Local Optimality of Self-Organising Neuro-Fuzzy Inference Systems

Xiaowei Gu [1,2], Plamen Angelov [1,2,3], Hai-Jun Rong [4]

1. School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK

2. Lancaster Intelligent, Robotic and Autonomous Systems Centre (LIRA), Lancaster University, UK

3. Technical University, Sofia, 1000, Bulgaria (Honorary Professor)

4. State Key Laboratory for Strength and Vibration of Mechanical Structures, Shaanxi Key Laboratory of Environment and Control for Flight Vehicle, School of Aerospace, Xi'an Jiaotong University, Shaanxi, China 710049

Email: x.gu3@lancaster.ac.uk, p.angelov@lancaster.ac.uk, hjrong@mail.xjtu.edu.cn

Abstract: Optimality of the premise, IF part is critical to a zero-order evolving intelligent system (EIS) because this part determines the validity of the learning results and overall system performance. Nonetheless, a systematic analysis of optimality has not been done yet in the state-of-the-art works. In this paper, we use the recently introduced self-organising neuro-fuzzy inference system (SONFIS) as an example of typical zero-order EISs and analyse the local optimality of its solutions. The optimality problem is firstly formulated in a mathematical form, and detailed optimality analysis is conducted. The conclusion is that SONFIS does not generate a locally optimal solution in its original form. Then, an optimisation method is proposed for SONFIS, which helps the system to attain local optimality in a few iterations using historical data. Numerical examples presented in this paper demonstrate the validity of the optimality analysis and the effectiveness of the proposed optimisation method. In addition, it is further verified numerically that the proposed concept and general principles can be applied to other types of zero-order EISs with similar operating mechanisms.

Keywords- local optimality, neuro-fuzzy system, evolving intelligent system, self-organising, data partitioning.

1. Introduction

Evolving intelligent systems (EISs) [1], as a typical form of multi-model systems, are viewed as a powerful tool for handling complex problems with both measurement and motion uncertainties, and have demonstrated success in many real-world application scenarios [28]. By exploiting the divide-and-conquer strategy, multi-model systems are capable of decomposing complex problems into a set of simpler ones that can be approximated using simple local models through data partitioning [8]. As the main goal of a multi-model system is to partition the data space and identify underlying data patterns through local partitions, the optimality of the solution is of paramount importance to the multi-model system because it determines the validity and effectiveness of the learning results. Nonetheless, to the best of the authors' knowledge, optimality analysis has not been touched in the state-of-the-art works in the area of EISs [27],[31].

Self-organising neuro-fuzzy inference system (SONFIS) was recently introduced in [18] as a generic approach for data and image classification, and it can be further extended to various problems and applications including, but not limited to, online data analytics, prediction, etc. As a new type of zero-order AnYa type fuzzy rule-based systems [5], SONFIS is able to self-organise and self-evolve its multi-model system structure from empirically observed data in a non-iterative, computationally lean and objective manner. It is composed of a set of highly transparent, massively parallel IF…THEN rules consisting of meaningful prototypes that represent local peaks of the multimodal data distribution. SONFIS does not impose any models with parameters on data generation, and the learning process only involves nonparametric statistic operators, which can objectively disclose the ensemble properties and mutual distributions of data [4]. After being primed offline, SONFIS can further continuously self-update its system structure and meta-parameters with new observations from data streams to follow the potential shifts and/or drifts of data patterns [29]. Thanks to its prototype-based nature, SONFIS performs classification on unlabelled data in a human-like reasoning style following the well-known "winner takes all" principles.

Prototypes play an instrumental role in SONFIS because they represent local models of data patterns in terms of multimodal data density [4]. In this paper, we conduct a detailed mathematical analysis on the

optimality of the premise, IF part of SONFIS, namely, prototypes. Starting with the mathematical formulation of the problem, we firstly investigate the optimality of data partitioning solutions obtained by SONFIS and prove that the system does not yield an optimal solution. Then, we introduce a highly efficient optimisation method that enables SONFIS to always attain local optimality in few iterations. This, in turn, effectively improves its classification performance. Moreover, the optimality analysis and the optimisation method presented in this paper are not limited to SONFIS, but are more generic and applicable to other online, non-iterative machine learning algorithms with similar operating mechanisms, which include, but not limited to, autonomous learning multi-model (ALMMo) neuro-fuzzy systems [3],[19], eClass0 [6]. Numerical examples presented in this paper justify the validity of the optimality analysis and further demonstrate that the proposed optimisation method can effectively improve the classification accuracy of SONFIS on various challenging benchmark problems with minor additional computational cost. In addition, numerical results also show that the proposed optimisation method substantially enhances the performance of other types of zero-order EISs with similar operating mechanisms.

The key contributions of this paper include: *1)* the mathematical formulation of the optimality problem of zero-order EISs; *2)* a detailed analysis on the optimality of the solutions obtained by zero-order EISs; *3)* a generic method for autonomously optimising the premise, IF part of zero-order EISs based on historical data; *4)* a general strategy for zero-order EISs to obtain the locally optimal solutions and effectively enhance the classification performance.

The remainder of this paper is organised as follows. Section 2 provides a critical review of related works. The architecture, learning and validation processes of SONFIS are briefed in Section 3. Section 4 presents the optimality analysis. The method, which provides the locally optimal solution, is described in section 5. Numerical examples are given in section 6. Section 7 concludes this paper and gives the direction for future work.

2. Related Works

Prototype-based systems have been widely used for multi-class classification purposes [3],[18]. Prototypes play a key role in such systems, and it is the prototype selection process that determines their performance, transparency and computational efficiency. Well-known prototype-based classifiers include support vector machine (SVM) [10], learning vector quantization (LVQ) [23] and self-organising map (SOM) [30], etc. SVM iteratively selects prototypes, namely, support vectors, from observed data samples to identify the maximum-margin hyperplane in the data space. LVQ and SOM, just like other types of ANNs, gradually approach the optimal solution in the entire data space by minimising the objective function. Another widely used classifier, K-nearest neighbour (KNN) [35], can be viewed as an extreme example of prototype-based systems as well. Nonetheless, KNN is very different from SVM, LVQ and SOM in the sense that all the data samples are stored in the memory and treated as prototypes.

EISs, as a key branch of computational intelligence, are becoming increasingly popular owing to their highly transparent system structure and the explainable learning and decision-making processes [11]. EISs can be implemented in the form of neuro-fuzzy or rule-based models [1]. However, unlike the other typical type of multi-model systems, namely, ANNs [17], most of the EISs are designed for processing streaming data “on the fly”, and they are able to self-update and self-evolve their system structure and meta-parameters to follow the rapidly changing data patterns of data streams [29]. Currently, EISs have been successfully implemented for various real-world applications including classification [33], prediction [20], control [34], anomaly detection [27], etc. The most popular (neuro-) fuzzy systems include, but not limited to, eTS [2], DENFIS [22], eClass [6], SAFIS [36], PANFIS [31], GENIFS [32] and IT2FNN [41]. Interested readers are referred to the recent surveys [28],[40] for more details.

Zero-order EISs [3],[6],[18] are based on prototypes, and they, generally, have a simpler, more flexible and transparent system structure compared with other types of EISs, e.g. first-order [19] and higher-order ones [41]. Thanks to the prototype-based nature, they are highly computational efficient and capable of handling complex multi-class classification problems. Despite that the operating mechanisms might be very different, prototypes of zero-order EISs are usually identified as the most representative data samples through data

partitioning. Prototypes determine the validity and effectiveness of the systems and significantly influence their performance. Compared with other types of prototype-based systems, e.g., SVM, SOM and LVQ, the prototypes of zero-order EISs are usually obtained through an "one pass" learning process without iteratively searching the data space for optimal solutions. Many zero-order EISs are also capable of continuously updating existing prototypes based on newly observed data samples and adding new prototypes to follow new data patterns. Nonetheless, the optimality of zero-order EISs remains a question, and no mathematical analysis has not been conducted yet [19]. The main reason for this is that zero-order EISs are designed for learning from nonstationary, complex data streams in an efficient, non-iterative and "one pass" manner. Nonetheless, a systematic study on system optimality is of paramount importance and required to be done in order to better understand the operating mechanisms of zero-order EISs and further improve their performance.

Another problem that the majority of zero-order EISs [3],[6] (as well as many other machine learning algorithms, e.g., first-order EISs [19], type-2 EISs [33], K-means [38], Laplacian SVM [15]) suffer from is the need of problem- and user-specific parameters. The structural learning algorithms for the premise, IF part of EISs usually require certain parameters to be defined in advance, and this is a challenging issue for real-world applications. The prototype identification results of an EIS might vary significantly by using different parameter settings, and this can significantly influence the performance and objectiveness of the system. Properly predefining such parameters requires certain levels of prior knowledge about the problems from users, and, sometimes, assumptions on data generation model are also needed to be made. However, in real-world scenarios, prior knowledge is usually very limited, while the assumptions are seldom hold true, especially for streaming data.

As a recently introduced generic classification approach, SONFIS [18] employs nonparametric statistical operators to objectively disclose the underlying data patterns behind the empirically observed data samples, and extracts local peaks from the multimodal data distribution as prototypes. SONFIS is nonparametric and highly objective in the sense that no generation model with parameters is imposed on data, and all the involved meta-parameters are directly derived from data without prior knowledge of the problems. It is able to approach any problem at different levels of granularity, in other words, different levels of details depending on the complexity of the problems, availability of computational resources and particular needs from users. Moreover, SONFIS supports both offline and online learning modes and can use various types of distance/dissimilarity measures for classification. Thus, SONFIS has a strong adaptive ability and has demonstrated the state-of-the-art performance on various problems. As a highly representative and well-performing zero-order EIS, in this paper, we will use SONFIS as an example and conduct optimality analysis on its solutions.

3. SONFIS

In this section, we will briefly describe the architecture, learning process and validation process of SONFIS [18] to make this paper self-contained, which also serve as the foundation for the optimality analysis conducted in the next section.

First of all, let $\{\boldsymbol{x}\}_K = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_i, \dots, \boldsymbol{x}_K\}$ ($\boldsymbol{x}_i = \left[x_{i,1}, x_{i,2}, \dots, x_{i,M}\right]^T \in \mathbf{R}^M$) be a particular data set/stream in a real metric space, $\mathbf{R}^M$, with the dimensionality of M. The subscript i indicates the time instance at which $\boldsymbol{x}_i$ is observed. We assume that this data set/stream is composed of data samples of C different classes. At the *Kth* time instance, the set of observed data samples is denoted as $\{\boldsymbol{x}\}_K$. Based on their class labels, these data samples can be further divided into C subsets denoted by $\{\boldsymbol{x}\}_{N^j}^j = \left\{\boldsymbol{x}_1^j, \boldsymbol{x}_2^j, \dots, \boldsymbol{x}_{N^j}^j\right\}$ $(j = 1,2, \dots, C)$, where the superscript j denotes the *jth* class and there is $\sum_{j=1}^{C} N^j = K$. For each subset, $\{\boldsymbol{x}\}_{N^j}^j$, we further consider that some samples may share the same values, i.e., $\boldsymbol{x}_m^j = \boldsymbol{x}_n^j$ and $m \neq n$. Thus, we denote the set of unique data samples of the *jth* class ($j = 1,2, \dots, C$) as $\{\boldsymbol{u}\}_{L^j}^j = \left\{\boldsymbol{u}_1^j, \boldsymbol{u}_2^j, \dots, \boldsymbol{u}_{L^j}^j\right\}$, and the corresponding occurrence frequencies are denoted as $\{f\}_{L^j}^j = \left\{f_1^j, f_2^j, \dots, f_{L^j}^j\right\}$. Furthermore, there are $\{\boldsymbol{u}\}_{L^j}^j \subseteq \{\boldsymbol{x}\}_{N^j}^j$; $L^j \leq N^j$ and $\sum_{k=1}^{L^j} f_k^j = N^j$. Unless specifically declared otherwise, all the mathematical derivations in the remainder of this paper are conducted at the *Kth* time instance by default. Without loss of generality, we use Euclidean distance for derivation. However, it has to be stressed that SONFIS can support various types of distance/dissimilarity

measures [18]. For clarity, we summarise the key notations of this paper and the respective definitions in Table 1.

