mathematics

MDPI

Article

First-Order Sparse TSK Nonstationary Fuzzy Neural Network Based on the Mean Shift Algorithm and the Group Lasso Regularization

Bingjie Zhang [1], Jian Wang [2,*], Xiaoling Gong [3], Zhanglei Shi [2], Chao Zhang [1,*], Kai Zhang [4,5], El-Sayed M. El-Alfy [6] and Sergey V. Ablameyko [7]

1 School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China; bingjie_zhang_1993@163.com
2 College of Science, China University of Petroleum (East China), Qingdao 266580, China; zlshi@upc.edu.cn
3 College of Control Science and Engineering, China University of Petroleum (East China), Qingdao 266580, China; gongxiaoling@s.upc.edu.cn
4 School of Petroleum Engineering, China University of Petroleum (East China), Qingdao 266580, China; zhangkai@upc.edu.cn
5 School of Science, Qingdao University of Technology, Qingdao 266580, China
6 Fellow SDAIA-KFUPM Joint Research Center for Artificial Intelligence, Interdisciplinary Research Center of Intelligent Secure Systems, Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia; alfy@kfupm.edu.sa
7 Faculty of Applied Mathematics and Computer Science, Belarusian State University, 220030 Minsk, Belarus; ablameyko@bsu.by
* Correspondence: wangjiannl@upc.edu.cn (J.W.); chao.zhang@dlut.edu.cn (C.Z.)

Abstract: Nonstationary fuzzy inference systems (NFIS) are able to tackle uncertainties and avoid the difficulty of type-reduction operation. Combining NFIS and neural network, a first-order sparse TSK nonstationary fuzzy neural network (SNFNN-1) is proposed in this paper to improve the interpretability/translatability of neural networks and the self-learning ability of fuzzy rules/sets. The whole architecture of SNFNN-1 can be considered as an integrated model of multiple sub-networks with a variation in center, variation in width or variation in noise. Thus, it is able to model both "intraexpert" and "interexpert" variability. There are two techniques adopted in this network: the Mean Shift-based fuzzy partition and the Group Lasso-based rule selection, which can adaptively generate a suitable number of clusters and select important fuzzy rules, respectively. Quantitative experiments on six UCI datasets demonstrate the effectiveness and robustness of the proposed model.

Keywords: nonstationary neuro-fuzzy network; mean shift; group lasso; rule reduction

MSC: 68T07; 94D05; 68T27; 03B52; 68T05

Citation: Zhang, B.; Wang, J.; Gong, X.; Shi, Z.; Zhang, C.; Zhang, K.; El-Alfy, E.-S.M.; Ablameyko, S.V. First-Order Sparse TSK Nonstationary Fuzzy Neural Network Based on the Mean Shift Algorithm and the Group Lasso Regularization. *Mathematics* **2024**, *12*, 120. https://doi.org/10.3390/math12010120

Academic Editor: Georgios Tsekouras

Received: 27 October 2023
Revised: 19 December 2023
Accepted: 28 December 2023
Published: 29 December 2023

1. Introduction

As a flexible, interpretable machine learning model, fuzzy neural networks (FNNs) have been widely used in various fields, such as image processing [1], fuzzy control [2,3], ranking challenges, risks and threats [4], actual classification and prediction [5–8], and so on. One of the most commonly used FNN structures is the Takagi-Sugeno-Kang (TSK) [9] fuzzy system, also called TSK neuro-fuzzy system because it can be represented as a neural network [10–12]. The most famous TSK neuro-fuzzy system is the adaptive-network-based fuzzy inference system (ANFIS) [10], which can be regarded as a first-order type-1 TSK FNN. This kind of first-order type-1 TSK FNN can bridge the gap between the linguistic and numerical representation of knowledge. Its fuzzy logic component provides a linguistic representation of knowledge, whereas the neural network component provides a numerical representation. Thus, it is a powerful tool that can address practical and theoretical gaps

in modeling complex systems. Its ability to handle uncertain and imprecise data, provide interpretable rules, and perform online learning make it useful in various applications.

The training of FNNs is a necessary work. There are many existing training algorithms, such as backpropagation [5], particle swarm algorithm [13], hybrid algorithm [14] and so on [15]. Although evolutionary algorithms and hybrid-type algorithms work well, they require considerable running time. As an efficient and commonly used algorithm, the backpropagation algorithm has become a common scheme to optimize TSK FNNs. Many convergence results of gradient-based backpropagation algorithms [16,17] in FNNs provide theoretical guarantees for a wide range of applications of this algorithm. Hence, the gradient-based backpropagation algorithm is used in this paper to train the FNN model.

For fuzzy neural networks (FNNs), the fuzzy partition of input space is an important task in structure recognition, which affects the number of generated fuzzy rules (FRs). There are two typical partitioning methods: grid-type partition [10] and clustering-based partition [17–20]. The grid-type partition partitions the input space into multiple grids. Each grid represents a FR. It is simple but leads to the generation of rules that increase dramatically with the number of dimensions. Compared to the grid-type partitioning method, the clustering-based partition reduces the number of FRs yielded. It clusters input training vectors into input space to provide more flexible partitioning. To make the meaning of each clustering center and the matching fuzzy term of each input transparent to their users, the formed clusters can be projected onto each dimension of the input space. On each dimension, one input variable corresponds to a mapping membership function (MF), and one cluster can be represented by the product of mapping MFs. The clustering-based partition commonly uses K-means or fuzzy c-means as the clustering algorithm. They need to set the number of clusters in advance, which is usually equal to the number of classes, making the generated fuzzy rule base not expressive enough. If the right number of clusters is chosen, the clustering-based partition can greatly improve the performance of FNNs. Therefore, it is an important and significant work for FNNs to adaptively generate suitable clusters and then gain a rich fuzzy rule base.

Redundancy inevitably occurs when enough FRs are yielded in a rich fuzzy rule base. The interpretability of FNNs is mainly reflected in the fuzzy rule base, which is a collection of FRs in the form of IF-THEN statements. Since too many rules will weaken the interpretability of FNNs, a reduction in the number of rules is necessary and meaningful. In order to reduce the number of FRs, various methods have been proposed, such as direct extraction techniques from numerical data [21,22], genetic algorithm-based approaches [23,24], and embedded neuro-fuzzy approaches [5,25–27]. Among these methods, the embedded neuro-fuzzy approaches simultaneously perform the rule extraction and evaluation of the model, enhancing the effectiveness and reducing the computational burden. In [5,27], the gate functions are introduced and embedded into the neuro-fuzzy models for fuzzy rule selection, which is an effective method. However, they require the introduction of the additional functions. In this paper, we attempt to explore another method without introducing an additional function to perform embedded rule selection.

The FNNs mentioned above are based upon type-1 fuzzy sets (FSs), which are precise. Thus, these type-1 FNNs struggle to tackle the uncertainty of rules. To tackle uncertainties (such as noise measurements, semantics variations, and so on), the type-2 FNNs are presented [28] and have excellent performance [29–31]. Unfortunately, due to type-reduction procedure (such as Karnik-Mendel iteration [32]) from type-2 to type-1, significant additional computational costs will incur for type-2 FNNs. Moreover, the type-2 fuzzy sets cannot capture the notion of variability, nor to model the variation (over time) in the opinions of individual experts and expert groups, called "intraexpert" and "interexpert" variability [33], respectively. To address the problems of the type-2 FSs, the notion of nonstationary FSs (called NFSs) are introduced in [33], as well as the nonstationary fuzzy inference systems (NFISs). Different from type-2 FSs, a NFS can actually be viewed as a collection of multiple type-1 fuzzy sets yielded by the perturbation MFs without secondary MFs. Therefore, the NFIS has a fundamentally different inference mechanism than the type-2

fuzzy inference system. Combining neural networks with NFIS, a zero-order nonstationary FNN (NFNN-0) is presented in [17], which can directly address the uncertainties and model the "intraexpert" and "interexpert" variability. These previous works have demonstrated the validity and strong robustness of NFS and nonstationary FNN. However, the nonlinear mapping ability of NFNN-0 is weak due to the use of zero-order, so improving its nonlinear representation ability is also a valuable work.

