_{A_k^l}(x_k^2)\underline{\mu}_{A_k^m}(x_k^1)
\end{aligned}
\tag{9}
$$

Substituting (9) into (8) leads to

$$
\begin{aligned}
&y_l(\boldsymbol{x}^2) - y_l(\boldsymbol{x}^1) \\
&\quad = \frac{1}{\varepsilon}\left[\sum_{s=1}^{M}\sum_{t=1}^{M}\alpha^s\alpha^t\sum_{l=1}^{N_k}\sum_{m=1}^{N_k}(\underline{w}^{sl}-\underline{w}^{sm})\underline{\mu}_{A_k^l}(x_k^2)\underline{\mu}_{A_k^m}(x_k^1)\right] \\
&\quad = \frac{1}{\varepsilon}\left[\sum_{s=1}^{M}\sum_{t=1}^{M}\alpha^s\alpha^t\sum_{l=1}^{N_k}\sum_{m=l+1}^{N_k}(\underline{w}^{sm}-\underline{w}^{sl})\cdot\right. \\
&\quad \left.\left(\underline{\mu}_{A_k^m}(x_k^2)\underline{\mu}_{A_k^l}(x_k^1) - \underline{\mu}_{A_k^l}(x_k^2)\underline{\mu}_{A_k^m}(x_k^1)\right)\right].
\end{aligned}
\tag{10}
$$

The first condition implies that $\frac{\partial}{\partial x_k}\left[\frac{\underline{\mu}_{A_k^m}(x_k)}{\underline{\mu}_{A_k^l}(x_k)}\right] \ge 0$, which means that $\frac{\underline{\mu}_{A_k^m}(x_k)}{\underline{\mu}_{A_k^l}(x_k)}$ is monotonically increasing w.r.t x_k.

Hence,

$$
\frac{\underline{\mu}_{A_k^m}(x_k^2)}{\underline{\mu}_{A_k^l}(x_k^2)} - \frac{\underline{\mu}_{A_k^m}(x_k^1)}{\underline{\mu}_{A_k^l}(x_k^1)} \ge 0,
\tag{11}
$$

which implies that

$$
\underline{\mu}_{A_k^m}(x_k^2)\underline{\mu}_{A_k^l}(x_k^1) - \underline{\mu}_{A_k^l}(x_k^2)\underline{\mu}_{A_k^m}(x_k^1) \ge 0.
\tag{12}
$$

From the second condition, for $m > l$

$$\underline{w}^{sm} - \underline{w}^{sl} \geq 0. \tag{13}$$

From (10), (12), and (13), we can observe that $y_l(\boldsymbol{x}^2) \geq y_l(\boldsymbol{x}^1)$.

In the similar way, we can also obtain that $y_u(\boldsymbol{x}^2) \geq y_u(\boldsymbol{x}^1)$.

Obviously, this theorem holds, as the output of the T2FNN is a linear combination of $y_l(\boldsymbol{x})$ and $y_u(\boldsymbol{x})$.

In [37], we have derived parameter conditions of triangular, trapezoidal, and Gaussian T2FSs to satisfy that $\frac{\partial \underline{\mu}_{\underline{A}_k^l}(x_k)}{\partial x_k} \underline{\mu}_{\underline{A}_k^m}(x_k) \leq \frac{\partial \underline{\mu}_{\underline{A}_k^m}(x_k)}{\partial x_k} \underline{\mu}_{\underline{A}_k^l}(x_k)$ and $\frac{\partial \overline{\mu}_{\underline{A}_k^l}(x_k)}{\partial x_k} \cdot \overline{\mu}_{\underline{A}_k^m}(x_k) \leq \frac{\partial \overline{\mu}_{\underline{A}_k^m}(x_k)}{\partial x_k} \cdot \overline{\mu}_{\underline{A}_k^l}(x_k)$. The related conclusion about Gaussian T2FSs (see Fig. 2) is listed as follows:

For Gaussian T2FSs , if $c_k^l \leq c_k^m$, $(\underline{\delta}_k^l)^2 = (\underline{\delta}_k^m)^2 = \underline{\delta}_k^2$ and $(\overline{\delta}_k^l)^2 = (\overline{\delta}_k^m)^2 = \overline{\delta}_k^2$, then the first condition in Theorem 1 can be met. In other words, when Gaussian T2FSs are adopted in Layer 2, the first condition in Theorem 1 can be satisfied, if $c_k^1 \leq c_k^2 \cdots \leq c_k^{N_k}$ for $k = 1, 2, \ldots p$ and the uncertain widths of Gaussian T2FSs for each input variable are set to be the same.

In this section, sufficient parameter constraints of T2FNNs are derived. These monotonicity conditions provide the basis for the design of monotonic T2FNNs.

4 Data-driven design model for monotonic T2FNN

This section tries to build the mathematical model for the data-driven design of monotonic T2FNNs. In our study, the conditions in Theorems 1 are used to constrain the parameters of T2FNNs, and the data are utilized to train and optimize such parameters further.

Let us denote θ as the vector of the constrained parameters appearing in the two conditions in Theorem 1. If Gaussian T2FSs are adopted, θ will be the vector of the centers of Gaussian T2FSs in Layer 2 and the weights between Layer 3 and Layer 4.

From the conclusions in the previous section, if Gaussian T2FSs are adopted, then the inequality constraints on the centers of T2FSs in Layer 2 and the interval weights between Layer 3 and Layer 4 are linear. As a result, θ satisfies the linear inequality below:

$$A\theta = \begin{bmatrix} a_1^T\theta \\ a_2^T\theta \\ \vdots \\ a_L^T\theta \end{bmatrix} \geq 0 \tag{14}$$

where L is the number of the inequalities for the monotonicity constraint.