Table 1. Definitions of key notations

Notations	Definitions
G	Level of granularity
M	Dimensionality of the real metric space
$\mathbf{R}^M$	M-dimensional real metric space
K	Number of observed data samples/current time instance
C	Number of classes
$\{\boldsymbol{x}\}_K$	Collection of observed data samples at the *Kth* time instance
$\boldsymbol{x}_i$	$M \times 1$ dimensional data sample observed at the *ith* time instance
N^j	Number of observed data samples of the *jth* class at the *Kth* time instance
L^j	Number of observed unique data samples of the *jth* class at the *Kth* time instance
$\{\boldsymbol{x}\}_{N^j}^j$	Collection of observed data samples of the *jth* class at the *Kth* time instance
$\boldsymbol{x}_i^j$	The *ith* $M \times 1$ dimensional data sample of the *jth* class
$\{\boldsymbol{u}\}_{L^j}^j$	Collection of observed unique data samples of the *jth* class at the *Kth* time instance
$\boldsymbol{u}_i^j$	The *ith* $M \times 1$ dimensional unique data sample of the *jth* class
f_i^j	Occurrence frequency of $\boldsymbol{u}_i^j$
$\boldsymbol{\mu}_{N^j}^j$	Mean of $\{\boldsymbol{x}\}_{N^j}^j$
$X_{N^j}^j$	Mean of $\{\|\boldsymbol{x}\|^2\}_{N^j}^j$
P^j	Number of identified data clouds/prototypes of the *jth* class
$\{\mathbb{C}\}^j$	Collection of data clouds of the *jth* class
$\mathbb{C}_i^j$	The *ith* data cloud of the *jth* class
$\{\boldsymbol{p}\}^j$	Collection of prototypes of the *jth* class
$\boldsymbol{p}_i^j$	The *ith* prototype of the *jth* class with the dimensionality of $M \times 1$
S_i^j	Number of members of $\mathbb{C}_i^j$
$\boldsymbol{P}^j$	$M \times P^j$ dimensional matrix form of $\{\boldsymbol{p}\}^j$
$\boldsymbol{W}^j$	$P^j \times N^j$ dimensional weight matrix
γ_G^j	Radius of local influential areas of data clouds of the *jth* class corresponding to the *Gth* level of granularity
λ^j	Score of confidence of the *jth* massively parallel fuzzy rule

3.1. Multi-Model Architecture

An illustrative diagram of the multi-model architecture of SONFIS is depicted in Fig. 1. Fig. 1 (a) depicts the structure of SONFIS during the system identification process; Fig. 1(b) gives the system structure during the validation stage; Fig. 1(c) is the zoom-in structure of the *jth* massively parallel fuzzy rule.

It is demonstrated in Fig. 1 that SONFIS consists of C massively parallel fuzzy rules. During the learning stage, the C fuzzy rules are trained in parallel using data samples from the respective classes (one rule per class). Each rule is composed of a number of prototypes identified from data samples of the corresponding class, and these prototypes are connected by logical "OR" connectives ($j = 1,2, \dots, C$):

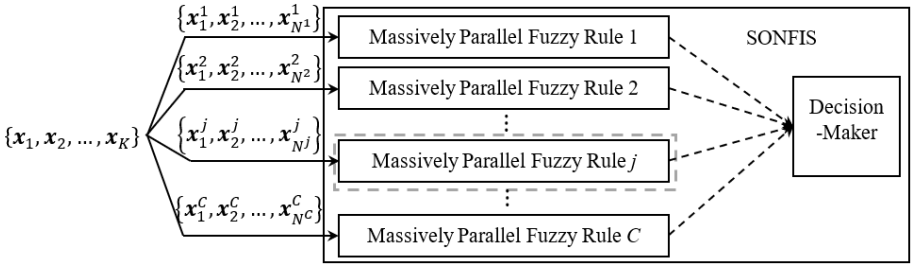
$$\mathbf{R}^j: \quad IF\ \left(\boldsymbol{x}\sim\boldsymbol{p}_1^j\right)\ OR\ \left(\boldsymbol{x}\sim\boldsymbol{p}_2^j\right)\ OR\ \ldots\ OR\ \left(\boldsymbol{x}\sim\boldsymbol{p}_{P^j}^j\right)\quad THEN\ (Class\ j) \tag{1a}$$

where $\boldsymbol{p}_i^j$ is the *ith* prototype of the *jth* fuzzy rule; P^j is the number of identified prototypes. As a result, each massively parallel fuzzy rule can be viewed as a combination of multiple simpler fuzzy rules with singleton consequences in the form of equation (1b) ($i = 1,2,\ldots,P^j$):

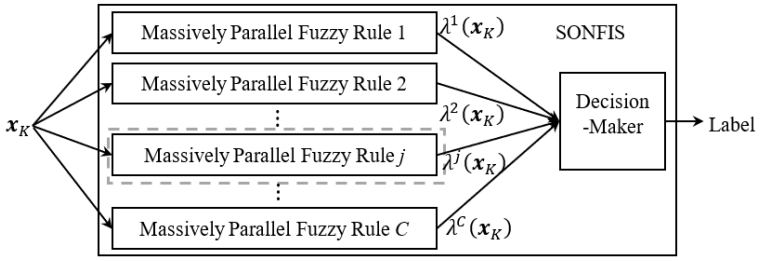
$$\mathbf{R}_i^j: \quad IF\ \left(\boldsymbol{x}\sim\boldsymbol{p}_i^j\right)\quad THEN\ (Class\ j) \tag{1b}$$

During the validation process, for each unlabelled data sample, every massively parallel fuzzy rule within the rule base will produce a score of confidence on it and, in total, C scores will be generated. The overall decision-maker will, then, estimate the class label for the sample based on the scores following the "winner takes all" principle.

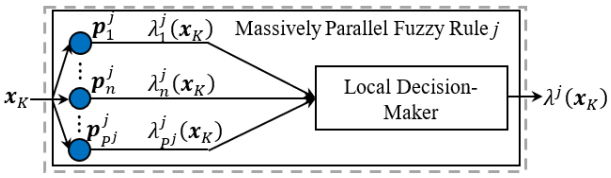
The learning and validation processes of SONFIS will be detailed in the following two subsections, respectively.



(a) The multi-model architecture (identification stage)



(b) The multi-model architecture (validation stage)



(c) Zoom-in structure of the j^{th} fuzzy rule

Fig. 1. Multi-model architecture of SONFIS

3.2. Identification Process

In this subsection, the identification process of SONFIS is presented [18]. SONFIS is capable of self-organising its fuzzy rule base from static data and then, continuing to self-evolve its system structure and self-update the meta-parameters with streaming data recursively. Therefore, the offline learning process will be presented first followed by the online learning process. As each massively parallel fuzzy rule is trained separately, we use the *jth* rule as an example for demonstration. The same principles can be applied to all other fuzzy rules within the rule base. The level of granularity of SONFIS is set as G, which can be any positive integers.

A. Offline Learning Process

The main algorithmic procedure of the offline learning process is as follows.

Stage 1. Forming Voronoi tessellation from data

In this stage, the observed unique data samples of the *jth* class are, firstly, ranked in an indexing list based on their mutual distances and ensemble properties. Firstly, the multimodal densities D^{MM} at all unique data samples, $\{\boldsymbol{u}\}_{Lj}^{j}$ are calculated by equation (2) ($k = 1,2,\dots,L^j$) [4]:

$$D^{MM}\left(\boldsymbol{u}_k^j\right) = f_k^j \frac{1}{1+\frac{\left\|\boldsymbol{u}_k^j-\boldsymbol{\mu}_{Nj}^j\right\|^2}{X_{Nj}^j-\left\|\boldsymbol{\mu}_{Nj}^j\right\|^2}} \tag{2}$$

where $\boldsymbol{\mu}_{Nj}^j$ and X_{Nj}^j are the means of $\{\boldsymbol{x}\}_{Nj}^j$ and $\{\|\boldsymbol{x}\|^2\}_{Nj}^j$, respectively, which can be calculated by the following expressions [4]:

$$\boldsymbol{\mu}_{Nj}^j = \frac{1}{N^j}\sum_{k=1}^{N^j}\boldsymbol{x}_k^j; \quad X_{Nj}^j = \frac{1}{N^j}\sum_{k=1}^{N^j}\left\|\boldsymbol{x}_k^j\right\|^2 \tag{3}$$

and $\|\boldsymbol{x}\|$ denotes the Euclidean norm of $\boldsymbol{x}$: $\|\boldsymbol{x}\| = \sqrt{\sum_{i=1}^{M} x_i^2}$.

The expression of multimodal density (equation (2)) is very fundamental because it combines information about repeated data values and the scattering across the data space, and it is derived directly from data resembling the well-known probability mass function [4]. When Euclidean distance or some other types of well-known distances and dissimilarity, e.g., Mahalanobis distance, cosine dissimilarity, is used for calculation, it can be recursively calculated in an elegant form [4],[18].

In this paper, we use the real climate data[1] measured in Manchester, UK for the period 2010-2015 as an example to visualise the concept. This dataset is composed of 938 data samples, 479 of which are measured in winter (class 1) and the rest are obtained during summer (class 2). For visual clarity, we only consider the first two attributes, namely, temperature, ^oC (x_1) and wind speed, *mph* (x_2). The multimodal density of the first 70% of data samples is visualised in Fig. 2.

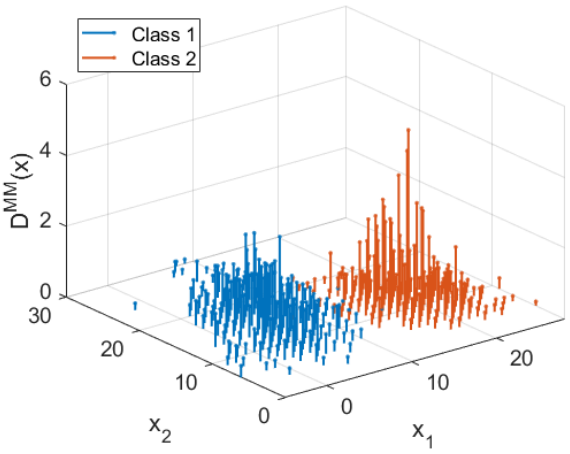


Fig. 2. Multimodal density of real climate data

After the multimodal density values at all unique data samples have been calculated, the unique data sample with the highest multimodal density value is identified as: $\boldsymbol{r}_1 = \underset{k=1,2,\dots,L^j}{\operatorname{argmax}}\left(D^{MM}\left(\boldsymbol{u}_k^j\right)\right)$, and it is set as the

[1] Available from: http://www.worldweatheronline.com

first element of the indexing list, denoted by $\{\boldsymbol{r}\}$. The remaining elements of $\{\boldsymbol{r}\}$ are identified one by one using the following principle ($k = 2,3, \dots, L^j$):

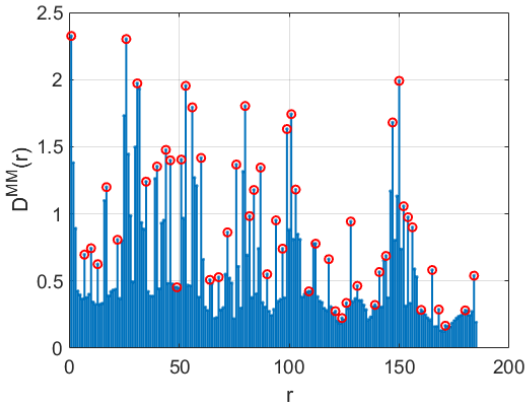
$$\boldsymbol{r}_k = \underset{\boldsymbol{u}\in\{\boldsymbol{u}\}_{Lj}^{j};\boldsymbol{u}\neq\boldsymbol{r}_1,\boldsymbol{r}_2,\dots,\boldsymbol{r}_{k-1}}{\operatorname{argmin}} \left(\|\boldsymbol{u}-\boldsymbol{r}_{k-1}\|^2\right) \tag{4}$$

Once the full indexing list $\{\boldsymbol{r}\}$ is built by equation (4), the ranked multimodal density $\{D^{MM}(\boldsymbol{r})\}$ is obtained accordingly. Based on $\{D^{MM}(\boldsymbol{r})\}$, we can identify the local maxima of multimodal density by using the following condition [18]:

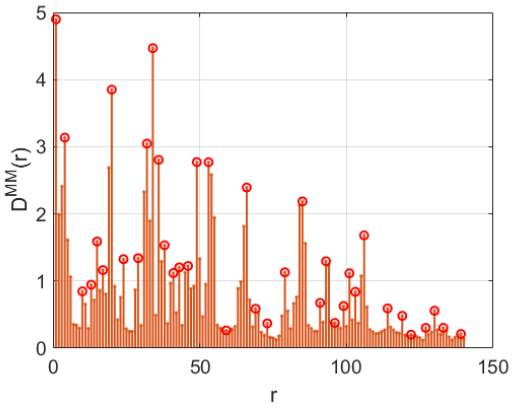
$$\textbf{\textit{Condition 1}}: \quad \begin{array}{c} If\ \left(sgn\left(D^{MM}(\boldsymbol{r}_k)-D^{MM}(\boldsymbol{r}_{k+1})\right)=1\right)\ and\ \left(sgn\left(D^{MM}(\boldsymbol{r}_k)-D^{MM}(\boldsymbol{r}_{k-1})\right)=1\right) \\ Then\ (\boldsymbol{r}_k\ is\ a\ local\ maximum\ of\ D^{MM}) \end{array} \tag{5}$$

where $sgn(x)=\begin{cases}1 & x>0\\ 0 & x=0\\ -1 & x<0\end{cases}$ is the sign function. The local maxima of D^{MM} are denoted as $\{\boldsymbol{u}^*\}$, and the cardinality is denoted as L^{j*}. By ranking and reordering the unique data samples in terms of their mutual distances (equation (4)) and their corresponding multimodal density values, the searching process for local maxima of the multimodal data distribution is significantly simplified because the original M-dimensional data space, $\mathbf{R}^M$ is reduced to a 1-dimensional indexing list, $\{\boldsymbol{r}\}$.