In this paper, based on the Mean Shift algorithm and the Group Lasso regularization, a first-order sparse TSK nonstationary fuzzy neural network is proposed. It combines the learning strategy of a neural network with the logical inference ability and language expression ability of first-order NFIS, making the neural network interpretable/translatable and realizing the self-learning of fuzzy rules/sets. It does not need to face the difficulty that type-2 fuzzy inference mechanism needs to perform type-reduction procedures, because it is actually a repetition of a first-order type-1 FNN with slightly different MFs over time. Like NFNN-0, the proposed model can also model the "intraexpert" and "interexpert" variability as well as directly deal with the uncertainties. The main contributions of this paper are summarized as follows:

(i) To improve the nonlinear representation ability of NFNN-0, we extend the nonstationary fuzzy neural network from zero-order to first-order, and propose a first-order sparse TSK nonstationary fuzzy neural network, called SNFNN-1. SNFNN-1 can significantly improve the performance of NFNN-0. Simulation experiments confirm the effectiveness and robustness of our model.

(ii) To adaptively generate a suitable number of clusters and FRs, the Mean Shift algorithm is used to partition the input space and construct the antecedent structure of our SNFNN-1 model. Compared with fuzzy partition based on K-means clustering or fuzzy c-means clustering, this fuzzy partition method does not need to set the number of clusters in advance and can provide more effective centers as well as the number of MFs. With the gained MFs, a rich fuzzy rule base is subsequently generated.

(iii) Considering the redundancy among rules of the rich fuzzy rule base, we add a regularization term, i.e., Group Lasso term, to the objective function to penalize each rule, thus producing sparsity of rules in a grouped manner. Then, in the rule-consequent structure of SNFNN-1, combined with a rule selection method, the important rules are retained and the useless or inappropriate rules are deleted, so as to achieve the purpose of rule reduction.

The rest of the research content is arranged as follows. Section 2 describes the architecture of our proposed SNFNN-1 model. In Section 3, supporting experiments are implemented. Section 4 draws the conclusions and outlines directions for future work.

2. First-Order Sparse TSK Nonstationary Fuzzy Neural Network (SNFNN-1)

In this section, we elaborate on the construction of the first-order sparse TSK nonstationary fuzzy neural network, namely SNFNN-1. There is no need to preset the number of clusters in the fuzzy partition of our model due to the use of the Mean Shift algorithm. Additionally, rule selection is achieved by adding a Group Lasso regularization to the objective function.

Assume that $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_n]^\top \in \mathbb{R}^n \times \mathbb{R}^D$ is the sample set, and $\mathbf{o} = [o_1, o_2, \ldots, o_n]^\top$ is the corresponding desired outputs, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]^\top$ is the i-th sample, D is the number of input dimensions, and n is the number of samples.

Before constructing our model, the number of clusters needs to be determined via the Mean Shift algorithm, as well as the corresponding centers and standard deviations. The following is the specific method. Giving a bandwidth value h, take an unlabeled sample as a cluster center $\hat{\mathbf{x}}^0$, and then update this center by

$$\hat{\mathbf{x}}^{m+1} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g_{pG}\left(\left\|\frac{\hat{\mathbf{x}}^m - \mathbf{x}_i}{h}\right\|\right)}{\sum_{i=1}^{n} g_{pG}\left(\left\|\frac{\hat{\mathbf{x}}^m - \mathbf{x}_i}{h}\right\|\right)}, \tag{1}$$

where $m = 0, 1, 2, \ldots$ is the m-th iteration. Mark all samples that were once within the bandwidth until it converges. Then, we can gain a clustering center $\mathbf{c}_1$. Repeat the above operation until all sample points are labeled. Eventually, we can obtain an adaptive number of clusters, S, as well as S clustering centers $\{\mathbf{c}_s\}_{s=1}^{S}$ and a clustering sample set $\mathbb{C} = \{\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_S\}$. According to [34], each standard deviation is calculated by

$$\sigma_{sd} = \sqrt{\sum_{\mathbf{x}_i \in \mathbb{C}_s} \frac{(x_{id} - c_{sd})^2}{n_{sd}}}, \tag{2}$$

where $d = 1, 2, \ldots, D$, and $s = 1, 2, \ldots, S$.

For brevity, the structure of a multi-input-single-output SNFNN-1 is constructed, as shown in Figure 1. It is easily extended to the case of multiple outputs. Actually, the proposed SNFNN-1 can be considered as an integrated network of T sub-networks, where T means the repetitions with variation in center/width or noise [33]. In this paper, the variation in center is taken as an example. Each sub-network is a first-order type-1 FNN based on the Mean Shift algorithm and the Group Lasso, abbreviated as SFNN-1.

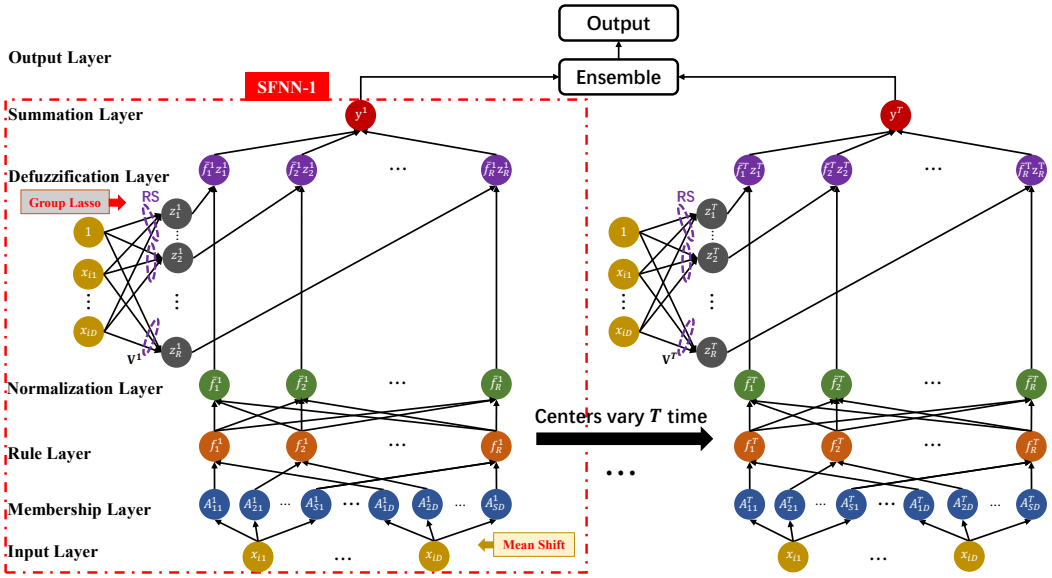

Figure 1. Structure of the SNFNN-1 model.

In the FNNs with the clustering-based fuzzy partition, the number of fuzzy rules are commonly the same with the number of clusters, that is, $R = S$. For an input vector $\mathbf{x}_i$ $(i \in 1, 2, \ldots, n)$, each first-order type-1 fuzzy rule of the t-th SFNN-1 is defined as:

$$\mathbf{R}_r^t : \textbf{IF } x_{i1} \textbf{ is } A_{r1}^t \textbf{ and } x_{i2} \textbf{ is } A_{r2}^t \textbf{ and } \ldots \textbf{ and } x_{iD} \textbf{ is } A_{rD}^t$$
$$\textbf{THEN } : z_r^t(\mathbf{x}_i) = \left(\mathbf{v}_r^t\right)^\top \tilde{\mathbf{x}}_i,$$

where r $(r = 1, 2, \ldots, R)$ means the r-th rule, and R represents the total number of rules. t $(t = 1, 2, \ldots, T)$ is the t-th sub-network (i.e., SFNN-1). A_{rd}^t is the fuzzy set associated with d-th feature in the r-th rule of t-th sub-network, whose membership function contains triangular, trapezoidal, Gaussian functions, and so on. In this paper, the widely used Gaussian functions are adopted. The clustering centers generated by the Mean Shift algorithm and the corresponding standard deviations are regarded as the centers and widths of membership functions, respectively. $\mathbf{v}_r^t = [v_{r0}^t, v_{r1}^t, \ldots, v_{rD}^t]^\top$ is the consequent parameter vector of r-th rule, and $\mathbf{V}^t = [\mathbf{v}_1^t, \mathbf{v}_2^t, \ldots, \mathbf{v}_R^t]^\top$ is all consequent parameters. $\tilde{\mathbf{x}}_i = [1; \mathbf{x}_i]$ indicates the $(1 + D) \times 1$ dimensional vector.