Suppose that there are N input–output data points $(\boldsymbol{x}^t, y^t) = (x_1^t, x_2^t, \ldots, x_p^t, y^t), (t = 1, 2, \ldots, N)$. And, the training criteria are chosen to minimize the following squared error function:

$$E = \sum_{t=1}^{N} (y_o(\boldsymbol{x}^t, \Theta) - y^t)^2 \tag{15}$$

where $y_o(\boldsymbol{x}^t, \Theta)$ is the output of the T2FNN. Θ is the set or vector of all the parameters that need to be tuned.

Therefore, the data-driven design of monotonic T2FNNs can be realized by solving the following constrained optimization problem:

$$\begin{cases} \min_{\Theta} \sum_{t=1}^{N} (y_o(\boldsymbol{x}^t, \Theta) - y^t)^2 \\ \text{subject to} \quad A\theta \geq 0. \end{cases} \tag{16}$$

As the outputs of the T2FNNs are linear with respect to the weights between Layer 3 and Layer 4, but nonlinear with respect to the centers and widths of Gaussian T2FSs in Layer 2, this optimization problem is one nonlinear optimization problem. Such a problem can be solved using classical constrained nonlinear optimization algorithms, for example, the penalty function method, or using the evolutionary computation algorithms, such as genetic algorithms and particle swarm algorithms.

Based on above discussion, we provide the following guidelines for the data-driven design of monotonic T2FNNs:

1. Set reasonable initial centers and widths of Gaussian T2FSs in Layer 2 and the weights between Layer 3 and Layer 4.
2. Optimize all or part of the parameters of the T2FNNs under the monotonicity constraints using classical constrained nonlinear optimization algorithms or the evolutionary computation algorithms.

In the following section, we will provide a hybrid algorithm to optimize the parameters of the monotonic T2FNN.

5 Parameter learning of monotonic T2FNN

To achieve better performance and obtain precise identification or prediction result, this section provides a hybrid algorithm to tune the parameters of the monotonic T2FNN. In this hybrid algorithm, all the parameters of the T2FNN are learned through the gradient descent algorithm after parameter initialization. As well known, the gradient descent algorithm that is the basis of the back-propagation algorithm is sensitive to initial values. So it is very

important to choose reasonable initial parameters. Below, we will first discuss the parameter initialization problem and then derive the gradient descent algorithm for the parameter learning of the monotonic T2FNN.

5.1 Parameter initialization

For the T2FNN, the parameters that need to be optimized are the parameters of T2FSs in Layer 2 (e.g., the centers and uncertain widths of the Gaussian T2FSs), the interval weights between Layer 3 and Layer 4, and the defuzzification coefficient η.

Usually, we can choose 0.5 as the initial value of the defuzzification coefficient η. And, it is easy to set reasonable initial parameters of the T2FSs in Layer 2 through intuitive fuzzy partition according to background knowledge or by clustering algorithms. But it is relatively difficult to determine reasonable initial values for the interval weights between Layer 3 and Layer 4. Fortunately, the output of the T2FNN is linear with such weights, so we can utilize the least squares method to realize their initialization. Below, we will discuss this issue in detail.

From (4)–(6),

$$y_o(\boldsymbol{x}) = \boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{w}, \tag{17}$$

where $\boldsymbol{w} = [\underline{w}^{11\ldots11}, \underline{w}^{11\ldots12}, \ldots, \underline{w}^{11\ldots1N_p}, \underline{w}^{11\ldots21}, \ldots \underline{w}^{11\ldots2N_p}, \ldots, \underline{w}^{N_1N_2\ldots N_{p-1}N_p}, \overline{w}^{11\ldots11}, \overline{w}^{11\ldots12}, \ldots, \overline{w}^{11\ldots1N_p}, \overline{w}^{11\ldots21}, \ldots, \overline{w}^{11\ldots2N_p}, \ldots, \overline{w}^{N_1N_2\ldots N_{p-1}N_p}]^{\mathrm{T}}$ is a $\prod_{j=1}^{p} N_j$ vector, and the orders of $\underline{\boldsymbol{\phi}}^{i_1i_2\cdots i_p}(\boldsymbol{x})$ and $\overline{\boldsymbol{\phi}}^{i_1i_2\cdots i_p}(\boldsymbol{x})$ in the vector $\boldsymbol{\phi}(\boldsymbol{x})$ are the same as the orders of the elements in $\boldsymbol{w}$. Here,

$$\underline{\boldsymbol{\phi}}^{i_1i_2\ldots i_p}(\boldsymbol{x}) = (1-\eta)\frac{\prod_{j=1}^{p}\underline{\mu}_{\underset{\sim}{A}_j^{i_j}}(x_j)}{\sum_{i_1=1}^{N_1}\cdots\sum_{i_p=1}^{N_p}\prod_{j=1}^{p}\underline{\mu}_{\underset{\sim}{A}_j^{i_j}}(x_j)}, \tag{18}$$

$$\overline{\boldsymbol{\phi}}^{i_1i_2\ldots i_p}(\boldsymbol{x}) = \eta\frac{\prod_{j=1}^{p}\overline{\mu}_{\underset{\sim}{A}_j^{i_j}}(x_j)}{\sum_{i_1=1}^{N_1}\cdots\sum_{i_p=1}^{N_p}\prod_{j=1}^{p}\overline{\mu}_{\underset{\sim}{A}_j^{i_j}}(x_j)}. \tag{19}$$

From (17), the output $y_o(\boldsymbol{x})$ of the T2FNN is linear with the interval weights between Layer 3 and Layer 4.