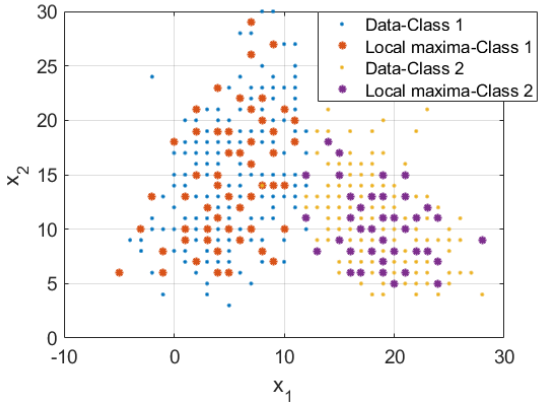
The ranked multimodal density of the real climate data as given by Fig. 2 are depicted in Figs. 3(a) and 3(b), where the identified local maxima are marked by red circles. The locations of these local maxima in the 2D data space are also visualized in Fig. 3(c).



(a) Class 1



(b) Class 2



(c) Local maxima in the data space

Fig. 3. The multimodal density of two-class real climate data and the identified local maxima

Then, Voronoi tessellations are formed in the data space by using these identified local maxima as prototypes to attract nearby data samples, which results in a number of data clouds, denoted by $\{\mathbb{C}\}$:

$$\mathbb{C}_{n*} \leftarrow \mathbb{C}_{n*} \cup \{\boldsymbol{x}_k^j\}; \quad n^* \leftarrow \underset{\boldsymbol{y}\in\{\boldsymbol{u}^*\}}{\operatorname{argmin}} \left(\left\|\boldsymbol{y} - \boldsymbol{x}_k^j\right\|^2\right) \tag{6}$$

where $k = 1,2,\ldots,N^j$. Equation (6) naturally partitions the data space based on mutual distances between data samples and these prototypes. This process is free from any threshold and, the partitioning result objectively approximates the real data distribution.

Stage 2. Deriving the radius of local influential area around prototypes

In this stage, the radius of local influential area around each prototype is derived based on the mutual distances of these observed data samples and the level of granularity defined by users. The radius can be viewed as an estimation of the average distance between any two strongly connected prototypes under a specific level of granularity, and it condenses the mutual distribution information extracted from the empirically observed data.

Under the first level of granularity ($G = 1$), the radius of local influential area, γ_1^j is calculated by [18]:

$$\gamma_1^j = \frac{1}{Q_1^j}\sum_{\boldsymbol{x},\boldsymbol{y}\in\{\boldsymbol{x}\}_{N^j}^j;\boldsymbol{x}\neq\boldsymbol{y};\ \|\boldsymbol{y}-\boldsymbol{x}\|^2\leq\bar{d}_{N^j}^j} \|\boldsymbol{y}-\boldsymbol{x}\|^2 \tag{7}$$

where Q_1^j is the number of pairs of data samples of $\{\boldsymbol{x}\}_{N^j}^j$ between which the distance is smaller than the average distance between any two data samples of the *jth* class, $\bar{d}_{N^j}^j$ [4], namely:

$$\bar{d}_{N^j}^j = 2\left(X_{N^j}^j - \left\|\boldsymbol{\mu}_{N^j}^j\right\|^2\right) \tag{8}$$

From the second level to any higher level of granularity ($G = 2,3,\ldots.$), one can calculate the radius iteratively by using the following expression [18]:

$$\gamma_G^j = \frac{1}{Q_G^j}\sum_{\boldsymbol{x},\boldsymbol{y}\in\{\boldsymbol{x}\}_{N^j}^j;\boldsymbol{x}\neq\boldsymbol{y};\ \|\boldsymbol{y}-\boldsymbol{x}\|^2\leq\gamma_{G-1}^j} \|\boldsymbol{y}-\boldsymbol{x}\|^2 \tag{9}$$

where γ_G^j and γ_{G-1}^j represent the radii of local influential area corresponding to the *Gth* and *(G-1)th* levels of granularity, respectively; Q_G^j is the number of pairs of data samples between which the distance is smaller than γ_{G-1}^j.

Compared with using predefined threshold or hard-coding principles, deriving the radius of local influential area around each prototype in such a way has two strong advantages Firstly, γ_G^j is guaranteed to be valid all the time because it is derived from data directly and always meaningful. Secondly, γ_G^j can be decided without any prior knowledge, but only based on users' preferences and/or any specific requirements of the problems [18]. This significantly strengthens the applicability and adaptive ability of SONFIS to any real-world problems.

In general, with a high level of granularity, SONFIS is able to extract finer details from data and identify more prototypes, and it usually demonstrates better performance in terms of classification accuracy. However, in such cases, SONFIS can consume more computational and memory resources, and the problem of overfitting may occur as well. In contrast, using a low level of granularity may largely improve the computational and memory efficiency of SONFIS, but may also deteriorate its classification performance on complex, large-scale and high-dimensional problems.

Stage 3. Identifying prototypes from local maxima

In this stage, prototypes of the *jth* class are filtered out from the local maxima identified during Stage 1. Firstly, raw prototypes, denoted by $\{\boldsymbol{q}\}$, are derived as the centres of data clouds, $\{\mathbb{C}\}$:

$$\boldsymbol{q}_k = \frac{1}{S_k^j}\sum_{\boldsymbol{x}\in\mathbb{C}_k}\boldsymbol{x}; \quad \boldsymbol{q}_k \in \{\boldsymbol{q}\} \tag{10}$$

where S_k^j is the cardinality (number of members) of $\mathbb{C}_k$; $\mathbb{C}_k \in \{\mathbb{C}\}$; $k = 1,2,\dots,L^{j*}$. The multimodal density values at the raw prototypes $\{\boldsymbol{q}\}$ are calculated using equation (11):

$$D^{MM}(\boldsymbol{q}_k) = S_k^j \frac{1}{1+\frac{\|q_k-\mu^j\|^2}{x^j-\|\mu^j\|^2}}; \quad \boldsymbol{q}_k \in \{\boldsymbol{q}\} \tag{11}$$

Then, for each raw prototype (i.e. the *kth* one), its neighbouring prototypes (the collection is denoted by $\{\boldsymbol{q}\}_k^{n*}$) are identified by the following condition ($k = 1,2,\dots,L^{j*}$):

$$\textbf{\textit{Condition 2}}: \quad If\ \left(\|\boldsymbol{q}_k - \boldsymbol{q}_i\|^2 \le \gamma_G^j\right) \quad Then\ (\boldsymbol{q}_i \in \{\boldsymbol{q}\}_k^{n*}) \tag{12}$$

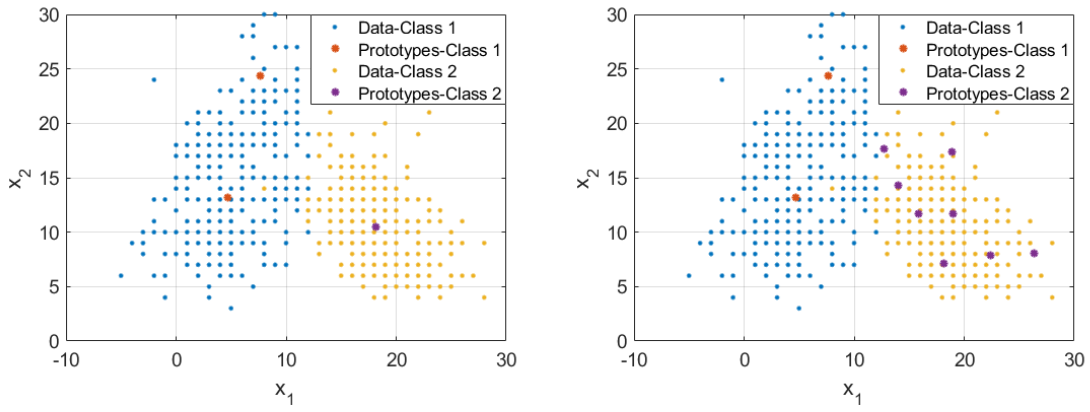
where $\boldsymbol{q}_i \neq \boldsymbol{q}_k$ and $\boldsymbol{q}_i \in \{\boldsymbol{q}\}$. **Condition 2** defines a local influential area around each prototype with the radius of γ_G^j and uses this further to identify neighbouring prototypes. This provides an intuitive understanding of mutual distributions and ensemble properties of data.

After that, the most representative prototypes of the *jth* class denoted by $\{\boldsymbol{p}\}^j$ are selected out from raw prototypes (local maxima) based on the following condition:

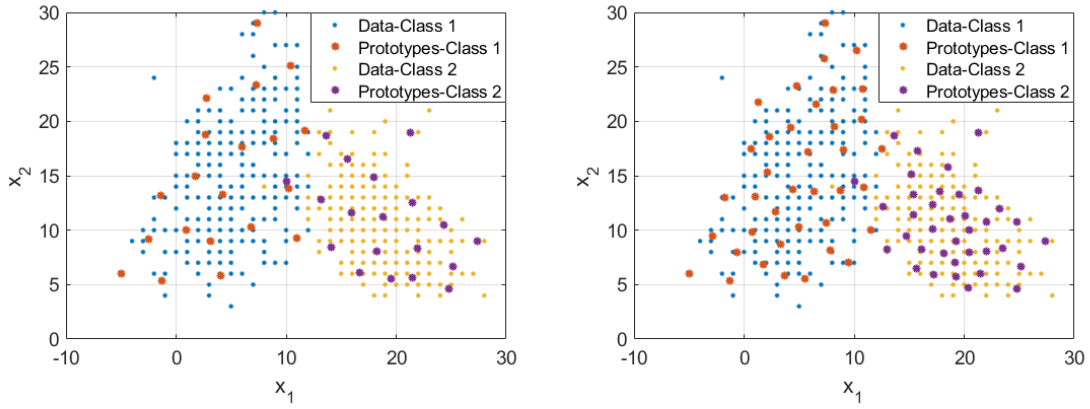
$$\textbf{\textit{Condition 3}}: \quad If\ \left(D^{MM}(\boldsymbol{q}_k) > \max_{\boldsymbol{q}\in\{\boldsymbol{q}\}_k^{n*}}\left(D^{MM}(\boldsymbol{q})\right)\right) \quad Then\ (\boldsymbol{q}_k \in \{\boldsymbol{p}\}^j) \tag{13}$$

and new data clouds are formed around prototypes $\{\boldsymbol{p}\}^j = \left\{\boldsymbol{p}_1^j, \boldsymbol{p}_2^j, \dots, \boldsymbol{p}_{pj}^j\right\}$ resembling Voronoi tessellation using equation (6), denoted by $\{\mathbb{C}\}^j = \left\{\mathbb{C}_1^j, \mathbb{C}_2^j, \dots, \mathbb{C}_{pj}^j\right\}$. **Condition 3** effectively filters out more representative prototypes by comparing these prototypes with their neighbours in terms of their values of multimodal density. **Conditions 2 and 3** together greatly facilitate the searching process for the most representative data samples in the data space. At the same time, these highly representative prototypes objectively reflect the ensemble properties and mutual distribution of data because only the mutual distances and multimodal density values of these empirically observed data samples are considered during the identification process. The identified prototypes by SONFIS in the data space under different levels of granularity are visualised in Fig. 4 using the same climate data as given in Fig. 2.

In the end, the *jth* massively parallel fuzzy rule $\mathbf{R}^j$ is created with $\{\boldsymbol{p}\}^j$ (the number of elements in $\{\boldsymbol{p}\}^j$ is denoted by P^j) in the same form as equation (1a).



(a) $G = 1$ (b) $G = 2$



(c) $G = 3$ (d) $G = 4$

Fig. 4. Identified prototypes from data under different levels of granularity during offline learning process

B. Online Learning Process

Once the offline learning process is finished, SONFIS can further self-update its structure and meta-parameters recursively with newly observed data on a sample-by-sample basis. This online learning ability allows SONFIS to rapidly adapt to new data patterns. It has to be stressed that SONFIS does not require any user-control parameters to be pre-determined for online learning, but uses nonparametric statistic operators and data-driven thresholds derived during the offline learning process.

Let us assume that SONFIS has been primed with the static dataset, $\{\boldsymbol{x}\}_K$ at the *Kth* time instance, and at the next time instance ($K \leftarrow K+1$), a new data sample of the *jth* class arrived ($N^j \leftarrow N^j + 1$). The online sample-by-sample learning process is described as follows.