A simple SNFNN-1 model owns seven layers and the detailed presentation of each layer is as follows:

Layer 1 (Input layer): each node indicates an input variable (crisp variable) in this layer. Then all input variables are fed to the next layer.

Layer 2 (Membership/Fuzzification layer): in this layer, the Gaussian membership functions (GMFs) are adopted to produce the membership degrees of the input variables. In order to avoid the derivation of the denominator, reduce the amount of calculation and the difficulty of theoretical analysis, we take the reciprocals of the widths of GMFs as the independent variables, referring to [16]. Hence, each membership value with variation in center is defined as

$$\mu_{A^t_{rd}}(x_{id}) = e^{-\frac{1}{2}(x_{id}-c^t_{rd})^2(b^t_{rd})^2}, \tag{3}$$

where $t = 1, 2, \ldots, T$, $r = 1, 2, \ldots, R$, $d = 1, 2, \ldots, D$, and x_{id} is the input variable of i-th sample on the d-th dimension. $c^t_{rd} = \tilde{c}_{rd} + \kappa \sin(\omega(t-1) + \theta)$ means the center yielded by the perturbation function [33] for t-th SFNN-1, where $\tilde{c}_{rd}$ is the benchmark center. κ, ω and θ denote three hyper-parameters used to cause tiny periodic perturbations in center for simulating the variation in opinions of expert groups. $b^t_{rd} = \tilde{b}_{rd}$ is the reciprocal of width without variation, where $\tilde{b}_{rd}$ means the benchmark reciprocal of width. Figure 2 shows an instantiation of nonstationary fuzzy set (NFS) based on Gaussian functions with $t = 1:30$, $x_{id} = -10:10$, $\tilde{c}_{rd} = 0$, $\kappa = 0.8$, $\omega = 1$, $\theta = 0$ and $\tilde{\sigma}_{rd} = 2.5$.

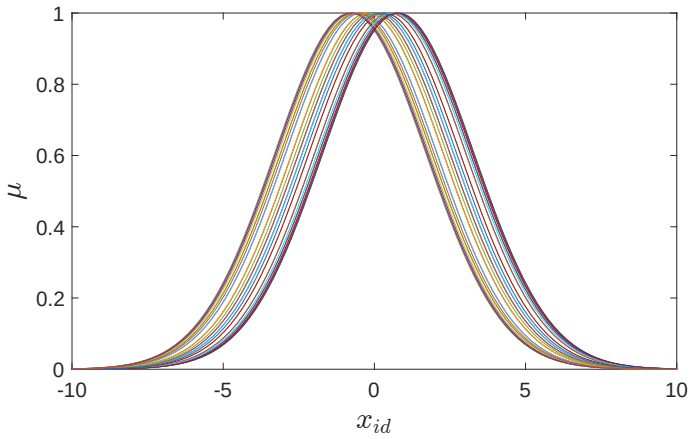

Figure 2. An instantiation of NFS based on Gaussian functions with variation in center.

Layer 3 (Rule layer): each node of this layer is a rule node indicating a term of a fuzzy rule. For the r-th rule of the t-th SFNN-1, the firing strength based on the product T-norm is calculated by:

$$f^t_r(\mathbf{x}_i) = \prod_{d=1}^{D} \mu_{A^t_{rd}}(x_{id}). \tag{4}$$

Layer 4 (Normalization layer): for each rule, the normalized firing strength is calculated in this layer. The corresponding strength of the r-th rule in the t-th SFNN-1, $\bar{f}^t_r(\mathbf{x}_i)$, is given as

$$\bar{f}^t_r(\mathbf{x}_i) = \frac{f^t_r(\mathbf{x}_i)}{\sum_{k=1}^{R} f^t_k(\mathbf{x}_i)}. \tag{5}$$

Layer 5 (Defuzzification layer): the defuzzification operation is applied in this layer. It multiplies each output conclusion in the first-order fuzzy rule with each normalized firing strength. The output of the r-th node in the t-th SFNN-1 is $\bar{f}^t_r(\mathbf{x}_i) z^t_r$, where z^t_r is called the consequent parameter of the r-th rule in the t-th SFNN-1.

Layer 6 (Summation layer): by summing the outputs of Layer 5, the actual output of each SFNN-1 is yielded in this layer:

$$y^t_i = \sum_{r=1}^{R} \bar{f}^t_r(\mathbf{x}_i) z^t_r. \tag{6}$$

Layer 7 (Output/Integration layer): through ensemble learning (such as an averaging or voting mechanism), this layer synthetically considers all outputs of T sub-networks. Due to the various centers of the GMFs over time, the SNFNN-1 is in fact a set of shifts of type-1 SFNN-1.

Remark 1. *The constructed nonstationary seven-layer network can be regarded as a special kind of ensemble network. Thus, it performs better than a simple type-1 FNN, especially for robustness, as illustrated by the experiments of Section 3.3.*

To realize the rule selection (RS), the Group Lasso regularization is added to the objective function. Compared with other Lasso regularization [35–37], it can induce row or column sparsity, thus producing sparsity of rules in a grouped manner, which provides the possibility for rule selection [38]. Therefore, the following objective function of each SFNN-1 contains two parts, that is, the mean square error (MSE) and the Group Lasso penalty term:

$$E(\mathbf{w}^t) = \frac{1}{2}\sum_{i=1}^{n}(y_i^t - o_i)^2 + \lambda \sum_{r=1}^{R}\left\|\mathbf{v}_r^t\right\|_2, \tag{7}$$

where $\mathbf{w}^t = \left[\mathbf{c}_1^t; \ldots; \mathbf{c}_R^t; \mathbf{b}_1^t; \ldots; \mathbf{b}_R^t; \mathbf{v}_1^t; \ldots; \mathbf{v}_R^t\right]$ is the weight vector containing all parameters, $\mathbf{c}_r^t = \left[c_{r1}^t, \ldots, c_{rD}^t\right]^\top$, $\mathbf{b}_r^t = \left[b_{r1}^t, \ldots, b_{rD}^t\right]^\top$, $\mathbf{v}_r^t = [v_{r0}^t, v_{r1}^t, \ldots, v_{rD}^t]^\top$, $r = 1, \ldots, R$, and λ is the hyper-parameters of the penalty term.

The gradients of Equation (7) with respect to $\mathbf{w}^t$ are calculated by

$$\frac{\partial E(\mathbf{w}^t)}{\partial \mathbf{w}^t} = \left(\frac{\partial E(\mathbf{w}^t)}{\partial c_{11}^t}, \ldots, \frac{\partial E(\mathbf{w}^t)}{\partial c_{RD}^t}, \frac{\partial E(\mathbf{w}^t)}{\partial b_{11}^t}, \ldots, \frac{\partial E(\mathbf{w}^t)}{\partial b_{RD}^t}, \frac{\partial E(\mathbf{w}^t)}{\partial v_{10}^t}, \ldots, \frac{\partial E(\mathbf{w}^t)}{\partial v_{RD}^t}\right)^T, \tag{8}$$

where

$$\frac{\partial E(\mathbf{w}^t)}{\partial c_{rd}^t} = \sum_{i=1}^{n}(y_i^t - o_i)\left(z_r^t - y_i^t\right)\bar{f}_r^t(\mathbf{x}_i)\left(x_{id} - c_{rd}^t\right)\left(b_{rd}^t\right)^2, \tag{9}$$

$$\frac{\partial E(\mathbf{w}^t)}{\partial b_{rd}^t} = -\sum_{i=1}^{n}(y_i^t - o_i)\left(z_r^t - y_i^t\right)\bar{f}_r^t(\mathbf{x}_i)\left(x_{id} - c_{rd}^t\right)^2\left(b_{rd}^t\right), \tag{10}$$

$$\frac{\partial E(\mathbf{w}^t)}{\partial v_{r0}^t} = \sum_{i=1}^{n}(y_i^t - o_i)\bar{f}_r^t(\mathbf{x}_i) + \lambda\frac{v_{r0}^t}{\left\|\mathbf{v}_r^t\right\|_2}, \tag{11}$$

$$\frac{\partial E(\mathbf{w}^t)}{\partial v_{rd}^t} = \sum_{i=1}^{n}(y_i^t - o_i)\bar{f}_r^t(\mathbf{x}_i)x_{id} + \lambda\frac{v_{rd}^t}{\left\|\mathbf{v}_r^t\right\|_2}, \tag{12}$$

and $r = 1, \ldots, R$, $d = 1, \ldots, D$.