Once the T2FSs in the second layer of the T2FNN are determined by initial type-2 fuzzy partition, given N pairs of input–output training data $(\boldsymbol{x}^t, y^t) = (x_1^t, x_2^t, \ldots, x_p^t, y^t)$, $(t = 1, 2, \ldots, N)$, reasonable initial interval weights between Layer 3 and Layer 4 should minimize the following squared error criteria under the second constraint in Theorem 1

$$\begin{aligned} E &= \sum_{t=1}^{N}(y_o(\boldsymbol{x}^t) - y^t)^2 = \sum_{t=1}^{N}(\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{x}^t)\boldsymbol{w} - y^t)^2 \\ &= (\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y})^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}), \end{aligned} \tag{20}$$

where

$$\boldsymbol{y} = [y^1, y^2, \ldots, y^N]^{\mathrm{T}}, \tag{21}$$

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}(\boldsymbol{x}^1), \boldsymbol{\phi}(\boldsymbol{x}^2), \ldots, \boldsymbol{\phi}(\boldsymbol{x}^N)]^{\mathrm{T}}. \tag{22}$$

The linear constraints on the interval weights between Layer 3 and Layer 4 of the monotonic T2FNN can be rewritten in the matrix form as: $\mathbf{P}\boldsymbol{w} \geq 0$.

From above discussion, reasonable initial weights between Layer 3 and Layer 4 can be obtained by solving the following linear-inequality constrained least squares optimization problem:

$$\begin{cases} \min_{\boldsymbol{w}} \quad (\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y})^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}) \\ \text{subject to } \mathbf{P}\boldsymbol{w} \geq 0 \end{cases} \tag{23}$$

Many algorithms can be used to solve this problem, for example, the MATLAB function *lsqlin*. Detailed materials about constrained least squares method can be found in [39].

5.2 Gradient descent learning of monotonic T2FNN

After all the parameters are initialized reasonably, to achieve better performance, we will optimize such parameters further in this subsection.

Given the input–output training data $(\boldsymbol{x}^t, y^t)$, the optimization algorithm is used to adjust all the parameters Θ of the T2FNN by optimizing the following problem:

$$\begin{cases} \min_{\Theta} J(t) = \dfrac{1}{2}e^2(t) = \dfrac{1}{2}[y_o(\boldsymbol{x}^t) - y^t]^2 \\ \text{subject to } A\theta \geq 0 \end{cases} \tag{24}$$

Through the penalty function method, this constrained optimization problem can be transformed to the following unconstrained optimization problem:

$$\min_{\Theta} \quad J(t) = \frac{1}{2}e^2(t) + \lambda L(\theta) \tag{25}$$

where $L(\theta) = \sum_{i=1}^{L}[\min\{0, a_i^T\theta\}]^2$ is the penalty item and λ is a large number.

To solve such optimization problem, the gradient descent algorithm can be used. Below, we will present the parameter update rules for all the parameters of the monotonic T2FNN, in which Gaussian T2FSs are adopted.

For any parameter $\theta_k \in \theta$, its update rule is

$$\begin{aligned} \theta_k(t+1) &= \theta_k(t) - \alpha_{\theta_k}\frac{\partial J(t)}{\partial \theta_k} \\ &= \theta_k(t) - \alpha_{\theta_k}e(t)\frac{\partial y_o(\boldsymbol{x}^t)}{\partial \theta_k} - \lambda\alpha_{\theta_k}\frac{\partial L(\theta)}{\partial \theta_k}, \end{aligned} \tag{26}$$

where α_{θ_k} is the learning rate, and $\frac{\partial L(\theta)}{\partial \theta_k}$ can be computed as

$$\frac{\partial L(\theta)}{\partial \theta_k} = \sum_{i\in I} 2\frac{\partial a_i^T\theta}{\partial \theta_k} = 2\sum_{i\in I} a_{i,k} \tag{27}$$

in which $I = \{j | a_j^T \theta < 0, \& j = 1, \ldots, L\}$.

For any other parameter $\phi \in \Theta - \theta$, its update rule is

$$\phi(t+1) = \phi(t) - \alpha_\phi \frac{\partial J(t)}{\partial \phi} = \phi(t) - \alpha_\phi e(t) \frac{\partial y_o(\boldsymbol{x}^t)}{\partial \phi} \quad (28)$$

From (26) and (28), we can see that the most important thing to obtain the update rules for parameters of the monotonic T2FNN is to compute the derivatives $\frac{\partial y_o(\boldsymbol{x}^t)}{\partial \theta_k}$ and $\frac{\partial y_o(\boldsymbol{x}^t)}{\partial \phi}$.

5.2.1 Derivatives for the weights between Layer 3 and Layer 4

As the weights between Layer 3 and Layer 4 are constrained in the second condition in Theorem 1, the update rules for these parameters should be chosen as the one in (26), and

$$\begin{aligned} \frac{\partial y_o(\boldsymbol{x}^t)}{\partial \underline{w}^{i_1 i_2 \cdots i_p}} &= (1 - \eta(t)) \frac{\partial y_l(\boldsymbol{x}^t)}{\partial \underline{w}^{i_1 i_2 \cdots i_p}} \\ &= (1 - \eta(t)) \frac{\prod_{j=1}^p \underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} \prod_{j=1}^p \underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}, \end{aligned} \quad (29)$$

$$\begin{aligned} \frac{\partial y_o(\boldsymbol{x}^t)}{\partial \overline{w}^{i_1 i_2 \cdots i_p}} &= \eta(t) \frac{\partial y_u(\boldsymbol{x}^t)}{\partial \overline{w}^{i_1 i_2 \cdots i_p}} \\ &= \eta(t) \frac{\prod_{j=1}^p \overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} \prod_{j=1}^p \overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}. \end{aligned} \quad (30)$$

5.2.2 Derivative for the defuzzification coefficient η between Layer 4 and Layer 5

As η is not constrained in Theorem 1, the update rules for this parameter should be chosen as the one in (28), and

$$\frac{\partial y_o(\boldsymbol{x}^t)}{\partial \eta} = y_u(\boldsymbol{x}^t) - y_l(\boldsymbol{x}^t) \quad (31)$$