Stage 4. Updating global meta-parameters

SONFIS firstly updates the global meta-parameters $\boldsymbol{\mu}^j_{N^j-1}$ and $X^j_{N^j-1}$ using $\boldsymbol{x}^j_{N^j}$ as:

$$\boldsymbol{\mu}^j_{N^j} \leftarrow \frac{N^j-1}{N^j}\boldsymbol{\mu}^j_{N^j-1} + \frac{1}{N^j}\boldsymbol{x}^j_{N^j}; \quad X^j_{N^j} \leftarrow \frac{N^j-1}{N^j}X^j_{N^j-1} + \frac{1}{N^j}\left\|\boldsymbol{x}^j_{N^j}\right\|^2 \tag{14}$$

and the radius of local influential area, γ^j_G is updated in a recursive manner based on the ratio between $\bar{d}^j_{N^j}$ and $\bar{d}^j_{N^j-1}$, namely [18],

$$\gamma^j_G \leftarrow \frac{\bar{d}^j_{N^j}}{\bar{d}^j_{N^j-1}}\gamma^j_G = \frac{X^j_{N^j} - \left\|\boldsymbol{\mu}^j_{N^j}\right\|^2}{X^j_{N^j-1} - \left\|\boldsymbol{\mu}^j_{N^j-1}\right\|^2}\gamma^j_G \tag{15}$$

The main reason for using equation (15) to update γ^j_G is because it provides a good approximation and is more computationally efficient. Equations (7)-(9) derive the radius of local influential area, γ^j_G derived from all static data samples during the offline learning process based on their mutual distances. However, it would be time consuming and a waste of computational resources to repeat the same process for each newly observed data sample during the online learning process. Thus, we use equation (15) instead.

Stage 5. Updating the identified prototypes

In this stage, data density values at $\boldsymbol{x}^j_{N^j}$ and all existing prototypes, $\{\boldsymbol{p}\}^j$ are calculated by the following expression [4]:

$$D(\boldsymbol{y}) = \frac{1}{1+\frac{\left\|\boldsymbol{y}-\boldsymbol{\mu}^j_{N^j}\right\|^2}{X^j_{N^j} - \left\|\boldsymbol{\mu}^j_{N^j}\right\|^2}} \tag{16}$$

where $\boldsymbol{y} = \boldsymbol{x}_{Nj}^{j}, \boldsymbol{p}_{1}^{j}, \boldsymbol{p}_{2}^{j}, \ldots, \boldsymbol{p}_{Pj}^{j}$.

Then, the nearest prototype to $\boldsymbol{x}_{Nj}^{j}$, denoted by $\boldsymbol{p}_{n*}^{j}$, is identified by equation (6), and the following condition is used for checking whether $\boldsymbol{x}_{Nj}^{j}$ can be a new prototype:

$$\textbf{Condition 4}: \quad \begin{array}{c} If\ \left(D\left(\boldsymbol{x}_{Nj}^{j}\right) > \max\limits_{\boldsymbol{q}\in\{\boldsymbol{p}\}^{j}}\left(D(\boldsymbol{q})\right)\right)\ or\ \left(D\left(\boldsymbol{x}_{Nj}^{j}\right) < \min\limits_{\boldsymbol{q}\in\{\boldsymbol{p}\}^{j}}\left(D(\boldsymbol{q})\right)\right) \\ or\ \left(\left\|\boldsymbol{x}_{Nj}^{j} - \boldsymbol{p}_{n*}^{j}\right\|^{2} > \gamma_{G}^{j}\right) \\ Then\left(\boldsymbol{x}_{Nj}^{j}\ becomes\ a\ new\ prototype\right) \end{array} \tag{17}$$

Condition 4 is a combination of Conditions 4 and 5 in the original version of SONFIS [6], and the rationale behind it is very clear. If $D\left(\boldsymbol{x}_{Nj}^{j}\right)$ is larger than the maximum data density value of the existing prototypes $\{\boldsymbol{p}\}^{j}$, $\boldsymbol{x}_{Nj}^{j}$ is more descriptive and has more summarisation power than all other prototypes. Alternatively, if $D\left(\boldsymbol{x}_{Nj}^{j}\right)$ is smaller than the minimum data density value of any prototypes, it represents an emerging pattern very different from previously seen ones. In the third case, if the distance between $\boldsymbol{x}_{Nj}^{j}$ and the nearest prototype $\boldsymbol{p}_{n*}^{j}$ is larger than γ_{G}^{j}, $\boldsymbol{x}_{Nj}^{j}$ is distant with all prototypes and represents a pattern that no existing data clouds can describe. Therefore, in either case, $\boldsymbol{x}_{Nj}^{j}$ becomes a new prototype and initialise a new data cloud.

If **Condition 4** is satisfied, $\boldsymbol{x}_{Nj}^{j}$ is recognised as a new prototype of the *jth* class, and the meta-parameters of the new data cloud associated with $\boldsymbol{x}_{Nj}^{j}$ are initialised as:

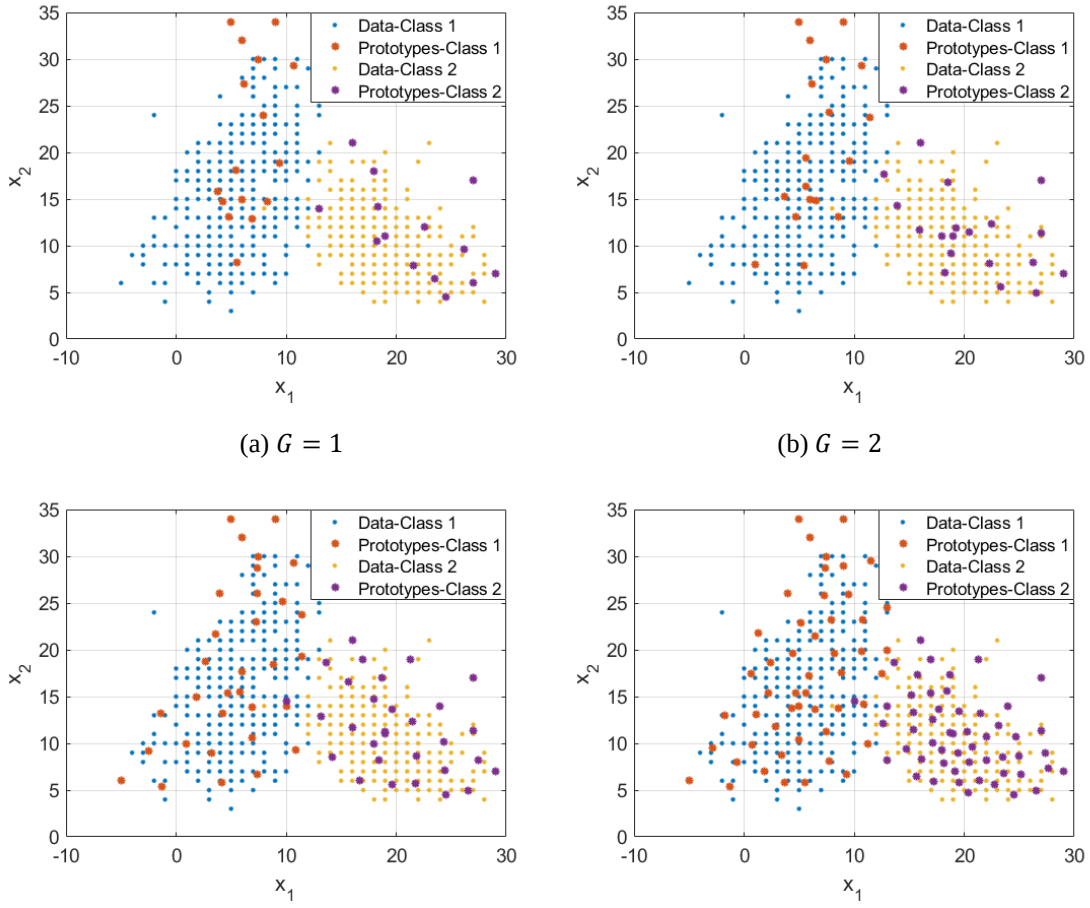
$$P^{j} \leftarrow P^{j} + 1; \quad \boldsymbol{p}_{Pj}^{j} \leftarrow \boldsymbol{x}_{Nj}^{j}; \quad \{\boldsymbol{p}\}^{j} \leftarrow \{\boldsymbol{p}\}^{j} \cup \left\{\boldsymbol{p}_{Pj}^{j}\right\}; \quad \mathbb{C}_{Pj}^{j} \leftarrow \left\{\boldsymbol{x}_{Nj}^{j}\right\}; \quad S_{Nj}^{j} \leftarrow 1 \tag{18}$$

Otherwise, the meta-parameters of the data cloud, $\mathbb{C}_{n*}^{j}$ associated with the nearest prototype are updated by $\boldsymbol{x}_{Nj}^{j}$ as:

$$\mathbb{C}_{n*}^{j} \leftarrow \mathbb{C}_{n*}^{j} \cup \left\{\boldsymbol{x}_{Nj}^{j}\right\}; \quad \boldsymbol{p}_{n*}^{j} \leftarrow \frac{S_{n*}^{j}}{S_{n*}^{j}+1}\boldsymbol{p}_{n*}^{j} + \frac{1}{S_{n*}^{j}+1}\boldsymbol{x}_{Nj}^{j}; \quad S_{n*}^{j} \leftarrow S_{n*}^{j} + 1 \tag{19}$$

After this, the fuzzy rule is updated with the newly updated $\{\boldsymbol{p}\}^{j}$, and SONFIS goes back to Stage 4 and gets ready for processing the next data sample. The online learning results of SONFIS from the remaining 30% of climate data under different levels of granularity are presented in Fig.5 for illustration.

From the offline and online identification processes described in this subsection one may conclude that, the prototypes of SONFIS are directly extracted from data and they objectively represent the local models of data distribution. It has to be stressed that a hybrid of offline and online learning processes is an important feature of SONFIS and is very useful in real-world scenarios. In most of real-world applications, a part of data has been available in a static form, while the remaining samples keep coming sequentially in a streaming form. By learning from the available static data in an offline manner, SONFIS can have a better understanding on ensemble properties and mutual distributions of data, which results in more robust and stronger performance. Then, by learning from streaming data on a sample-by-sample basis, SONFIS is capable of successfully tackling the problems with changing data pattern in nonstationary environments by continuously self-developing based on new observations.



(a) $G = 1$ (b) $G = 2$

(c) $G = 3$ (d) $G = 4$

Fig. 5. Identified prototypes from data under different levels of granularity after online learning process

The main procedures of the online and offline learning processes of SONFIS are summarised in the following pseudo codes.

1) Offline training process

Input: $\{\boldsymbol{x}\}_{N^j}^{j}$ ***Algorithm begins***
i. Calculate D^{MM} at $\{\boldsymbol{u}\}_{L^j}^{j}$ using (2); *ii.* Rank $\{\boldsymbol{u}\}_{L^j}^{j}$ into $\{\boldsymbol{r}\}$ using (4); *iii.* Identify $\{\boldsymbol{u}^*\}$ using **Condition 1**; *iv.* Form $\{\mathbb{C}\}$ around $\{\boldsymbol{u}^*\}$ using (6); *v.* Extract γ_G^j from $\{\boldsymbol{x}\}_{N^j}^{j}$ using (7)-(9); *vi.* Calculate $\{\boldsymbol{q}\}$ from $\{\mathbb{C}\}$ using (10); *vii.* Calculate D^{MM} at $\{\boldsymbol{q}\}$ using (11); *viii.* Identify $\{\boldsymbol{p}\}^j$ using **Conditions 2 and 3**; *ix.* Create $\mathbf{R}^j$ based on $\{\boldsymbol{p}\}^j$.
Algorithm ends ***Output:*** $\mathbf{R}^j$

2) Online learning process

Input: $\left\{\boldsymbol{x}^j_{N^j+1}, \boldsymbol{x}^j_{N^j+2}, \boldsymbol{x}^j_{N^j+3}, \ldots\right\}$ ***Algorithm begins***
While $\boldsymbol{x}^j_{N^j+1}$ is available or (until interrupted): *i.* Update $\boldsymbol{\mu}^j_{N^j}$ and $X^j_{N^j}$ by (14); *ii.* Update γ^j_G by (15); *iii.* Calculate D at $\boldsymbol{x}^j_{N^j+1}$ and $\{\boldsymbol{p}\}^j$ using (16); *iv.* **If** (**Condition 4** is met) **Then:** - Add $\boldsymbol{x}^j_{N^j}$ as a new prototype by (18); *v.* **Else:** - Update $\mathbb{C}^j_{n*}$ by (19); *vi.* **End If** *vii.* $N^j \leftarrow N^j + 1$; *viii.* Update $\mathbf{R}^j$; **End While**
Algorithm ends ***Output:*** $\mathbf{R}^j$

However, one may also notice that SONFIS as well as the majority of alterative zero-order EISs identify prototypes in a non-iterative and straightforward manner. There is no optimisation process involved because zero-order EISs need to be computationally lean. Therefore, the optimality of zero-order EISs require to be studied to better understand their merits and limitations. In the next section, we will present the optimality analysis.