For an initial weight vector $\mathbf{w}^{t,(0)}$, the updating formula of SNFNN-1 based on the typical gradient descent method is as follows:

$$\mathbf{w}^{t,(m+1)} = \mathbf{w}^{t,(m)} - \eta\frac{\partial E(\mathbf{w}^{t,(m)})}{\partial \mathbf{w}^{t,(m)}}, \tag{13}$$

where $m = 0, 1, 2, \ldots$ is the m-th iteration, and $\eta > 0$ is the learning rate.

To select the appropriate rules, we introduce a threshold ζ. The rules are selected in the following way:

$$\begin{aligned} Cond.RS: \quad & If\ \left(||\mathbf{v}_r^t||_2 \geq \tau\right), \\ & then\ (\text{the } r\text{-th rule is retained}). \end{aligned} \tag{14}$$

where

$$\tau = \min\{||\mathbf{v}_r^t||_2\} + \zeta\left(\max\{||\mathbf{v}_r^t||_2\} - \min\{||\mathbf{v}_r^t||_2\}\right). \tag{15}$$

To concisely construct the SNFNN model, we train a SFNN network in advance according to the above update method and rule selection method. Use this trained network as the

baseline network to generate T sub-networks according to the perturbation function, taking the variation in center as an example in this paper. Then, fine-tune the consequent parameters of each sub-network, while the centers and the reciprocals of widths are not retrained. Finally, we gain the outputs of all sub-networks, and then comprehensively consider all these outputs to yield the final output. Algorithm 1 summarizes the training procedure of SNFNN-1. Note that the fine-tuning process of Step 12 can be parallelized to save time.

Algorithm 1 Training procedure of SNFNN-1

Input: The training sample set $\mathbf{X}$ and its labels $\mathbf{o}$, the bandwidth h of Mean Shift algorithm, the initial consequence parameters $\mathbf{V}^{(0)}$, the learning rate η, the penalty parameter λ, the maximum iterations M and the stop threshold Θ for the model to converge, the hyper-parameter ζ for rule selection, the hyper-parameters of periodic perturbation function T, κ, ω and θ.

Output: The SNFNN-1 model and final results

1: Adopt the Mean Shift algorithm to partition the input space and generate S cluster centers as the initial centers of membership functions, $\{\mathbf{c}_r^{(0)}\}_{r=1}^{R}$, where $R = S$.
2: Calculate all widths $\{\sigma_r\}_{r=1}^{R}$ via Equation (2), where $\sigma_r = [\sigma_{r1}, \ldots, \sigma_{rD}]^\top$. Then gain the initial reciprocals of widths $\{\mathbf{b}_r^{(0)}\}_{r=1}^{R}$, where $\mathbf{b}_r^{(0)} = \sigma_r^{(0)}$
3: Define the initial weight vector $\mathbf{w}^{(0)}$, including all initial centers $\{\mathbf{c}_r^{(0)}\}_{r=1}^{R}$, reciprocals of widths $\{\mathbf{b}_r^{(0)}\}_{r=1}^{R}$ and consequent parameters $\mathbf{V}^{(0)}$.
4: Calculate the outputs of each layer of a SFNN-1 sub-network in turn by using Equations (3)–(6). Here, we can think of $t = 1$.
5: According to Equation (7), calculate the objective function of this SFNN-1.
6: Train this model based on the gradient descent method (13) until it converges.
7: By (15), calculate the thresholds τ. Use the *Cond.RS* (14) to select the important rules.
8: Retrain the SFNN-1 by using the selected rules until it converges.
9: Use this retrained SFNN-1 as the benchmark sub-network of SNFNN-1. The retrained centers and reciprocals of widths of this base model are regarded as the benchmark centers $\{\tilde{c}_{s_d d}\}_{r=1:R}^{d=1:D}$ and benchmark reciprocal of widths $\{\tilde{b}_{rd}\}_{r=1:R}^{d=1:D}$, respectively. The retrained consequent parameters is denoted by $\mathbf{V}$.
10: By Equation (3), we get T nonstationary MFs of SNFNN-1. Here, the perturbation function adopts the variation in center [33] to generate T various centers and T identical reciprocals of widths, separately denoted as $\{c_{rd}^t\}_{r=1:R}^{d=1:D}$ and $\{b_{rd}^t\}_{r=1:R}^{d=1:D}$, where $t = 1, 2, \ldots, T$.
11: Let $\mathbf{V}^t = \mathbf{V}$, where $t = 1, 2, \ldots, T$. Then, according to the T nonstationary MFs and $\mathbf{V}^t$, construct the whole SNFNN-1 model, which owns T SFNN-1 sub-networks. For various sub-networks of SNFNN-1, calculate their outputs of each layer based on the various centers, the identical reciprocals of widths and the identical consequent parameters.
12: For each SFNN-1 sub-network, only fine-tune the consequent parameters $\mathbf{V}^t$ by using the gradient descent method, while the centers and widths are no longer trained.
13: Via the ensemble learning, generate the final result of SNFNN-1 after comprehensively considering all outputs of T sub-networks.
14: Return the whole SNFNN-1 model and its final results.

3. Comparative Results and Analysis

To illustrate the superiority of SNFNN-1, we conduct experiments on six commonly used UCI datasets. For fairness, the first-order MGNF [16] and the NFNN-FCMnet-GD [17] serve as the comparison algorithms. The NFNN-FCMnet-GD is a zero-order nonstationary FNN (NFNN-0) with fuzzy c-means network-based fuzzy partition and typical gradient descent method. The experiments are conducted in PyTorch 3.9.13 framework on an Intel i5, 2.80 GHz CPU with 8.00 GB RAM.

3.1. Datasets and Experimental Settings

The UCI datasets used in this paper are downloaded from [39]. Their specific information is detailed in Table 1, including the number of samples, dimensions and classes. Before training, all sample inputs are normalized by z-score normalization [40] with a standard deviation of 1 and a mean of 0.

Table 1. Description of the datasets.

Datasets	Samples	Dimensions	Classes
Iris	150	4	3
Balance	625	4	3
Liver	345	6	2
Vertebral	310	6	3
Glass	214	9	6
Vehicle	846	18	4

In the experiments, we employ 10-fold cross-validation [5,38] to reduce the impact of initial settings when dividing the training and testing sets. Here, the specific approach of 10-fold cross-validation is to divide the data set into 10 parts as evenly as possible, select one part in turn as the testing set, and the other nine parts as the training set, thus performing 10 training and testing runs. Finally, the results of 10 times are averaged to eliminate the adverse effects caused by the unbalanced data division in a single division and avoid over-fitting. The consequent parameters $\mathbf{V}^{(0)}$ are initialized to 10^{-8}. Through grid search, the suitable bandwidth value h of the Mean Shift algorithm and penalty parameter λ of the objective function are set for each dataset. The setting of learning rate η references [41] to make the curves of objective function non-oscillatory. The maximum iterations M is set to be 1000 and the stop threshold Θ is 10^{-20} for the convergence. The hyper-parameter ζ of rule selection is set to be 0.1. The hyper-parameters of periodic perturbation function T, κ, ω and θ are set to be 30, 0.1, 0.3 and 0, respectively.

3.2. Competitive Performance of SFNN-1 and SNFNN-1

This section evaluates the performance results of SFNN-1 as well as SNFNN-1 on the above six datasets.

To show the effect of Group Lasso in Equation (7), for different datasets, the variation trend of L_2-norm of each rule is drawn in Figure 3. For a clear view, we only draw the experimental results of a single training run on each dataset. It can be seen from Figure 3 that the Group Lasso regularization is able to effectively separate important and unimportant fuzzy rules. Then, we can select the useful fuzzy rules combining the rule selection (RS) method (14) to reconstruct a concise SFNN-1 model and fine-tune it.