5.2.3 Derivatives for the centers of Gaussian T2FSs in Layer 2

As the centers of Gaussian T2FSs in Layer 2 are constrained in the first condition in Theorem 1, the update rules for these parameters should be chosen as the one in (26), and

$$\frac{\partial y_o(\boldsymbol{x}^t)}{\partial c_j^{i_j}} = (1 - \eta) \frac{\partial y_l(\boldsymbol{x}^t)}{\partial c_j^{i_j}} + \eta \frac{\partial y_u(\boldsymbol{x}^t)}{\partial c_j^{i_j}}, \quad (32)$$

where

$$\begin{aligned} &\frac{\partial y_l(\boldsymbol{x}^t)}{\partial c_j^{i_j}} \\ &= \frac{\sum_{\substack{q=1 \\ q \neq j}}^p \sum_{i_q=1}^{N_q} (\underline{w}^{i_1 \cdots i_p} - y_l(\boldsymbol{x}^t)) \frac{\partial \underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}{\partial c_j^{i_j}} \prod_{\substack{k=1 \\ k \neq j}}^p \underline{\mu}_{\widetilde{A}_k^{i_k}}(x_k^t)}{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} \prod_{j=1}^p \underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}, \end{aligned} \quad (33)$$

$$\begin{aligned} &\frac{\partial y_u(\boldsymbol{x}^t)}{\partial c_j^{i_j}} \\ &= \frac{\sum_{\substack{q=1 \\ q \neq j}}^p \sum_{i_q=1}^{N_q} (\overline{w}^{i_1 \cdots i_p} - y_u(\boldsymbol{x}^t)) \frac{\partial \overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}{\partial c_j^{i_j}} \prod_{\substack{k=1 \\ k \neq j}}^p \overline{\mu}_{\widetilde{A}_k^{i_k}}(x_k^t)}{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} \prod_{j=1}^p \overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}. \end{aligned} \quad (34)$$

in which $\sum_{\substack{q=1 \\ q \neq j}}^{p} \sum_{i_q=1}^{N_q} = \sum_{i_1=1}^{N_1} \cdots \sum_{i_{j-1}=1}^{N_{j-1}} \sum_{i_{j+1}=1}^{N_{j+1}} \cdots \sum_{i_p=1}^{N_p}$.

5.2.4 Derivatives for uncertain widths of Gaussian T2FSs in Layer 2

To meet the first condition in Theorem 1, the uncertain widths of Gaussian T2FSs $\widetilde{A}_j^{i_j}$ $(i_j = 1, 2, \ldots, N_j)$ for the same input variable x_j are assumed to be the same, denoted as $[\underline{\delta}_j^2, \overline{\delta}_j^2]$. Also, these parameters are not constrained by inequity equations, so update rules for them should be chosen as the one in (28), and

$$\frac{\partial y_o(\boldsymbol{x}^t)}{\partial \underline{\delta}_j^2} = (1 - \eta(t)) \frac{\partial y_l(\boldsymbol{x}^t)}{\partial \underline{\delta}_j^2}, \quad (35)$$

$$\frac{\partial y_o(\boldsymbol{x}^t)}{\partial \overline{\delta}_j^2} = \eta(t) \frac{\partial y_u(\boldsymbol{x}^t)}{\partial \overline{\delta}_j^2}. \quad (36)$$

where

$$\begin{aligned} &\frac{\partial y_l(\boldsymbol{x}^t)}{\partial \underline{\delta}_j^2} \\ &= \frac{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} (\underline{w}^{i_1 \cdots i_p} - y_l(\boldsymbol{x}^t)) \frac{\partial \underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}{\partial \underline{\delta}_j^2} \prod_{\substack{k=1 \\ k \neq j}}^p \underline{\mu}_{\widetilde{A}_k^{i_k}}(x_k^t)}{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} \prod_{j=1}^p \underline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}, \end{aligned} \quad (37)$$

$$\begin{aligned} &\frac{\partial y_u(\boldsymbol{x}^t)}{\partial \overline{\delta}_j^2} \\ &= \frac{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} (\overline{w}^{i_1 \cdots i_p} - y_u(\boldsymbol{x}^t)) \frac{\partial \overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}{\partial \overline{\delta}_j^2} \prod_{\substack{k=1 \\ k \neq j}}^p \overline{\mu}_{\widetilde{A}_k^{i_k}}(x_k^t)}{\sum_{i_1=1}^{N_1} \cdots \sum_{i_p=1}^{N_p} \prod_{j=1}^p \overline{\mu}_{\widetilde{A}_j^{i_j}}(x_j^t)}. \end{aligned} \quad (38)$$

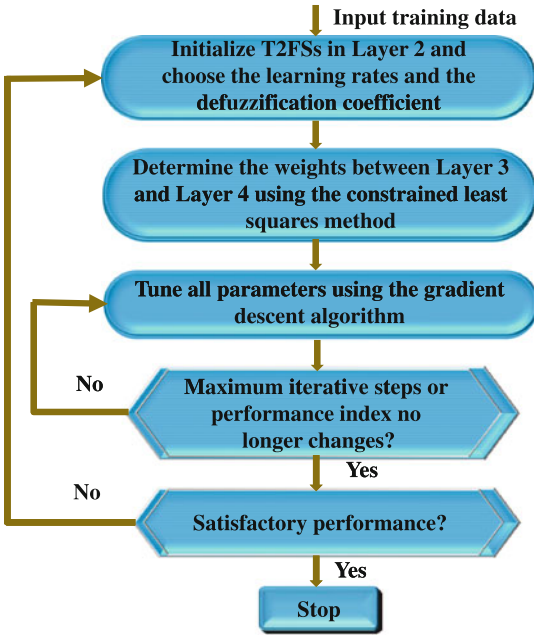

Fig. 3 Hybrid algorithm for the monotonic T2FNN

5.3 Hybrid learning algorithm for monotonic T2FNN

From above discussion, the centers and uncertain widths of Gaussian T2FSs in Layer 2 can be initialized by reasonable type-2 fuzzy partition, while the weights between Layer 3 and Layer 4 can be initialized through the constrained least squares method. After parameter initialization, the penalty function-based gradient descent algorithm can be adopted for learning all the parameters of the monotonic T2FNN. The flowchart of this hybrid algorithm is demonstrated in Fig. 3.