3.3. Validation Process

In this subsection, the decision-making procedure of SONFIS is presented. As one can see from Fig. 1, during the validation process, there is a two-level decision-making process involved for deciding the label of each unlabelled data sample. This includes the local decision-making and overall decision-making processes. The local decision-maker follows the "nearest prototype" principle, and the global decision-maker follows the "winner takes all" principle.

For each newly arrived data sample, $\boldsymbol{x}_K$, it is firstly sent to the C massively parallel fuzzy rules, and the local decision-maker will identify the most similar prototype to $\boldsymbol{x}_K$ within this fuzzy rule and calculate the score of confidence using equation (20):

$$\lambda^j(\boldsymbol{x}_K) = \max_{i=1,2,\ldots,P^j}\left(e^{-\left\|\boldsymbol{x}_K - \boldsymbol{p}^j_i\right\|^2}\right) \tag{20}$$

The score of confidence produced by a particular massively parallel fuzzy rule is determined by similarity between $\boldsymbol{x}_K$ and the nearest prototype of the corresponding class. The exponential function is used for confining the value of score of confidence into a more familiar range, namely, $[0,1]$ and, at the same time, enlarging the Euclidean distance between $\boldsymbol{x}_K$ and prototypes $\{\boldsymbol{p}\}^j$. Nonetheless, it has to be stressed that the exponential function used in equation (20) can be replaced by other functions as well.

Then, the C scores of confidence generated by the C fuzzy rules will be passed to the global decision-maker, and the label of $\boldsymbol{x}_K$ will be decided as:

$$class\ label = \underset{j=1,2,\ldots,C}{\operatorname{argmax}} \left(\lambda^j(\boldsymbol{x}_K)\right) \tag{21}$$

In the next section, we will study the optimality of the premise, IF part by SONFIS from the data partitioning point of view.

4. Analysis of the Optimality

The analysis of optimality of the initial solutions obtained from data by SONFIS [18] is performed in this section. As SONFIS identifies prototypes per class, we only consider the optimality of the solution of a particular class (assuming the *jth* one; $j = 1,2,\ldots,C$). The optimality analysis can be applied to all other classes as well. It is necessary to stress that the analysis and general principles presented in this section can also be applied to other online non-iterative learning algorithms with similar operating mechanisms.

4.1. Mathematical Formulation of the Problem

Thanks to the non-parametric nature of both, the consequent, THEN part and the premise, IF part (which is prototype-based), the optimality of SONFIS depends solely on the optimal positions of prototypes, namely, the most representative data samples in the data space. Therefore, the optimality problem of SONFIS is reduced to finding a locally optimal data partition solution. From machine learning point of view, this can be considered as locally optimal clustering. The formal mathematical condition for this can be described in the form of a mathematical programming problem [38] as follows.

Considering that SONFIS partitions data samples of *jth* class, $\{\boldsymbol{x}\}_{N^j}^j = \left\{\boldsymbol{x}_1^j, \boldsymbol{x}_2^j, \ldots, \boldsymbol{x}_{N^j}^j\right\}$, into P^j data clouds, we formulate the optimality problem in the form of the following mathematical programming problem for clustering/data partitioning [38]:

$$\boldsymbol{Problem\ 1}\text{: } f(\boldsymbol{W}^j, \boldsymbol{P}^j) = \sum_{k=1}^{P^j} \sum_{i=1}^{N^j} w_{i,k}^j\, d\left(\boldsymbol{x}_i^j, \boldsymbol{p}_k^j\right) \tag{22}$$

where $j = 1,2,\ldots,C$; $\boldsymbol{W}^j = \left[w_{i,k}^j\right]$ is a $P^j \times N^j$ real matrix; $\boldsymbol{P}^j = \left[\boldsymbol{p}_1^j, \boldsymbol{p}_2^j, \ldots, \boldsymbol{p}_{P^j}^j\right] \in \mathbf{R}^{M \times P^j}$.

$\boldsymbol{W}^j$ is subject to the following constraints ($k = 1,2,\ldots,P^j$, $i = 1,2,\ldots,N^j$):

$$w_{i,k}^j \in \{0,1\} \tag{23a}$$

$$\sum_{k=1}^{P^j} w_{i,k}^j = 1 \tag{23b}$$

and the collection of all $\boldsymbol{W}^j$ that meet the constraints (equation (23)) is denoted as: $\mathbf{N}^{P^j \times N^j}$. The reason for imposing such constraints on $\boldsymbol{W}^j$ is that each data sample can be associated with only one prototype according to equation (6).

Problem 1 is a nonconvex problem, and therefore, the local minimum point does not need to be a global minimum [38]. The necessary conditions for global optimality of **Problem 1** is called Karush–Kuhn–Tucker, which can be found in [25].

It has been proven in [28] that when square Euclidean distance is used, partially optimal solutions are always locally optimal as shown in **Theorem 1**. The detailed proof of **Theorem 1** can be found on page 5 of [38].

Theorem 1: Consider **Problem 1** where $d\left(\boldsymbol{x}_i^j, \boldsymbol{p}_k^j\right) = \left\|\boldsymbol{x}_i^j - \boldsymbol{p}_k^j\right\|^2$, a partially optimal solution of **Problem 1** is a local minimum point.

From **Theorem 1** one can see that, a locally optimal solution of **Problem 1** is equivalent to a partially optimal solution. The definition of a partially optimal solution is as follows [43].

***Definition* 1:** a point $(\boldsymbol{W}^{j*}, \boldsymbol{P}^{j*})$ is a partially optimal solution for **Theorem 1** if the following two inequalities are satisfied

$$\begin{aligned} &f(\boldsymbol{W}^{j*}, \boldsymbol{P}^{j*}) \leq f(\boldsymbol{W}^{j}, \boldsymbol{P}^{j*}) \quad for\ all\ \boldsymbol{W}^{j} \in \mathbf{N}^{P^j \times N^j} \\ &f(\boldsymbol{W}^{j*}, \boldsymbol{P}^{j*}) \leq f(\boldsymbol{W}^{j*}, \boldsymbol{P}^{j}) \quad for\ all\ \boldsymbol{P}^{j} \in \mathbf{R}^{M \times P^j} \end{aligned} \tag{24}$$

By solving the following two problems, one can get a partially optimal solution [43]:

***Problem* 2:** Given $\widehat{\boldsymbol{P}}^{j} \in \mathbf{R}^{M \times P^j}$, minimize $f\left(\boldsymbol{W}^{j}, \widehat{\boldsymbol{P}}^{j}\right)$ subject to $\boldsymbol{W}^{j} \in \mathbf{N}^{P^j \times N^j}$.

***Problem* 3:** Given $\widehat{\boldsymbol{W}}^{j} \in \mathbf{N}^{P^j \times N^j}$, minimize $f\left(\widehat{\boldsymbol{W}}^{j}, \boldsymbol{P}^{j}\right)$ subject to $\boldsymbol{P}^{j} \in \mathbf{R}^{M \times P^j}$.

In other words, $(\boldsymbol{W}^{j*}, \boldsymbol{P}^{j*})$ is a partially optimal solution on condition that $\boldsymbol{W}^{j*}$ solves **Problem 2** with $\widehat{\boldsymbol{P}}^{j} = \boldsymbol{P}^{j*}$ and $\boldsymbol{P}^{j*}$ solves **Problem 3** with $\widehat{\boldsymbol{W}}^{j} = \boldsymbol{W}^{j*}$.

4.2. Local Optimality Analysis of Data Partitioning by SONFIS

From subsection 4.1 one can see that the data partitioning result $(\{\boldsymbol{p}\}^j$ and $\{\mathbb{C}\}^j)$ is locally optimal if $(\boldsymbol{W}^{j}, \boldsymbol{P}^{j})$ is a partially optimal solution of **Problem 1**. As it is described in subsection 3.2, SONFIS self-organises its multi-model architecture from static data and, then, continues to self-develop with streaming data on a sample-by-sample basis. In this subsection, we will analyse the local optimality of the solutions achieved by the offline and online learning processes, separately.

Firstly, let us consider the offline learning process. As one can see from subsection 3.2, after $\{\boldsymbol{p}\}^j$ are filtered out from raw prototypes at the end of the process, the static data, $\{\boldsymbol{x}\}_{N^j}^{j}$ is partitioned into P^j data clouds, $\{\mathbb{C}\}^j = \left\{\mathbb{C}_1^j, \mathbb{C}_2^j, \dots, \mathbb{C}_{P^j}^j\right\}$ by using equation (6) to form Voronoi tessellations around $\{\boldsymbol{p}\}^j$. Based on $\{\boldsymbol{p}\}^j$ and $\{\mathbb{C}\}^j$, one can obtain the prototype matrix $\boldsymbol{P}^j$ and the weight matrix $\boldsymbol{W}^j$, respectively, and **Problem 1** can be reformulated as:

$$f(\boldsymbol{W}^{j}, \boldsymbol{P}^{j}) = \sum_{k=1}^{P^j} \sum_{i=1}^{N^j} w_{i,k}^{j}\, d\left(\boldsymbol{x}_i^j, \boldsymbol{p}_k^j\right) = \sum_{k=1}^{P^j} \sum_{\boldsymbol{x} \in \mathbb{C}_k^j} \left\|\boldsymbol{x} - \boldsymbol{p}_k^j\right\|^2 \tag{25}$$

By taking equation (6) into consideration, one can conclude that $\boldsymbol{W}^j$ is the minimum solution of **Problem 2** given $\boldsymbol{P}^j$. Thus, the data partitioning result, $\{\mathbb{C}\}^j$ is an optimal solution of **Problem 2**. However, there is no guarantee that the solution $(\boldsymbol{W}^{j}, \boldsymbol{P}^{j})$ can solve **Problem 3** as well because this problem is not taken into consideration during the entire offline learning process.

Considering the online learning process, which is of "one pass" type and non-iterative, there is also no guarantee that the current solution obtained by $(\boldsymbol{W}^{j}, \boldsymbol{P}^{j})$ can solve **Problems 2** and **3** at the same time. In fact, for a particular data sample, $\boldsymbol{x}_i^j$ $(i = 1,2, \dots, N^j - 1)$, only the following equation is guaranteed to be valid at the time instance when $\boldsymbol{x}_i^j$ was observed:

$$\sum_{k=1}^{P^j(i)} w_{i,k}^{j} d\left(\boldsymbol{x}_i^j, \boldsymbol{p}_k^j(i)\right) = \min_{k=1,2,\dots,P^j(i)} \left(\left\|\boldsymbol{x}_i^j - \boldsymbol{p}_k^j(i)\right\|^2\right) = \left\|\boldsymbol{x}_i^j - \boldsymbol{p}_{n*}^j(i)\right\|^2 \tag{26}$$

where $\boldsymbol{p}_k^j(i)(k = 1,2, \dots, P^j(i))$ are the existing prototypes at the time instance that $\boldsymbol{x}_i^j$ is observed; $P^j(i)$ is the corresponding number of prototype at the *ith* time instance; $\boldsymbol{p}_{n*}^j(i)$ is the nearest prototype to $\boldsymbol{x}_i^j$, which can be $\boldsymbol{x}_i^j$ itself if **Condition 4** is met. Otherwise, there are $w_{i,n*}^j = 1$ and $w_{i,k}^j = 0$ for $\forall k \neq n^*$.

From the time instance at which the next data sample of the *jth* class is observed, equation (26) is not guaranteed to be true anymore for $\boldsymbol{x}_i^j$ due to the possible shift of $\boldsymbol{p}_{n*}^j$ and/or the initialisation of new data clouds with prototypes closer to $\boldsymbol{x}_i^j$ than $\boldsymbol{p}_{n*}^j$. However, the values of $w_{i,k}^j$ $(k = 1,2, \dots P^j)$ has been already fixed when $\boldsymbol{x}_i^j$ was observed because $\boldsymbol{x}_i^j$ has been assigned as a member of $\mathbb{C}_{n*}^j$ and there is no reallocation in the future. As a result, we can state that the following inequality applies to all historically observed data samples, $\boldsymbol{x}_i^j$ $(i = 1,2, \dots, N^j - 1)$:

$$\sum_{k=1}^{P^j(N^j)} w_{N^j,k}^j d\left(\boldsymbol{x}_i^j, \boldsymbol{p}_k^j(N^j)\right) = \left\|\boldsymbol{x}_i^j - \boldsymbol{p}_{n*}^j(N^j)\right\|^2 \geq \min_{k=1,2,\ldots,P^j(N^j)} \left(\left\|\boldsymbol{x}_i^j - \boldsymbol{p}_k^j(N^j)\right\|^2\right) \quad (27)$$

which means that $(\boldsymbol{W}^j, \boldsymbol{P}^j)$ is not a minimum solution of **Problem 3** given $\boldsymbol{W}^j$. Therefore, one can conclude that the data partitioning result obtained by SONFIS is not locally optimal. In the next section, we will discuss a feasible approach for SONFIS to attain local optimality from the initial prototype solutions.