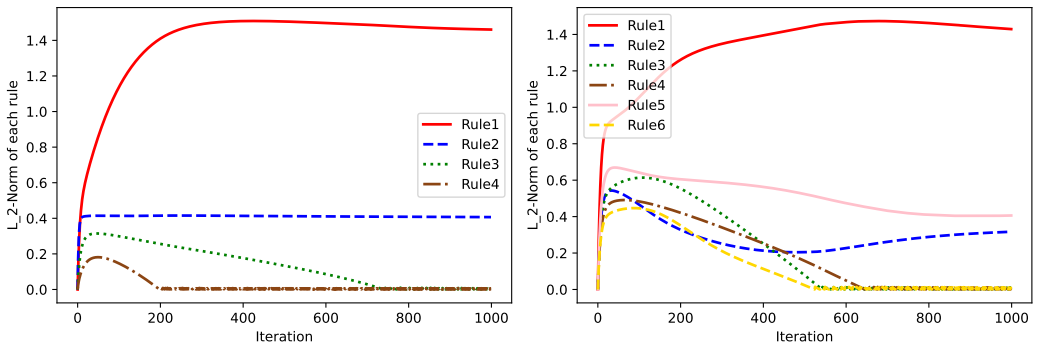

Figure 3. *Cont.*

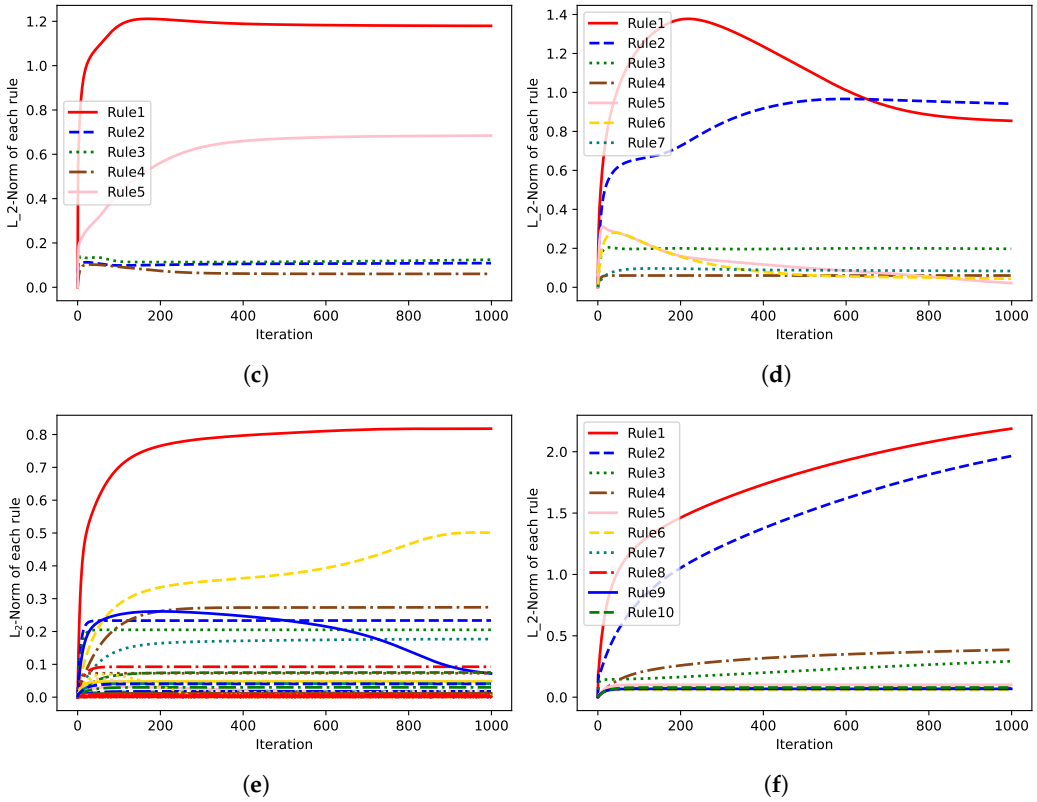

Figure 3. L_2-norm curves of fuzzy rules of a single training run for SFNN-1 on different datasets (each curve represents a fuzzy rule): (**a**) Iris. (**b**) Balance. (**c**) Liver. (**d**) Vertebral. (**e**) Glass (there are too many legends to display). (**f**) Vehicle.

To illustrate the effectiveness of RS, we present the comparison results between SFNN-1 without RS and SFNN-1, as shown in Table 2. These results are the average of 10-fold cross-validation. In Table 2, the average number of fuzzy rules for each SFNN-1 sub-network, the average training accuracy and the average testing accuracy (abbreviated to $\bar{R}$, TrAcc and TeAcc, respectively) are computed as the evaluation metrics. Specifically, $\bar{R}$ represents the average number of fuzzy rules obtained by running the model 10 times. TrAcc and TeAcc separately represent the average training accuracy and testing accuracy of 10 runs, where the training or testing accuracy for a single run is the fraction of training or test sample points that are correctly classified (the model output is consistent with the true class label) in the total training or test sample points. The standard deviation is provided within parentheses.

Table 2. Performance results of SFNN-1.

Datasets	SFNN-1 without RS *			SFNN-1 *		
	$\bar{R}$	TrAcc	TeAcc	$\bar{R}$	TrAcc	TeAcc
Iris	3.9	0.9733 (±0.0052)	0.9733 (±0.0466)	2.0	**0.9844 (±0.0042)**	**0.9800 (±0.0322)**
Balance	6.0	0.9102 (±0.0094)	0.9041 (±0.0550)	3.3	**0.9346 (±0.0205)**	**0.9279 (±0.0455)**
Liver	4.8	**0.7713 (±0.0106)**	0.7105 (±0.0803)	2.0	0.7488 (±0.0144)	**0.7250 (±0.0493)**
Vertebral	6.9	0.8789 (±0.0089)	0.8613 (±0.0664)	2.9	**0.8860 (±0.0109)**	**0.8645 (±0.0677)**
Glass	27.1	0.7663 (±0.0196)	0.6714 (±0.1341)	5.9	**0.7892 (±0.0265)**	**0.6950 (±0.1154)**
Vehicle	10.4	0.8345 (±0.0053)	0.8003 (±0.0266)	3.9	**0.8399 (±0.0047)**	**0.8108 (±0.0302)**

* Bold indicates the better one among the comparison results of the two algorithms.

From Table 2, we can observe that the performance of SFNN-1 yields higher TrAcc and TeAcc with a smaller number of fuzzy rules compared to the SFNN-1 without RS on all datasets. This reveals the effectiveness of our rule selection.

Moreover, taking the Iris dataset as an example, we give the detailed information about consequent parameters of the proposed SFNN-1 after a single training run. Here, we can think of $t = 1$. The number of rules after adopting the RS method is two. Due to the multi-outputs (i.e., three label classes) of Iris, the consequent parameters can be expressed as a three-dimensional tensor as follows:

$$\begin{aligned}(\mathbf{V}^1)^\top = [&[[-0.0193, 0.0007, 0.0062, 0.0206, 0.0007],\\ &[1.2925, 0.1781, 0.1137, -0.7817, -0.4882],\\ &[-0.2616, -0.1785, -0.1226, 0.7487, 0.4868]],\\ &[[0.5076, -0.0070, 0.0185, -0.2632, -0.1031],\\ &[-0.0377, -0.0075, -0.0002, -0.0030, -0.0099],\\ &[0.2319, 0.0127, -0.0131, 0.0945, 0.0592]]].\end{aligned} \tag{16}$$