6 Application to the thermal comfort index prediction

In this section, we will demonstrate how to design monotonic T2FNN through applying it to thermal comfort index prediction. Also, comparisons with linear regression method and T1FNN are made to show the superiority of the T2FNN.

6.1 Problem description

To evaluate the thermal comfort, a number of indices have been studied, but the most widely used thermal comfort index is the Predicted Mean Vote (PMV) index proposed by Fanger [40]. Based on the PMV index, ASHRAE (American Society of Heating, Refrigerating and Air-Condition Engineers) suggested to measure the thermal comfort level as: -3 (*cold*), -2 (*cool*), -1 (*slightly cool*), 0 (*neutral*), $+1$ (*slightly warm*), $+2$ (*warm*), $+3$ (*hot*).

The PMV index is a function of six variables, such as air temperature, radiant temperature, relative humidity, air velocity, human activity level, and clothing thermal resistance. The value of PMV ranges from -3 to 3 and can be calculated by [40–43]

$$\begin{aligned} PMV = {} & (0.303e^{-0.036M} + 0.028)\{M - W - 3.05 * 10^{-3} \\ & [5733 - 6.99(M - W) - P_a] - 0.42[(M - W) - 58.15] \\ & - 1.7 * 10^{-5} M(5867 - P_a) - 0.0014M * (34 - t_a) \\ & - 3.96 * 10^{-8} f_{c1}[(t_{c1} + 273)^4 - (t_r + 273)^4] \\ & - f_{c1} h_c (t_{c1} - t_a)\}, \end{aligned} \tag{39}$$

$$\begin{aligned} t_{c1} = {} & 35.7 - 0.0278(M - W) - I_{c1}\{3.96 * 10^{-8} f_{c1} * \\ & [(t_{c1} + 273)^4 - (t_r + 273)^4 - f_{c1} h_c (t_{c1} - t_a)]\}. \end{aligned} \tag{40}$$

where t_{c1}, h_c, f_{c1} and P_a can be computed, respectively, as

$$h_c = \begin{cases} 2.38(t_{c1} - t_a)^{0.25} & 2.38(t_{c1} - t_a)^{0.25} > 12.1\sqrt{v_a} \\ 12.1\sqrt{v_a} & 2.38(t_{c1} - t_a)^{0.25} < 12.1\sqrt{v_a} \end{cases} \tag{41}$$

$$f_{c1} = \begin{cases} 1.00 + 0.2I_{c1} & I_{c1} < 0.5clo \\ 1.05 + 0.1I_{c1} & I_{c1} > 0.5clo \end{cases} \tag{42}$$

$$P_a = \frac{P_s R_H}{100} \tag{43}$$

In above equations, *PMV* is the Predicted Mean Vote, M is the human metabolic rate (W/m^2), W is the external work (W/m^2), P_a is the water vapor pressure (Pa), t_a is the indoor air temperature (°C), t_r is the radiation temperature (°C), I_{c1} is the thermal resistance of clothing (clo), v_a is the relative air velocity (m/s), t_{c1} is the surface temperature of clothing, R_H is the relative humidity in percent, h_c is the convectional heat transfer coefficient (W/m^{2}K), f_{c1} is the ratio of clothed body surface area to nude body surface area, and P_s is the saturated vapor pressure at specific temperature (°C).

From the above PMV calculation equations, we can observe that such equations are nonlinear and rather complicated. Also, we need computing t_{c1} iteratively to obtain the root of the nonlinear equation (40). This is a problem for real-time applications. A promising way to solve this problem is to use a prediction model to approximate the input–output characteristic of the PMV model. In this study, we present one T2FNN prediction model for this PMV index.

The PMV index has six input variables, and, sometimes, some input variables are rather complicated and not easy to be measured online. In [43], the authors have discussed that "*the results show that thermal comfort votes were highly correlated with the two environmental conditions, namely, temperature and humidity.*" So, to make the proposed

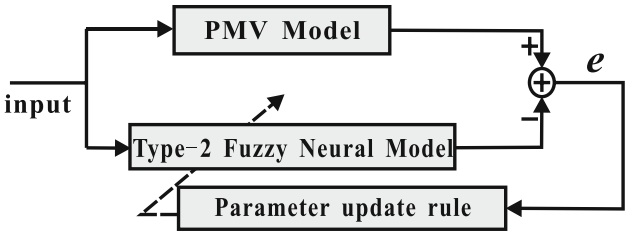

Fig. 4 Tuning schedule of the monotonic type-2 fuzzy neural prediction model

prediction model easier to apply in practice, in this study, we adopt the air temperature t_a and the relative humidity R_H as input variables of the T2FNN prediction model. And the other variables are set under reasonable assumptions in the next subsection.

On the other hand, we can see that the higher the air temperature t_a is, the larger the PMV index will be. So does the relative humidity R_H. In other words, the predicted model for the PMV index should be monotonically increasing w.r.t. the air temperature t_a and the relative humidity R_H. Hence, an appropriate T2FNN prediction model for the thermal comfort index should be monotonically increasing.

6.2 Monotonic T2FNN prediction model for the thermal comfort index

Figure 4 shows the tuning schedule of the T2FNN prediction model. To construct the T2FNN prediction model, the training data are obtained from Fanger's PMV model to reflect the relationship between the inputs and the thermal comfort index. The inputs of the training data are the values of t_a, R_H, while the outputs of the training data are the PMV index values. In the training process, the proposed hybrid algorithm is utilized to update the parameters of the prediction model.

The training data pairs are generated under the following reasonable assumptions:

1. Generally speaking, occupants usually engage in light work when they are indoor. In this case, the human metabolic rate is 69.78 W/m^2, so we suppose the activity level to be 69.78 W/m^2;
2. We set clothing thermal resistance to 0.7 (clo);
3. The radiant temperature is set to be equal to the air temperature;
4. As the specifications of the air conditioning design require the air velocity to be less than 0.25 m/s in summer, the air velocity is set to 0.20 m/s.