5. Attaining Locally Optimal Solution

As we have proven that the premise, IF part of SONFIS lacks local optimality, in this section, we will discuss how to optimise the positions of prototypes of SONFIS in the data space to attain the locally optimal solution. According to **Theorem 1**, in order to find a locally optimal solution for the premise, IF part of SONFIS, one can look for a partially optimal solution in the problem space instead. One possible way to obtain a partially optimal IF part from the initial partitioning result by SONFIS is to further apply an iterative optimisation process, for example, using the similar iterative process used by the well-known K-means clustering algorithm [38].

In this section, the proposed prototype optimisation (PO) algorithm for SONFIS is presented. This algorithm concerns only the identified prototypes and the historical data; thus, it is entirely data-driven and nonparametric. The main algorithmic procedure is composed of the following four steps. However, it has to be stressed that the optimisation process is performed on prototypes of each class separately, which means different massively parallel rules in the rule base can be optimised at the same time in parallel. In this section, we use prototypes of the *jth* class as an example.

Step 1. Re-denote the identified prototype matrix, $\boldsymbol{P}^j$ from the offline learning process and/or online learning process as $\boldsymbol{P}^j(t)$ ($t = 0$, which indicates the current number of iterations) and solve **Problem 2** by setting $\boldsymbol{P}^j(t)$ to $\widehat{\boldsymbol{P}}^j$ and obtain $\boldsymbol{W}^j(t) = \left[w_{i,k}^j(t)\right]$ as the optimal solution.

The solution of **Problem 2** can be expressed as follows ($i = 1,2,\ldots,N^j$):

$$\begin{cases} w_{i,k}^j(t) = 1 & k = \underset{l=1,2,\ldots,P^j}{\operatorname{argmin}} \left(\left\|\boldsymbol{x}_i^j - \widehat{\boldsymbol{p}}_l^j\right\|^2\right) \\ w_{i,k}^j(t) = 0 & k \in else \end{cases} \quad (28)$$

Step 2. Solve **Problem 3** by setting $\boldsymbol{W}^j(t)$ as $\widehat{\boldsymbol{W}}^j$ and identify new prototypes denoted by $\boldsymbol{P}^j(t+1)$.

The solution of **Problem 3** may not be as obvious as **Problem 2**. With the given $\widehat{\boldsymbol{W}}^j$, it is obvious that **Problem 3** is equivalent to the problem of finding $\boldsymbol{P}^j(t+1) \in \mathbf{R}^{M \times P^j}$, which satisfies the following equation:

$$f_1\left(\boldsymbol{P}^j(t+1)\right) = \min_{\boldsymbol{Z} \in \mathbf{R}^{M \times P^j}} \left(f_1(\boldsymbol{Z})\right) \quad (29)$$

where $f_1(\boldsymbol{Z}) = f\left(\widehat{\boldsymbol{W}}^j, \boldsymbol{Z}\right)$. $f_1(\boldsymbol{Z})$ can be reformulated as:

$$f_1(\boldsymbol{Z}) = \sum_{k=1}^{P^j} \sum_{\forall i \in \left\{l | w_{l,k}^j(t)=1, l=1,2,\ldots,N^j\right\}} \left\|\boldsymbol{x}_i^j - \boldsymbol{z}_k\right\|^2 = f_2(\boldsymbol{z}_1) + f_2(\boldsymbol{z}_2) + \cdots + f_2\left(\boldsymbol{z}_{Pj}\right) \quad (30)$$

where $f_2(\boldsymbol{z}_k) = \sum_{\forall i \in \left\{l | w_{l,k}^j(t)=1, l=1,2,\ldots,N^j\right\}} \left\|\boldsymbol{x}_i^j - \boldsymbol{z}_k\right\|^2$, $k = 1,2,\ldots,P^j$.

The problem of minimising $f_1(\boldsymbol{Z})$ (equation (29)) can be further simplified to the problem of finding $\boldsymbol{p}_k^j(t+1) \in \mathbf{R}^M$ ($k = 1,2,\ldots,P^j$) that meets the following equation:

$$\boldsymbol{p}_k^j(t+1) = \underset{\boldsymbol{z}_k \in \mathbf{R}^M}{\operatorname{argmin}} \left(f_2(\boldsymbol{z}_k)\right) \quad (31)$$

Because $\left\|\boldsymbol{x}_i^j - \boldsymbol{z}_k\right\|^2 = \sum_{h=1}^{M} \left(x_{i,h}^j - z_{k,h}\right)^2$ ($i \in \left\{l | w_{l,k}^j(t) = 1, l = 1,2,\ldots,N^j\right\}$), $f_2(\boldsymbol{z}_k)$ is a convex function and is differentiable for $\boldsymbol{z}_k \in \mathbf{R}^M$. Therefore, $\boldsymbol{p}_k^j(t+1)$ is the minimum value of $f_2(\boldsymbol{z}_k)$.

According to Fermat's Theorem, the partial derivative of $f_2(\boldsymbol{z}_k)$ at each dimension will have a value of 0 at $\boldsymbol{p}_k^j(t+1) = \left[p_{k,1}^j(t+1), p_{k,2}^j(t+1), \dots, p_{k,M}^j(t+1)\right]^T$, namely:

$$\frac{\partial f_2(\boldsymbol{z}_k)}{\partial z_{k,h}} = 2\sum_{\forall i \in \left\{l | w_{l,k}^j(t)=1, l=1,2,\dots,N^j\right\}} \left(x_{i,h}^j - z_{k,h}\right) = 0 \quad iff\ \boldsymbol{z}_k = \boldsymbol{p}_k^j(t+1) \tag{32}$$

where $h = 1,2,\dots,M$, $k = 1,2,\dots,P_i$. Equation (32) can be further simplified in the following form with $\boldsymbol{z}_k = \boldsymbol{p}_k^j(t+1)$:

$$\begin{aligned} &\sum_{\forall i \in \left\{l | w_{l,k}^j(t)=1, l=1,2,\dots,N^j\right\}} \left(x_{i,h}^j - p_{k,h}^j(t+1)\right) = \\ &\sum_{\forall i \in \left\{l | w_{l,k}^j(t)=1, l=1,2,\dots,N^j\right\}} x_{i,h}^j - \sum_{l=1}^{N^j} w_{l,k}^j(t) \cdot p_{k,h}^j(t+1) = 0 \end{aligned} \tag{33}$$

Based on equation (33) one can see that, $\boldsymbol{p}_k^j(t+1)$ is the mean of data samples, $\boldsymbol{x}_i^j$ ($\forall i \in \left\{l | w_{l,k}^j(t) = 1, l = 1,2,\dots,N^j\right\}$) that are associated with $\boldsymbol{p}_k^j(t)$, namely ($k = 1,2,\dots,P^j$):

$$\boldsymbol{p}_k^j(t+1) = \frac{1}{\sum_{l=1}^{N^j} w_{l,k}^j(t)} \sum_{\forall i \in \left\{l | w_{l,k}^j(t)=1, l=1,2,\dots,N^j\right\}} \boldsymbol{x}_i^j \tag{34}$$

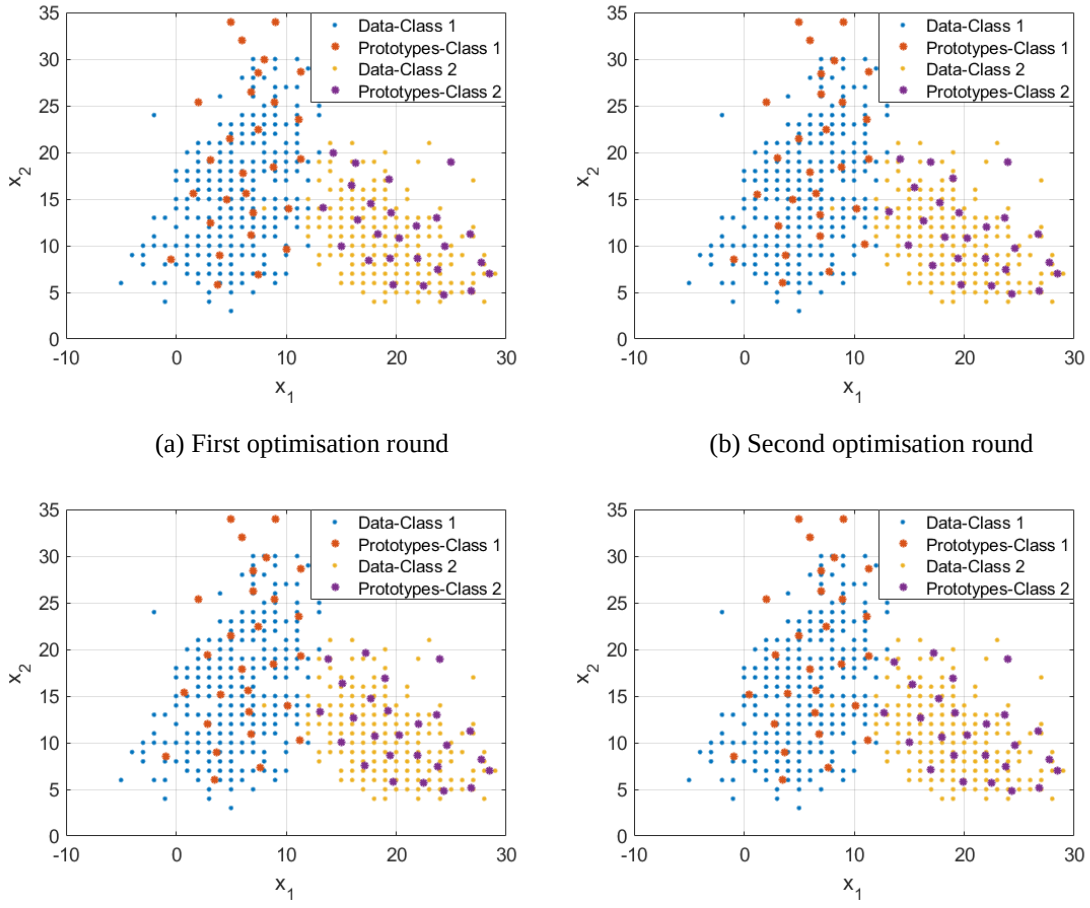
Step 3. Solve **Problem 2** by setting $\boldsymbol{P}^j(t+1)$ as $\widehat{\boldsymbol{P}}^j$ and obtain $\boldsymbol{W}^j(t+1) = \left[w_{i,k}^j(t+1)\right]$ as the optimal solution.

Step 4. If $f\left(\boldsymbol{W}^j(t+1), \boldsymbol{P}^j(t+1)\right) = f\left(\boldsymbol{W}^j(t), \boldsymbol{P}^j(t)\right)$, the optimum solution is reached, the optimisation algorithm stops and $\left(\boldsymbol{W}^j(t+1), \boldsymbol{P}^j(t+1)\right)$ is set to be $(\boldsymbol{W}^{j*}, \boldsymbol{P}^{j*})$; Otherwise, $t \leftarrow t+1$ and go back to Step 2.

To better illustrate the proposed concept and principles, we use the PO algorithm for optimising the prototype identification results obtained by SONFIS and present the results in the following figure, where the obtained solution with $G = 3$ as given in Fig. 5(a) is used for visual clarity. As we can see from Fig. 6, after a few iterations, all prototypes have reached their best positions in the data space.

The main procedure of the PO algorithm is also summarised by the following pseudo code:

Input: $\{\boldsymbol{x}\}_{N^j}^j$ and $\{\boldsymbol{p}\}^j$ ***Algorithm begins***
i. $t \leftarrow 0$; *ii.* Obtain $\boldsymbol{W}^j(t)$ by solving **Problem 2** with $\boldsymbol{P}^j(t)$; *iii.* Calculate $f\left(\boldsymbol{W}^j(t), \boldsymbol{P}^j(t)\right)$ by (25); *iv.* **While** $f\left(\boldsymbol{W}^j(t), \boldsymbol{P}^j(t)\right) \neq f\left(\boldsymbol{W}^j(t-1), \boldsymbol{P}^j(t-1)\right)$**:** - Obtain $\boldsymbol{P}^j(t+1)$ by solving **Problem 3** with $\boldsymbol{W}^j(t)$; - Obtain $\boldsymbol{W}^j(t+1)$ by solving **Problem 2** with $\boldsymbol{P}^j(t+1)$; - Calculate $f\left(\boldsymbol{W}^j(t+1), \boldsymbol{P}^j(t+1)\right)$ by (25); - $t \leftarrow t+1$; *v.* **End While**
Algorithm ends ***Output:*** $\{\boldsymbol{p}\}^{j*}$



(a) First optimisation round

(b) Second optimisation round

(c) Third optimisation round

(d) Final optimisation round

Fig. 6. Illustration of the optimisation process

The proposed PO algorithm guarantees a partially optimal solution of **Problem 1** as stated in **Theorem 2** as follows.