The consequent output is expressed by a matrix $\mathbf{Z}^1$ and its all elements are given as follows:

$$\begin{aligned} z^1_{11}(\mathbf{x}_i) &= -0.0193 + 0.0007x_{i1} + 0.0062x_{i2} + 0.0206x_{i3} + 0.0007x_{i4},\\ z^1_{12}(\mathbf{x}_i) &= 1.2925 + 0.1781x_{i1} + 0.1137x_{i2} - 0.7817x_{i3} - 0.4882x_{i4},\\ z^1_{13}(\mathbf{x}_i) &= -0.2616 - 0.1785x_{i1} - 0.1226x_{i2} + 0.7487x_{i3} + 0.4868x_{i4},\\ z^1_{21}(\mathbf{x}_i) &= 0.5076 - 0.0070x_{i1} + 0.0185x_{i2} - 0.2632x_{i3} - 0.1031x_{i4},\\ z^1_{22}(\mathbf{x}_i) &= -0.0377 - 0.0075x_{i1} - 0.0002x_{i2} - 0.0030x_{i3} - 0.0099x_{i4},\\ z^1_{23}(\mathbf{x}_i) &= 0.2319 + 0.0127x_{i1} - 0.0131x_{i2} + 0.0945x_{i3} + 0.0592x_{i4}. \end{aligned} \tag{17}$$

Given a test sample $x_j = [-1.7489, -0.3564, -1.3413, -1.3130]^\top$ whose label class is 1, the normalized firing strength calculated from the antecedent of SFNN-1 is that $\bar{\mathbf{f}}^1(\mathbf{x}_j) = [\bar{f}^1_1(\mathbf{x}_j); \bar{f}^1_2(\mathbf{x}_j)] = [0.0033; 0.6529]$. And we can gain that $\mathbf{Z}^1 = [-0.0513, 2.6301, -1.5492; 1.0016, -0.0076, 0.0099]$. Then, according to Equation (6), we can calculate the final output of SFNN-1 as follows:

$$\begin{aligned}\mathbf{y}^1_j &= [\sum_{r=1}^{2} \bar{f}^1_r(\mathbf{x}_j) z^1_{r1}; \sum_{r=1}^{2} \bar{f}^1_r(\mathbf{x}_j) z^1_{r2}; \sum_{r=1}^{2} \bar{f}^1_r(\mathbf{x}_j) z^1_{r3}]\\ &= [0.9962; 0.0059; 0.0019].\end{aligned} \tag{18}$$

The index of the occurrence of the largest element in $\mathbf{y}^1_j$ is taken as the output category of the model with respect to the test sample x_j. Thus, the category is 1, which is consistent with the actual label class. These analyses illustrate the validity of the consequent parameters after rule selection.

To show the competitive performance of SNFNN-1, we compare the results of SFNN-1 and SNFNN-1 on different datasets, as shown in Table 3. Compared with SFNN-1, we can see that SNFNN-1 has the same or better performance (i.e., same or higher TrAcc and TeAcc) on all datasets.

Table 3. Performance results of SNFNN-1.

Datasets	SFNN-1 *		SNFNN-1 *	
	TrAcc	TeAcc	TrAcc	TeAcc
Iris	0.9844 (±0.0042)	0.9800 (±0.0322)	**0.9844 (±0.0042)**	**0.9800 (±0.0322)**
Balance	0.9346 (±0.0205)	0.9279 (±0.0455)	**0.9467 (±0.0241)**	**0.9391 (±0.0471)**
Liver	0.7488 (±0.0144)	0.7250 (±0.0493)	**0.7491 (±0.0148)**	**0.7250 (±0.0493)**
Vertebral	0.8860 (±0.0109)	0.8645 (±0.0677)	**0.8867 (±0.0116)**	**0.8677 (±0.0704)**
Glass	0.7892 (±0.0265)	0.6950 (±0.1154)	**0.7944 (±0.0272)**	**0.6950 (±0.1238)**
Vehicle	0.8399 (±0.0047)	0.8108 (±0.0302)	**0.8427 (±0.0074)**	**0.8155 (±0.0310)**

* Bold indicates the better one among the comparison results of the two algorithms.

3.3. Robustness of SNFNN-1

To further indicate the superiority of SNFNN-1, we add the Gaussian noise to the centers, reciprocals of widths and consequence parameters, abbreviated as C-noise, W-noise and Co-noise, respectively. Let N-noise denote the case without noise.

Figure 4 shows the robustness comparison between SFNN-1 and SNFNN-1. For obvious contrast, the means of the used Gaussian noises are all set to be 0, but the standard deviations are set differently for various datasets, whose values are described in the caption of Figure 4. From Figure 4, we can see that our SNFNN-1 owns a more stable performance than SFNN-1 on all datasets, which reveals the strong robustness of SNFNN-1.

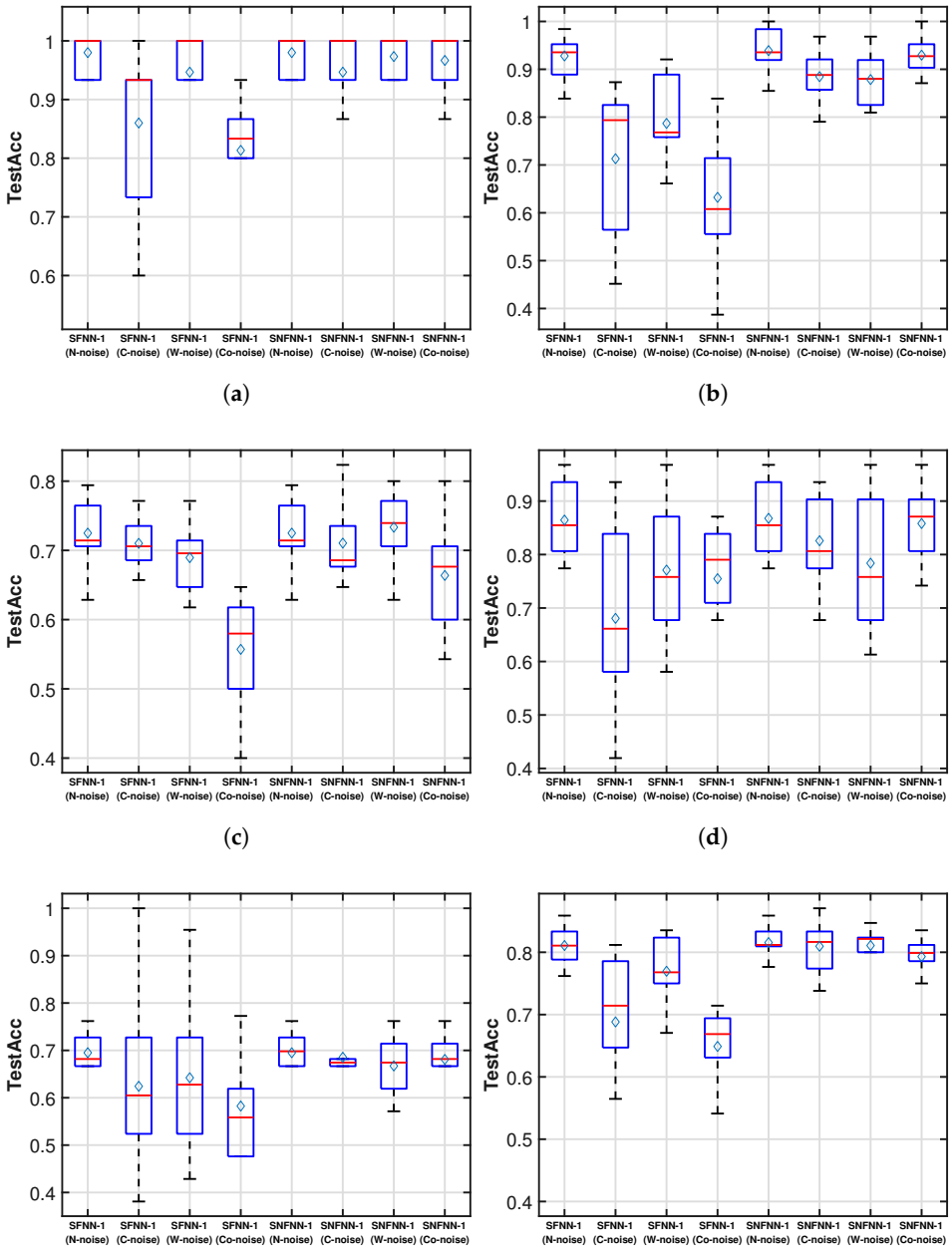

Figure 4. Robustness comparison (box plot) between SFNN-1 and SNFNN-1 on different datasets (in parentheses is the standard deviation of C-noise, W-noise and Co-noise): (**a**) Iris (1.2; 1.2; 0.3). (**b**) Balance (0.6; 0.6; 0.4). (**c**) Liver (0.8; 0.8; 0.4). (**d**) Vertebral (0.8; 0.8; 0.2). (**e**) Glass (0.6; 0.6; 0.1). (**f**) Vehicle (1.2; 1.2; 0.1).