6.3 Simulation results and comparison

Under the assumptions above, training data pairs are generated. The sampling range of the air temperature is [10 °C, 36 °C], and the sampling step is 1 °C. The sampling range of the relative humidity is [0, 100 %], and the sampling step is 5 %. Thus, 567 training data pairs are obtained.

In the simulation, we assign each input variable 5 Gaussian T2FSs as shown in Fig. 5a, b, that is, $N_1 = N_2 = 5$ and there are $N_1{*}N_2 = 25$ fuzzy rules in this T2FNN. The initial interval weights between Layer 3 and Layer 4 are set to zero, and the initial defuzzification coefficient η is set to 0.5. In

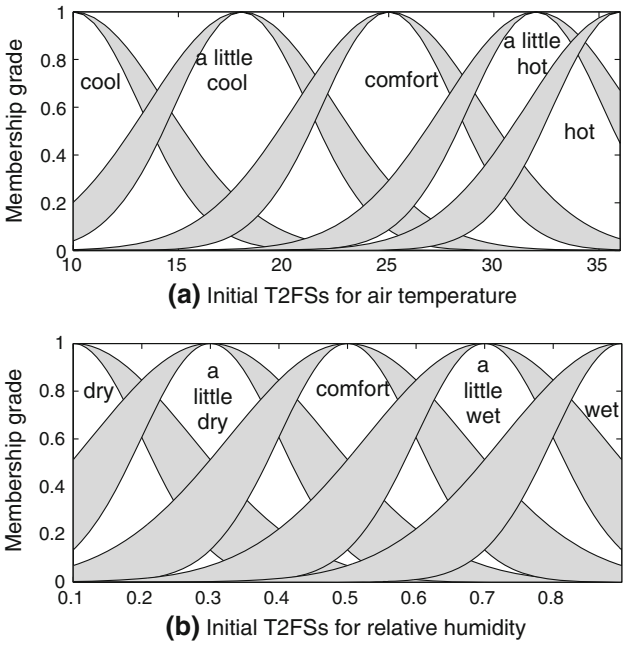

Fig. 5 Initial T2FSs in Layer 2

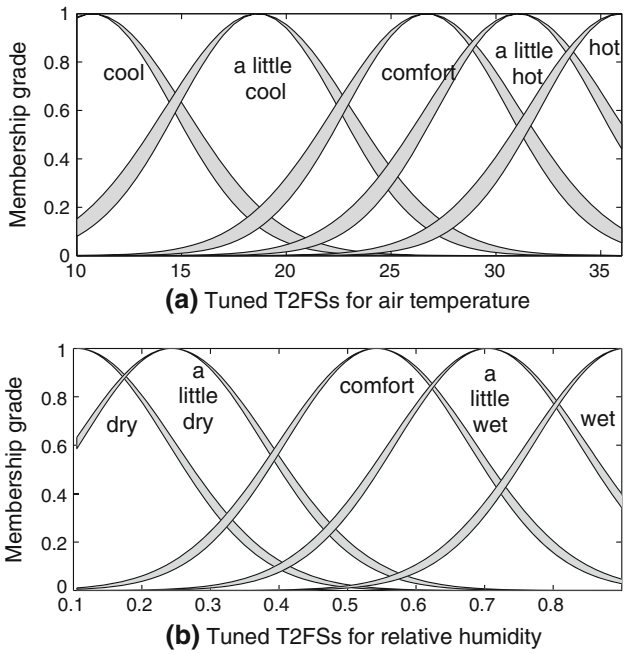

Fig. 6 Tuned T2FSs in Layer 2

Table 1 Final interval weights between Layer 3 and Layer 4

$[\underline{w}^{i_1 i_2}, \overline{w}^{i_1 i_2}]$		i_1				
		1	2	3	4	5
i_2	1	[−4.00, −4.00]	[−1.79, −1.78]	[−0.86, 0.48]	[0.55, 1.90]	[3.68, 3.71]
	2	[−4.00, −4.00]	[−1.57, −1.56]	[−0.60, 0.51]	[0.60, 2.12]	[4.35, 4.38]
	3	[−3.99, −3.99]	[−1.47, −1.40]	[−0.50, 0.80]	[0.65, 2.43]	[4.48, 4.52]
	4	[−3.94, −3.90]	[−1.45, −1.38]	[−0.47, 0.76]	[1.23, 2.46]	[5.00, 5.00]
	5	[−3.87, −3.85]	[−1.20, −1.10]	[−0.29, 1.05]	[1.90, 3.39]	[5.00, 5.00]

Note: $[\underline{w}^{i_1 i_2}, \overline{w}^{i_1 i_2}]$s are the interval weights of the 25 fuzzy rules, where $i_1 = 1, 2, ..., 5$ and $i_2 = 1, 2, ..., 5$