***Theorem* 2:** The PO algorithm converges to a partially optimal solution of **Problem 1** in a finite number of iterations.

The detailed proof of **Theorem 2** can be found on page 3 of [38].

The proposed PO algorithm can be applied to SONFIS during the system identification process in a way as depicted in Fig. 7. After all prototypes of SONFIS have been identified at the end of the learning process, the PO algorithm is used to help SONFIS achieve a locally optimal partitioning by refining the positions of the prototypes in the data space. This effectively updates/fine-tunes the premise, IF part of the fuzzy rules. By involving the proposed PO algorithm, the local optimality of partitioning results can be guaranteed at the price of lower computational efficiency because of the iterative optimisation process. Nonetheless, it is worth to be noticed that in practice, only a few iterations are needed.

By using the framework depicted in Fig. 7, the offline and online learning processes of SONFIS are still highly efficient, but all historical data samples are required to be kept in system memory. Additional computational resources will be consumed at the end of the learning process only to perform the optimisation process. On the other hand, one may consider alternative ways to use the proposed PO algorithm, for example, conducting optimisation every time the system is updated. However, the aim of this paper is to deliver the general concept and principles, and thus, we only consider the implementation presented in Fig. 7 without loss of generality. It is also necessary to stress that proposed PO algorithm is generic and applicable for other types of evolving learning algorithms with similar operating mechanisms, and this will be demonstrated through

numerical examples presented in section 7. Alteratively, one may also consider using generic optimisation algorithms to optimise the solution obtained by SONFIS, e.g. particle swarm optimisation (PSO) algorithm [9]. However, the attractiveness of the PO algorithm comes from the two aspects. Firstly, the proposed algorithm is a modified form of the most widely used K-means algorithm, it is designed specifically for optimising data partitioning problems and its effectiveness and validity are guaranteed. Secondly, the PO algorithm is computationally efficient and only adds minor additional computational cost to SONFIS. In the next section, a detailed computational complexity analysis on SONFIS and the PO algorithm will be provided as the supporting evidence.

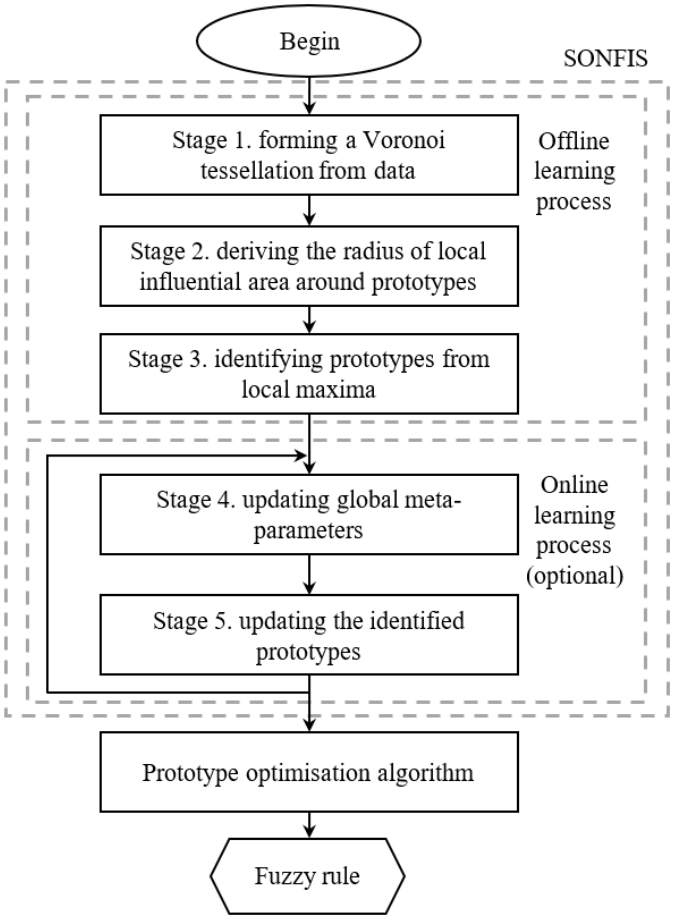


Fig. 7 The diagram of optimising SONFIS using the proposed algorithm

6. Computational Complexity Analysis

In this section, we will analyse the computational complexity of SONFIS and the proposed PO algorithm.

During the first stage of the offline learning process of SONFIS, the computational complexity of calculating the multimodal density values at the observed unique data samples of the *jth* class is $O(L^j M)$. The computational complexity for forming Voronoi tessellations from data is $O(L^{j*} N^j M)$. During stage 2, the computational complexity for estimating the radius of influential area around each prototype is $O((N^j)^2 M)$. In the third stage, the computational complexity for calculating the multimodal density value at each raw prototype is $O(L^{j*} M)$, the computational complexity for the highly representative prototype identification is negligible, and the complexity for forming data clouds around the identified prototypes is $O(P^j N^j M)$. Therefore, the overall computational complexity of the offline learning process of SONFIS is $O\left(M \sum_{j=1}^{C} (N^j)^2\right)$

During the online learning process, SONFIS will update one of its fuzzy rules in the rule base for each newly arrived data sample. Assuming the new data sample belongs to the *jth* class, the computational complexity of stage 4 is $O(M)$. The computational complexity of stage 5 is $O\left((P^j+1)M\right)$, which is mainly caused by the calculation of data density values at the new data sample and the previously identified prototypes. Therefore, the overall computational complexity of updating SONFIS with each new data sample during the online learning process is $O(MP^j)$.

For the proposed PO algorithm, the overall computational complexity is hard to estimate because prototypes of difference classes require different numbers of iterations to reach the optimal positions. However, we still can estimate that, for prototypes of the *jth* class ($j = 1,2,\dots,C$), the computational complexity of each iteration is $O(P^jN^jM)$. Thus, the overall computational complexity of the optimisation process using the PO algorithm is $O\left(M\sum_{j=1}^{C}P^jN^jT^j\right)$, where T^j is the number of iteration steps for prototypes of the *jth* class to converge.

7. Numerical Examples and Discussions

In this section, numerical examples are presented for validating the proposed concept and general principles. The experiments are conducted using MATLAB R2018a on a PC with dual core processor 3.60 GHz×2 and 16 GB RAM. As it was stated in section 5, we only apply the proposed PO algorithm at the end of the learning process of SONFIS to optimise the obtained solutions, namely, prototypes.

7.1. Experiments on Benchmark Numerical Datasets

In this subsection, the validity and effectiveness of the proposed PO algorithm are demonstrated through numerical examples on a number of benchmark datasets. The following nine real-world challenging problems are considered, and details of these datasets are tabulated in Table 2:

1) Wilt (WI) dataset[2];

2) Occupancy detection (OD) dataset[3];

3) Optical recognition of handwritten digits (OR) dataset[4];

4) Pen-based recognition of handwritten digits (PR) dataset[5];

5) Multiple features (MF) dataset[6];

6) Epileptic seizure recognition (ES) dataset[7];

7) Letter recognition (LR) dataset[8];

8) Crowdsourced Mapping (CM) dataset[9], and;

9) Forest cover type (FC) dataset[10].

By default, SONFIS uses Euclidean distance in the experiments conducted in this section. In this paper, the time stamps of the OD dataset are removed in advance, and the two testing sets are combined into one for numerical experiments. Binary classification is performed on the ES dataset to distinguish the subjects with and without epileptic seizure, namely, class 1 versus the rest.

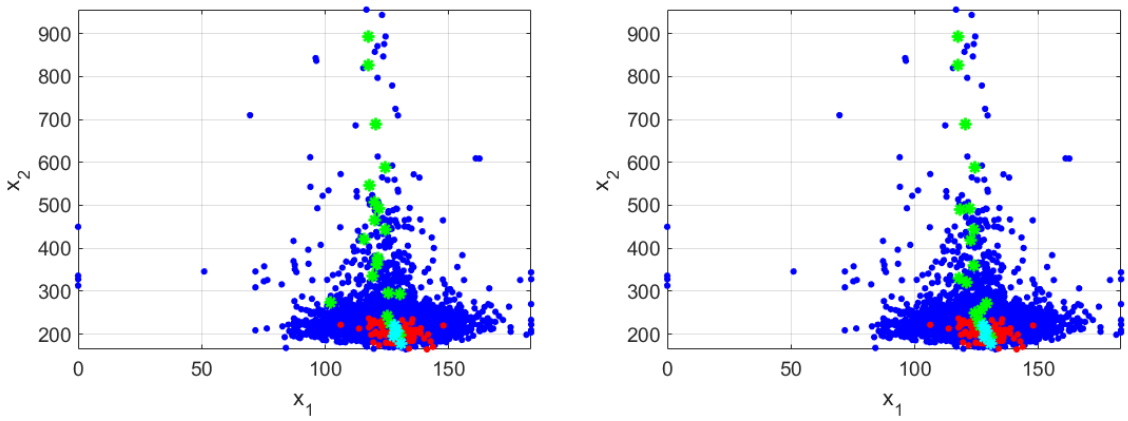
[2] Available from http://archive.ics.uci.edu/ml/datasets/wilt
[3] Available from https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+
[4] Available from https://archive.ics.uci.edu/ml/datasets/optical+recognition+of+handwritten+digits
[5] Available from https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits
[6] Available from https://archive.ics.uci.edu/ml/datasets/Multiple+Features
[7] Available from https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition
[8] Available from https://archive.ics.uci.edu/ml/datasets/letter+recognition
[9] Available from https://archive.ics.uci.edu/ml/datasets/Crowdsourced+Mapping
[10] Available from https://archive.ics.uci.edu/ml/datasets/covertype

As the structure of WI and OD datasets are both relatively simpler compared with other benchmark datasets considered within this subsection, we use the two datasets for demonstrating the concept of the proposed approach. The offline data partitioning results obtained by SONFIS on the training sets of the two benchmark datasets are depicted in Fig. 8, where dots "·" in red and blue represent data samples from classes 1 and 2, respectively; asterisks "*" in cyan and green represent prototypes identified from the data samples of the corresponding classes. By using the proposed PO algorithm, we obtain the local optimal data partitioning, and the results obtained by the locally optimal SONFIS (LO-SONFIS) are also given in Fig. 8 for comparison, where only the first two attributes of the datasets are presented for visual clarity. The value changes of $f(\boldsymbol{W}^j, \boldsymbol{P}^j)$ $(j = 1,2)$ after each iteration are given in Fig. 9, where the values have been normalised to the range $[0,1]$ for better visualisation. In this example, the level of granularity of SONFIS, G is set to be $G = 2$ because it enables SONFIS to identify a smaller number of prototypes and is more suitable for illustration. From Figs. 8 and 9 one can see that the PO algorithm is capable of assisting SONFIS to achieve the locally optimal solutions after a few iterations.

Table 2. Details of datasets

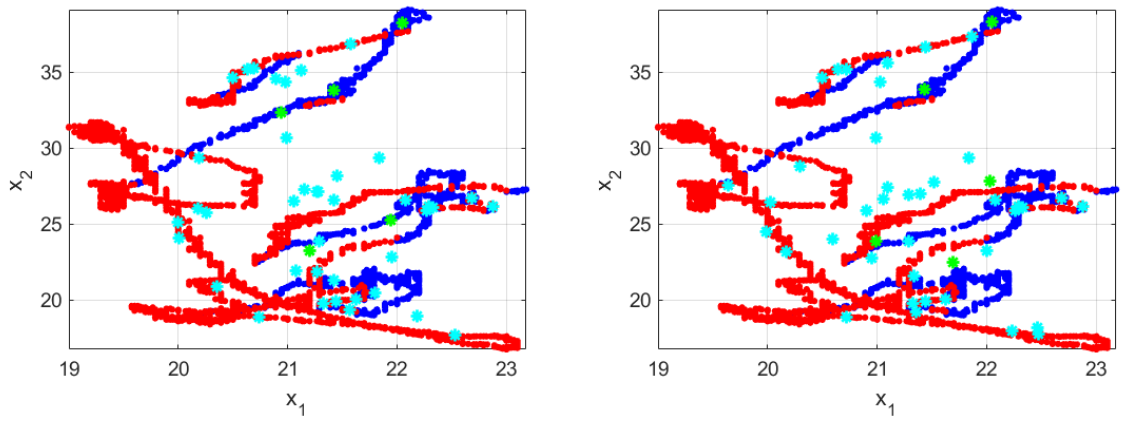
Datasets		Number of Classes	Number of Samples	Number of Attributes
WI	Training set	2	4339	5+1 label
	Testing set		500	
OD	Training set	2	8143	5+1 label
	Testing set 1		2664	
	Testing set 2		9752	
OR	Training set	10	3823	64+1 label
	Testing set		1797	
PR	Training set	10	7494	16+1 label
	Testing set		3498	
MF		10	2000	649+1 label
ES		2	11500	178+1 label
LR		26	20000	16+1 label
CM	Training set[a]	16	10545	28+1 label
	Testing set		300	
FC		7	581012	54+1 label

[a] containing many class labelling errors.