3.4. Comparison with Other Algorithms

In this subsection, we compare our proposed models (SFNN-1 and SNFNN-1) with two other related advanced algorithms (first-order MGNF [16] and NFNN-FCMnet-GD [17]). The respective comparison results of single networks (first-order MGNF and SFNN-1) and nonstationary networks (NFNN-FCMnet-GD and SNFNN-1) are shown in Tables 4 and 5. Note that the results of NFNN-FCMnet-GD in Table 5 are directly taken from [17].

Table 4. Comparison between first-order MGNF and SFNN-1.

Datasets	First-Order MGNF [16] *		SFNN-1 *	
	$\bar{R}$	TeAcc	$\bar{R}$	TeAcc
Iris	3.0	0.9600 (±0.0466)	2.0	**0.9800 (±0.0322)**
Balance	3.0	0.9230 (±0.0365)	3.3	**0.9279 (±0.0455)**
Liver	2.0	0.7015 (±0.0681)	2.0	**0.7250 (±0.0493)**
Vertebral	3.0	0.8613 (±0.0681)	2.9	**0.8645 (±0.0677)**
Glass	6.0	0.6916 (±0.0837)	5.9	**0.6950 (±0.1154)**
Vehicle	4.0	0.8049 (±0.0359)	3.9	**0.8108 (±0.0302)**

* Bold indicates the better one among the comparison results of the two algorithms.

Table 5. Comparison between NFNN-FCMnet-GD and SNFNN-1.

Datasets	NFNN-FCMnet-GD [17] *		SNFNN-1 *	
	$\bar{R}$	TeAcc	$\bar{R}$	TeAcc
ine Iris	4.0	0.9703 (±0.0237)	2.0	**0.9800 (±0.0322)**
Balance	4.0	0.9231 (±0.0171)	3.3	**0.9391 (±0.0471)**
Liver	3.0	0.7128 (±0.0384)	2.0	**0.7250 (±0.0493)**
Vertebral	4.0	0.8377 (±0.0502)	2.9	**0.8677 (±0.0704)**
Glass	7.0	0.6774 (±0.0662)	5.9	**0.6950 (±0.1238)**
Vehicle	5.0	0.7171 (±0.0376)	3.9	**0.8155 (±0.0310)**

* Bold indicates the better one among the comparison results of the two algorithms.

From Table 4, we can observe that our SFNN-1 shows higher TeAcc with similar or fewer fuzzy rules than its comparison on all datasets, which illustrates the superiority of SFNN-1. From Table 5, it can be seen that our SNFNN-1 wins NFNN-FCMnet-GD in terms of both $\bar{R}$ and TeAcc on all datasets. These results also illustrate the effectiveness of using Group Lasso for rule selection from the side.

4. Conclusions and Future Work

In this paper, we propose a first-order sparse TSK nonstationary fuzzy neural network, abbreviated as SNFNN-1, based on the Mean Shift-based fuzzy partition and the Group Lasso regularization. It makes the neural network more interpretable/translatable and improves the self-learning of fuzzy rules/sets. Compared with the zero-order nonstationary fuzzy neural network, SNFNN-1 has stronger nonlinear representation ability. In addition, to optimize the structure of SNFNN-1, it employs two techniques: the Mean Shift-based fuzzy partition and the rule selection method based on Group Lasso. The used Mean Shift-based fuzzy partition is able to adaptively yield a appropriate number of clusters and a rich fuzzy rule base, while the proposed rule selection method based on Group Lasso can remove redundant or useless rules and simplify the network structure. The whole architecture of SNFNN-1 can be seen as an integrated model of multiple sub-networks with variation in center, width or noise. Each sub-network is a first-order sparse TSK fuzzy neural network. Our SNFNN-1 generates first-order nonstationary fuzzy sets, which is able to deal with uncertainties and does not face the difficulties of type-reduction operation from type-2 to type-1 as the type-2 fuzzy inference mechanisms does. Moreover, the proposed SNFNN-1 can model the "intraexpert" variability and "interexpert" variability. Quantitative simulations support the efficiency of the proposed models.

In the future work, it is necessary to further discuss the differences and connections among nonstationary fuzzy sets, type-1 fuzzy sets and type-2 fuzzy sets, and to really use them to

solve practical industrial problems. In addition, it is well-known that solving high-dimensional problems is difficult for fuzzy neural networks due to the computational overflow caused by the product T-norm operator. Therefore, determining how to improve the proposed model and apply it to solve high-dimensional problems is also an important future work.

Author Contributions: Writing—original draft preparation, B.Z.; writing—review and editing, B.Z. and J.W.; methodology, B.Z.; software, B.Z.; resources, J.W., Z.S. and C.Z.; validation, X.G., Z.S. and C.Z.; supervision, K.Z., S.V.A. and E.-S.M.E.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Natural Science Foundation of China under Grant 62173345; and in part by the Fundamental Research Funds for the Central Universities under Grant 22CX03002A; and in part by the China-CEEC Higher Education Institutions Consortium Program under Grant 2022151; and in part by the Introduction Plan for High Talent Foreign Experts under Grant G2023152012L; and in part by the "The Belt and Road" Innovative Talents Exchange Foreign Experts Project under Grant DL2023152001L; and in part by the National Natural Science Foundation of China under Grant 62176040 and Grant 62306337; and in part by SDAIA-KFUPM Joint Research Center for Artificial Intelligence Fellowship Program under Grant JRC-AI-RFP-04.

Data Availability Statement: The data used in this paper are not publicly available due limitations of consent for the original study but could be obtained from Zhang upon the reasonable request.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Zhang, X.; Han, Y.; Lin, S.; Xu, C. A Fuzzy Plug-and-Play Neural Network-Based Convex Shape Image Segmentation Method. *Mathematics* **2023**, *11*, 1101. [CrossRef]
2. Syed Ali, M.; Stamov, G.; Stamova, I.; Ibrahim, T.F.; Dawood, A.A.; Osman Birkea, F.M. Global Asymptotic Stability and Synchronization of Fractional-Order Reaction-Diffusion Fuzzy BAM Neural Networks with Distributed Delays via Hybrid Feedback Controllers. *Mathematics* **2023**, *11*, 4248. [CrossRef]
3. Abudusaimaiti, M.; Abudukeremu, A.; Sabir, A. Fixed/Preassigned-Time Stochastic Synchronization of Complex-Valued Fuzzy Neural Networks with Time Delay. *Mathematics* **2023**, *11*, 3769. [CrossRef]
4. Božanić, D.; Tešić, D.; Puška, A.; Štilić, A.; Muhsen, Y. Ranking challenges, risks and threats using Fuzzy Inference System. *Decis. Mak. Appl. Manag. Eng.* **2023**, *6*, 933–947. [CrossRef]
5. Xue, G.; Chang, Q.; Wang, J.; Zhang, K.; Pal, N.R. An Adaptive Neuro-Fuzzy System With Integrated Feature Selection and Rule Extraction for High-Dimensional Classification Problems. *IEEE Trans. Fuzzy Syst.* **2023**, *31*, 2167–2181. [CrossRef]
6. Pamucar, D.; Božanić, D.; Puška, A.; Marinkovic, D. Application of neuro-fuzzy system for predicting the success of a company in public procurement. *Decis. Mak. Appl. Manag. Eng.* **2022**, *5*, 135–153. [CrossRef]
7. Ljepava, N.; Jovanović, A.; Aleksić, A. Industrial Application of the ANFIS Algorithm-Customer Satisfaction Assessment in the Dairy Industry. *Mathematics* **2023**, *11*, 4221. [CrossRef]
8. Zhang, L.; Zhong, M.; Han, H. Detection of sludge bulking using adaptive fuzzy neural network and mechanism model. *Neurocomputing* **2022**, *481*, 193–201. [CrossRef]
9. Takagi, T.; Sugeno, M. Fuzzy Identification of Systems and Its Applications to Modeling and Control. In *Readings in Fuzzy Sets for Intelligent Systems*; Dubois, D., Prade, H., Yager, R.R., Eds.; Morgan Kaufmann: Burlington, MA, USA, 1993; pp. 387–403. [CrossRef]
10. Jang, J.S. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Trans. Syst. Man Cybern.* **1993**, *23*, 665–685. [CrossRef]
11. Wu, D.; Lin, C.T.; Huang, J.; Zeng, Z. On the Functional Equivalence of TSK Fuzzy Systems to Neural Networks, Mixture of Experts, CART, and Stacking Ensemble Regression. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 2570–2580. [CrossRef]
12. Beke, A.; Kumbasar, T. More Than Accuracy: A Composite Learning Framework for Interval Type-2 Fuzzy Logic Systems. *IEEE Trans. Fuzzy Syst.* **2023**, *31*, 734–744. [CrossRef]
13. Kowalski, P.A.; Sloczynski, T. A New Particle Swarm Optimisation Based Memetic Procedure for Fuzzy J-K Flop Neural Networks Learning. In Proceedings of the 2023 IEEE International Conference on Fuzzy Systems (FUZZ), Incheon, Republic of Korea, 13–17 August 2023; pp. 1–6. [CrossRef]
14. Abdo, M.I.; Abd Al Majeed, N.; Mahmoud, T.A.; Elsheikh, E.A. A Hybrid Fractional Order Fuzzy PID Controller Using Interval Type- 2 Fuzzy Neural Network. In Proceedings of the 2023 3rd International Conference on Electronic Engineering (ICEEM), Menouf, Egypt, 7–8 October 2023; pp. 1–8. [CrossRef]
15. Shi, Z.L.; Li, X.P.; Leung, C.S.; So, H.C. Cardinality Constrained Portfolio Optimization via Alternating Direction Method of Multipliers. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–9. [CrossRef] [PubMed]