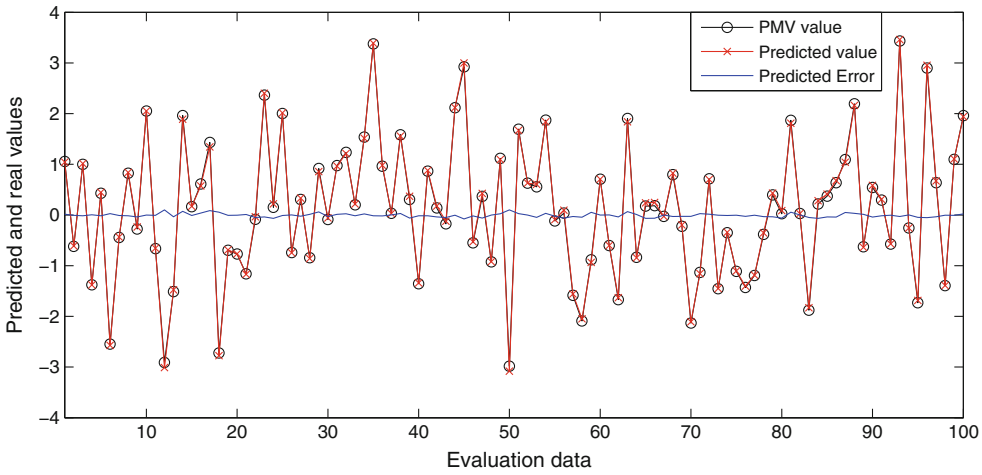

Fig. 7 Comparison between the predicted value and the PMV value

the hybrid algorithm, the penalty factor λ is chosen to be 100, and the learning rates are chosen as $\alpha_w = 0.01$, $\alpha_\eta = 0.001$, $\alpha_c = 0.5$ for the air temperature and $\alpha_c = 5e{-}6$ for the relative humidity, $\alpha_\delta = 1$ for the air temperature, and $\alpha_\delta = 5e{-}7$ for the relative humidity.

After being trained, $\eta = 0.5527$, and the T2FSs in Layer 2 are shown in Fig. 6a, b. The optimized interval weights between Layer 3 and Layer 4 are shown in Table 1. According to the conditions in Theorem 1, the optimized T2FNN is monotonic.

For comparison, a T1FNN is designed and experimented. The T1FSs in Layer 2 of the T1FNN are still Gaussian fuzzy sets with the same centers $c_j^{i_j}$ as in the T2FNN, but the widths of them are the average of $\underline{\delta}_j^2$ and $\overline{\delta}_j^2$. In the same way, the crisp weights between Layer 3 and Layer 4 of the T1FNN are set as the average of the interval weights of the T2FNN.

A linear regression model is also provided. For the above training data, the linear regression model can be obtained as: $-7.1160 + 0.2818{*}t_a + 0.6883{*}R_H$.

For the 567 pairs of training data, the root-mean-square error (RMSE) of the final T2FNN and T1FNN and the linear regression model are 0.0692, 0.0774, and 0.1067, respectively.

Table 2 RMSEs of evaluation data

Case	RMSE		
	T2FNN	T1FNN	Linear regression
1	0.0378	0.0478	0.0480
2	0.0387	0.0457	0.0484
3	0.0383	0.0516	0.0488
4	0.0386	0.0517	0.0513
5	0.0381	0.0516	0.1027
6	0.0340	0.0471	0.0943
7	0.0346	0.0462	0.0574
8	0.0354	0.0460	0.0657
9	0.0357	0.0511	0.0513
10	0.0375	0.0489	0.0821
Average	0.0369	0.0488	0.0650

One hundred data pairs are generated randomly for evaluation. And, such evaluation process is run for ten times. In one of the ten runs, comparison between the PMV value and the value obtained from the monotonic T2FNN prediction model is demonstrated in Fig. 7. The prediction errors are also shown in Fig. 7 (blue line). For the evaluation data, in each run, the RMSEs of the final T2FNN and

T1FNN and the linear regression model are listed in Table 2.

From Fig. 7, we can see that the proposed prediction model can achieve satisfactory performance and the prediction error lies in a fine scale. Hence, the proposed monotonic T2FNN prediction model can be used as a good alternative in real-time applications where the PMV index should be computed online. From Table 2, we can conclude that the monotonic T2FNN model performs better than its counterpart—T1FNN model and the linear regression model.

7 Conclusion

In lots of identification and prediction applications, specific physical structure knowledge about systems may be difficult to obtain. But, some properties (e.g., monotonicity) of the system may be obvious. Hence, it is quite important for us to utilize such properties to construct the identification or prediction models. We have in this paper addressed the topic on the monotonicity property of T2FNNs. We have presented sufficient monotonicity conditions on the parameters of T2FNN—the conditions on T2FSs in Layer 2 and the conditions on the interval weights between Layer 3 and Layer 4—to meet the monotonicity property between its input and output. Furthermore, data-driven design and optimization of the monotonic T2FNN are studied. To optimize the parameters of the monotonic T2FNN, we have provided a hybrid algorithm which is the combination of the constrained least squares method and the gradient descent algorithm. Application to the thermal comfort index prediction and comparisons with other methods have demonstrated the effectiveness and advantage of the monotonic T2FNN.

Acknowledgments This work is supported by National Natural Science Foundation of China (61105077, 61273149, 61074149 and 61273326), and the Excellent Young and Middle-Aged Scientist Award Grant of Shandong Province of China (BS2012DX026).