16. Wu, W.; Li, L.; Yang, J.; Liu, Y. A modified gradient-based neuro-fuzzy learning algorithm and its convergence. *Inf. Sci.* **2010**, *180*, 1630–1642. [CrossRef]
17. Zhang, B.; Gong, X.; Wang, J.; Tang, F.; Zhang, K.; Wu, W. Nonstationary fuzzy neural network based on FCMnet clustering and a modified CG method with Armijo-type rule. *Inf. Sci.* **2022**, *608*, 313–338. [CrossRef]
18. Yager, R.R.; Filev, D.P. Generation of Fuzzy Rules by Mountain Clustering. *J. Intell. Fuzzy Syst.* **1994**, *2*, 209–219. [CrossRef]
19. Delgado, M.; Gomez-Skarmeta, A.; Martin, F. A fuzzy clustering-based rapid prototyping for fuzzy rule-based modeling. *IEEE Trans. Fuzzy Syst.* **1997**, *5*, 223–233. [CrossRef]
20. Juang, C.F.; Lin, C.T. An online self-constructing neural fuzzy inference network and its applications. *IEEE Trans. Fuzzy Syst.* **1998**, *6*, 12–32. [CrossRef]
21. Abe, S.; Lan, M.S. A classifier using fuzzy rules extracted directly from numerical data. In Proceedings of the Second IEEE International Conference on Fuzzy Systems, San Francisco, CA, USA, 28 March–1 April 1993; Volume 2, pp. 1191–1198. [CrossRef]
22. Abe, S.; Lan, M.S. A method for fuzzy rules extraction directly from numerical data and its application to pattern classification. *IEEE Trans. Fuzzy Syst.* **1995**, *3*, 18–28. [CrossRef]
23. Ishibuchi, H.; Yamamoto, T. Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy Sets Syst.* **2004**, *141*, 59–88. [CrossRef]
24. Ishibuchi, H.; Nojima, Y.; Kuwajima, I. Fuzzy Data Mining by Heuristic Rule Extraction and Multiobjective Genetic Rule Selection. In Proceedings of the 2006 IEEE International Conference on Fuzzy Systems, Vancouver, BC, Canada, 16–21 July 2006; pp. 1633–1640. [CrossRef]
25. Chen, Y.C.; Pal, N.R.; Chung, I.F. An Integrated Mechanism for Feature Selection and Fuzzy Rule Extraction for Classification. *IEEE Trans. Fuzzy Syst.* **2012**, *20*, 683–698. [CrossRef]
26. Chakraborty, D.; Pal, N. A neuro-fuzzy scheme for simultaneous feature selection and fuzzy rule-based classification. *IEEE Trans. Neural Netw.* **2004**, *15*, 110–123. [CrossRef] [PubMed]
27. Chung, I.F.; Chen, Y.C.; Pal, N.R. Feature Selection With Controlled Redundancy in a Fuzzy Rule Based Framework. *IEEE Trans. Fuzzy Syst.* **2018**, *26*, 734–748. [CrossRef]
28. Zadeh, L. The concept of a linguistic variable and its application to approximate reasoning—I. *Inf. Sci.* **1975**, *8*, 199–249. [CrossRef]
29. Castro, J.R.; Castillo, O.; Melin, P.; Rodríguez-Díaz, A. A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks. *Inf. Sci.* **2009**, *179*, 2175–2193. [CrossRef]
30. Ontiveros-Robles, E.; Melin, P. Toward a development of general type-2 fuzzy classifiers applied in diagnosis problems through embedded type-1 fuzzy classifiers. *Soft Comput.* **2020**, *24*, 83–99. [CrossRef]
31. Liu, L.; Fei, J.; Yang, X. Adaptive Interval Type-2 Fuzzy Neural Network Sliding Mode Control of Nonlinear Systems Using Improved Extended State Observer. *Mathematics* **2023**, *11*, 605. [CrossRef]
32. Wu, D.R.; Mendel, J.M. Enhanced Karnik–Mendel Algorithms. *IEEE Trans. Fuzzy Syst.* **2009**, *17*, 923–934.
33. Garibaldi, J.M.; Jaroszewski, M.; Musikasuwan, S. Nonstationary Fuzzy Sets. *IEEE Trans. Fuzzy Syst.* **2008**, *16*, 1072–1086. [CrossRef]
34. Pal, N.R.; Saha, S. Simultaneous Structure Identification and Fuzzy Rule Generation for Takagi-Sugeno Models. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2008**, *38*, 1626–1638. [CrossRef]
35. Wu, D.; Yuan, Y.; Huang, J.; Tan, Y. Optimize TSK Fuzzy Systems for Regression Problems: Minibatch Gradient Descent With Regularization, DropRule, and AdaBound (MBGD-RDA). *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 1003–1015. [CrossRef]
36. Zhang, N.; Xue, X.; Xia, X.; Jiang, W.; Zhang, C.; Peng, T.; Shi, L.; Li, C.; Zhou, J. Robust T-S Fuzzy Model Identification Approach Based on FCRM Algorithm and L1-Norm Loss Function. *IEEE Access* **2020**, *8*, 33792–33805. [CrossRef]
37. Shi, Z.L.; Li, X.P.; Li, W.; Yan, T.; Wang, J.; Fu, Y. Robust Low-Rank Matrix Recovery as Mixed Integer Programming via ℓ_0-Norm Optimization. *IEEE Signal Process. Lett.* **2023**, *30*, 1012–1016. [CrossRef]
38. Wang, J.; Zhang, H.; Wang, J.; Pu, Y.; Pal, N.R. Feature Selection Using a Neural Network With Group Lasso Regularization and Controlled Redundancy. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 1110–1123. [CrossRef] [PubMed]
39. UCI Machine Learning Repository. 2020. Available online: http://archive.ics.uci.edu/ml (accessed on 5 September 2023).
40. Shalabi, L.A.; Shaaban, Z.; Kasasbeh, B. Data Mining: A Preprocessing Engine. *J. Comput. Sci.* **2006**, *2*, 735–739. [CrossRef]
41. Wang, J.; Zhang, B.; Sun, Z.; Hao, W.; Sun, Q. A novel conjugate gradient method with generalized Armijo search for efficient training of feedforward neural networks. *Neurocomputing* **2018**, *275*, 308–316. [CrossRef]