References

1. Wang LX (1994) Adaptive fuzzy system and control: design and stability analysis. Prentice-Hall, New Jersy
2. Jang JR, Sun CT, Mizutani E (1997) Neuro-fuzzy and soft computing. Prentice-Hall, New Jersy
3. Jang JR (1993) ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 23(3):665–684
4. Wu S, Er MJ (2000) Dynamic fuzzy neural networks—a novel approach to function approximation. IEEE Trans Syst Man Cybern B 30(2):358–364
5. Lin D, Wang X, Nian F, Zhang Y (2010) Dynamic fuzzy neural networks modeling and adaptive backstepping tracking control of uncertain chaotic systems. Neurocomputing 73(16–18):2873–2881
6. Pratama M, Er MJ, Li X, et al (2011) Genetic dynamic fuzzy neural network (GDFNN) for nonlinear system identification. Lect Notes Comput Sci 6676/2011:525–534
7. Han H, Qiao J (2010) A self-organizing fuzzy neural network based on a growing-and-pruning algorithm. IEEE Trans Fuzzy Syst 18(6):1129–1143
8. Zadeh LA (1975) The concept of a linguistic variable and its application to approximate reasoning—1. Inf Sci 8:199–249
9. Mendel JM (2001) Uncertain rule-based fuzzy logic systems: introduction and new directions. Prentice-Hall, New Jersy
10. Liang Q, Mendel JM (2000) Interval type-2 fuzzy logic systems: theory and design. IEEE Trans Fuzzy Syst 8(5):535–550
11. Juang CF, Hsu CH (2009) Reinforcement ant optimized fuzzy controller for mobile robot wall following control. IEEE Trans Ind Electron 56(10):3931–3940
12. Begian M, Melek W, Mendel JM (2008) Stability analysis of type-2 fuzzy systems. In: Proceedings of 2008 IEEE international conference on fuzzy systems, pp 947–953
13. Li C, Yi J, Wang T (2011) Encoding prior knowledge into data driven design of interval type-2 fuzzy logic systems. Int J Innov Comput Inf Control 7(3):1133–1144
14. Li C, Yi J (2010) SIRMs based interval type-2 fuzzy inference systems: properties and application. Int J Innov Comput Inf Control 6(9):4019–4028
15. Wang CH, Cheng CS, Lee TT (2004) Dynamical optimal training for interval type-2 fuzzy neural network. IEEE Trans Syst Man Cybern 34(3):1462–1477
16. Hagras H (2006) Comments on dynamical optimal training for interval type-2 fuzzy neural network (T2FNN). IEEE Trans Syst Man Cybern 36(5):1206–1209
17. Lee CH, Hong JL, Lin YC, Lai WY (2003) Type-2 fuzzy neural network systems and learning. Int J Comput Cognit 1(4):79–90
18. Juang CF, Tsao YW (2008) A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning. IEEE Trans Fuzzy Syst 16(6):1411–1424
19. Juang CF, Lin YY, Huang RB (2011) Dynamic system modeling using a recurrent interval-valued fuzzy neural network and its hardware implementation. Fuzzy Set Syst 179:83–99
20. Contreras RJ, Vellasco M, Tanscheit R (2011) Hierarchical type-2 neuro-fuzzy BSP model. Inf Sci 181:3210–3224
21. Aliev RA, Pedrycz W, Guirimov BG, et al. (2011) Type-2 fuzzy neural networks with fuzzy clustering and differential evolution optimization. Inf Sci 181:1591–1608
22. Lin FJ, Shieh PH, Hung YC (2008) An intelligent control for linear ultrasonic motor using interval type-2 fuzzy neural network. IET Electr Power Appl 2(1):32–41
23. Li C, Yi J, Zhao D (2008) Interval type-2 fuzzy neural network controller (IT2FNNC) and its application to a coupled-tank liquid-level control system. In: Proceedings of 3rd international conference on innovative computing information and control, pp 508–511
24. Li C, Yi J, Yu Y, Zhao D (2010) Inverse control of cable-driven parallel mechanism using type-2 fuzzy neural network. Acta Autom Sinica 36(3):459–464
25. Abiyev RH, Kaynak O (2010) Type 2 fuzzy neural structure for identification and control of time-varying plants. IEEE Trans Ind Electron 57(12):4147–4159
26. Tu CC, Juang CF (2012) Recurrent type-2 fuzzy neural network using haar wavelet energy and entropy features for speech detection in noisy environments. Expert Syst Appl 39:2479–2488
27. Chen CS, Lin WC (2011) Self-adaptive interval type-2 neural fuzzy network control for PMLSM drives. Expert Syst Appl 38:14679–14689
28. Abiyev RH, Kaynak O, Alshanableh T, Mamedov F (2011) A type-2 neuro-fuzzy system based on clustering and gradient techniques applied to system identification and channel equalization. Appl Soft Comput 11:1396–1406

29. Lindskog P, Ljung L (2000) Ensuring monotonic gain characteristics in estimated models by fuzzy model structures. Automatica 36:311–317
30. Won JM, Park SY, Lee JS (2002) Parameter conditions for monotonic Takagi-Sugeno-Kang fuzzy system. Fuzzy Set Syst 132:135–146
31. Wu CJ, Sung AH (1996) A general purpose fuzzy controller for monotone functions. IEEE Trans Syst Man Cybern B 26(5): 803–808
32. Wu CJ (1997) Guaranteed accurate fuzzy controllers for monotone functions. Fuzzy Set Syst 92:71–82
33. Zhao H, Zhu C (2000) Monotone fuzzy control method and its control performance. In: Proceedings of 2000 IEEE international conference on system, man, cybernetics, pp 3740–3745
34. Koo K, Won JM, Lee JS (2004) Least squares identification of monotonic fuzzy systems. In: Proceedings of annual meeting of the North American fuzzy Information Processing Society (NAFIPS), pp 745–749
35. Seki H, Ishii H, Mizumoto M (2007) On the monotonicity of single input type fuzzy reasoning methods. IEICE Trans Fundam E90-A(7):1462–1468
36. Broekhoven EV, Baets BD (2008) Monotone Mamdani–Assilian models under mean of maxima defuzzification. Fuzzy Set Syst 159(21):2819–2844
37. Li C, Zhang G, Yi J, Wang T (2011) On the properties of SIRMs connected type-1 and type-2 fuzzy inference systems. In: Proceedings of 2011 IEEE international conference on fuzzy systems, pp 1982–1988
38. Li C, Yi J, Zhao D (2009) Analysis and design of monotonic type-2 fuzzy inference systems. In: Proceedings of 2009 IEEE international conference on fuzzy systems, pp 1193–1198
39. Nelles O (2001) Nonlinear system identification. Springer, Berlin
40. Fanger PO (1970) Thermal comfort: analysis and applications in environmental engineering. McGraw-Hill, New York
41. Atthajariyakul S, Leephakpreeda T (2005) Neural computing thermal comfort index for HVAC systems. Energ Convers Manage 46:2553–2565
42. Ma B, Shu J, Wang Y (2011) Experimental design and the GA-BP prediction of human thermal comfort index. In: Proceedings of the 2011 seventh international conference on natural computation, pp 771–775
43. Chen K, Jiao Y, Lee ES (2006) Fuzzy adaptive networks in thermal comfort. Appl Math Lett 19:420–426