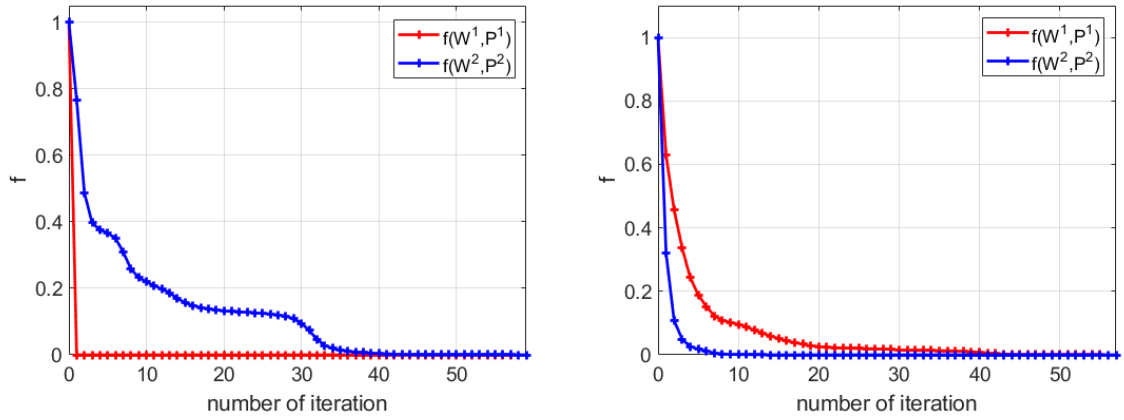
(a) Initial data partitioning-WI

(b) Local optimal data partitioning-WI



(c) Initial data partitioning-OD (d) Local optimal data partitioning-OD

Fig. 8. Data partitioning results.



(a) WI (b) OD

Fig. 9. The changes of the values of $f(\boldsymbol{W}^j, \boldsymbol{P}^j)$ $(j = 1,2)$ after each iteration

In the following numerical example, we will evaluate the effectiveness of the proposed PO algorithm by comparing the classification performance between SONFIS and LO-SONFIS. Here, WI, OD, OR and PR datasets are used for experiments. The training sets are used for priming SONFIS in an offline scenario and the performance of LO-SONFIS and SONFIS in terms of classification accuracy (*ACC*) on testing sets and execution time (t_{exe} in sec) are tabulated in Table 3. The level of granularity of SONFIS is set to be $G = 1,2,3,4,5,6$. The reported numerical results are the average of 10 Monte Carlo experiments by randomly scrambling the order of the training samples. The total number of prototypes (*NP*) identified by SONFIS is given in Fig. 10(a) to profile the system complexity.

Table 3. Performance comparison between SONFIS and LO-SONFIS – scenario 1

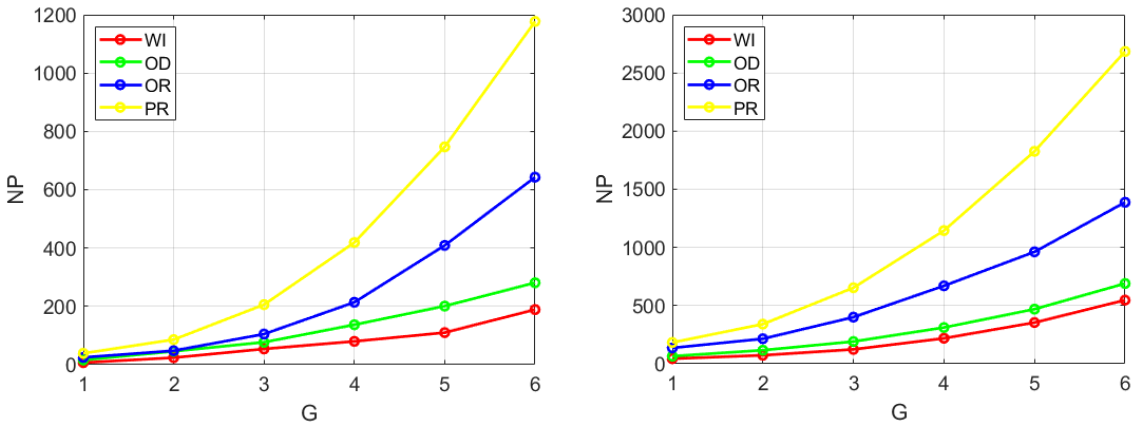
Dataset	Algorithm	Measures	Granularity, G					
			1	2	3	4	5	6
WI	SONFIS	*ACC*	0.4960	0.6460	0.6900	0.8060	0.7960	0.8100
		t_{exe}	1.04	1.07	1.09	1.12	1.12	1.13
	LO-SONFIS	*ACC*	0.5420	0.7080	0.7820	0.8100	0.8040	0.8000
		t_{exe}	1.07	1.16	1.38	1.54	1.57	1.78
OD	SONFIS	*ACC*	0.8107	0.8403	0.8618	0.9112	0.9382	0.9513
		t_{exe}	2.27	2.40	2.55	2.48	2.51	2.57

	LO-SONFIS	*ACC*	0.8150	0.8405	0.8770	0.9165	0.9410	0.9524
		t_{exe}	2.30	2.63	3.07	3.59	2.95	3.25
OR	SONFIS	*ACC*	0.9160	0.9421	0.9499	0.9716	0.9766	0.9761
		t_{exe}	0.09	0.09	0.09	0.09	0.09	0.09
	LO-SONFIS	*ACC*	0.9254	0.9505	0.9549	0.9755	0.9777	0.9777
		t_{exe}	0.12	0.12	0.12	0.15	0.15	0.15
PR	SONFIS	*ACC*	0.9028	0.9503	0.9588	0.9700	0.9743	0.9780
		t_{exe}	0.28	0.29	0.29	0.29	0.29	0.31
	LO-SONFIS	*ACC*	0.9108	0.9528	0.9663	0.9720	0.9746	0.9771
		t_{exe}	0.34	0.35	0.37	0.40	0.40	0.42

In the following numerical example, we repeat the same experiments as reported in Table 3 under the same protocol. However, in this case, one third of the training samples are used for priming SONFIS offline first, and the remaining samples are treated as data streams and used for training SONFIS on a sample-by-sample basis. The experimental results are given in Table 4, and the total number of prototypes (*NP*) identified by SONFIS during experiments is given in Fig. 10(b).

Table 4. Performance comparison between SONFIS and LO-SONFIS – scenario 2

Dataset	Algorithm	Measures	Granularity, *G*					
			1	2	3	4	5	6
WI	SONFIS	*ACC*	0.6410	0.6842	0.7328	0.7964	0.8138	0.8222
		t_{exe}	0.46	0.43	0.46	0.49	0.53	0.52
	LO-SONFIS	*ACC*	0.7456	0.7708	0.7696	0.8016	0.8218	0.8268
		t_{exe}	0.67	0.70	0.83	0.94	1.11	1.24
OD	SONFIS	*ACC*	0.9139	0.8839	0.8856	0.9058	0.9245	0.9361
		t_{exe}	0.91	0.99	0.96	1.09	1.05	1.23
	LO-SONFIS	*ACC*	0.9206	0.8872	0.8969	0.9113	0.9196	0.9315
		t_{exe}	1.25	1.71	2.06	2.66	2.87	2.79
OR	SONFIS	*ACC*	0.9554	0.9705	0.9728	0.9762	0.9789	0.9797
		t_{exe}	0.35	0.31	0.34	0.37	0.39	0.44
	LO-SONFIS	*ACC*	0.9648	0.9741	0.9745	0.9769	0.9784	0.9802
		t_{exe}	0.41	0.38	0.42	0.47	0.49	0.53
PR	SONFIS	*ACC*	0.9392	0.9575	0.9637	0.9690	0.9745	0.9748
		t_{exe}	0.68	0.61	0.62	0.67	0.70	0.73
	LO-SONFIS	*ACC*	0.9515	0.9650	0.9674	0.9711	0.9752	0.9760
		t_{exe}	0.77	0.71	0.74	0.82	0.92	1.01



(a) Scenario 1 (b) Scenario 2

Fig. 10. The total number of prototypes (*NP*) identified by SONFIS during the learning processes

As one can see from Tables 3 and 4, the proposed PO algorithm can effectively improve the classification accuracy of SONFIS, and only slightly influences its computational efficiency. Moreover, it is worth to be noticed that the local optimality of the solutions is more important to SONFIS when the system identifies a smaller number of prototypes from data. This is because that with a lower level of granularity, SONFIS partitions the data space coarsely resulting in more space for further improvement. Therefore, in such cases, the proposed PO algorithm can significantly improve the performance of SONFIS.

Furthermore, the proposed PO algorithm is compared with the widely used PSO [9] and genetic learning PSO (GLPSO) [16] algorithms. In this example, both algorithms are applied to SONFIS for optimising the identified prototypes of each class after the offline training process. The performances of the PSO-optimised and GLPSO-optimised SONFISs (namely, PSO-SONFIS and GLPSO-SONFIS) are tested on the validation sets. The comparison in terms of classification accuracy and execution time (t_{exe} in sec) between LO-SONFIS, PSO-SONFIS and GLPSO-SONFIS is reported in Table 5, where the level of granularity of SONFIS, G varies from 1 to 6. The parameters of the PSO algorithm used for this example are set as: $\omega = 0.7298$; $\varphi_1 = 1.49618$; $\varphi_2 = 1.49618$; $\omega_{damp} = 1$; the parameters of the GLPSO algorithm are set as: $\omega = 0.7298$; $\varphi_1 = 1.49618$; $\varphi_2 = 1.49618$; $\varphi = 1.49618$; $\omega_{damp} = 1$; the population size for both PSO algorithms is equal to 50; the maximum iteration number is set as 100 and equation (22) is used as the cost function.

Table 5. Performance comparison between the proposed PO, PSO and GLPSO algorithms

Dataset	Algorithm	Measures	Granularity, G					
			1	2	3	4	5	6
WI	LO-SONFIS	*ACC*	0.5420	0.7080	0.7820	0.8100	0.8040	0.8000
		t_{exe}	1.07	1.16	1.38	1.54	1.57	1.78
	PSO-SONFIS	*ACC*	0.5506	0.6060	0.6748	0.7170	0.7276	0.7720
		t_{exe}	3.57	7.98	19.14	25.19	42.13	69.83
	GLPSO-SONFIS	*ACC*	0.5552	0.6224	0.7392	0.7772	0.7926	0.7692
		t_{exe}	6.31	12.59	34.27	49.92	68.62	125.19
OD	LO-SONFIS	*ACC*	0.8150	0.8405	0.8770	0.9165	0.9410	0.9524
		t_{exe}	2.30	2.63	3.07	3.59	2.95	3.25
	PSO-SONFIS	*ACC*	0.8244	0.8894	0.8809	0.8984	0.9216	0.9260
		t_{exe}	8.93	25.46	38.01	61.20	84.27	119.82
	GLPSO-SONFIS	*ACC*	0.8193	0.8662	0.8824	0.9190	0.9383	0.9429
		t_{exe}	15.98	49.80	75.62	120.89	170.69	239.17
OR	LO-SONFIS	*ACC*	0.9254	0.9505	0.9549	0.9755	0.9777	0.9777
		t_{exe}	0.12	0.12	0.12	0.15	0.15	0.15
	PSO-SONFIS	*ACC*	0.9264	0.9431	0.9544	0.9706	0.9738	0.9765
		t_{exe}	12.88	14.14	14.64	31.92	48.31	59.28
	GLPSO-SONFIS	*ACC*	0.9283	0.9425	0.9539	0.9706	0.9761	0.9760
		t_{exe}	30.19	31.98	32.65	68.08	100.55	122.19
PR	LO-SONFIS	*ACC*	0.9108	0.9528	0.9663	0.9720	0.9746	0.9771
		t_{exe}	0.34	0.35	0.37	0.40	0.40	0.42
	PSO-SONFIS	*ACC*	0.9123	0.9497	0.9545	0.9638	0.9670	0.9684
		t_{exe}	9.03	11.48	20.69	34.46	55.40	79.41
	GLPSO-SONFIS	*ACC*	0.9119	0.9506	0.9607	0.9696	0.9730	0.9760
		t_{exe}	20.14	25.37	43.48	70.60	114.11	162.17

As one can see from Table 5, despite that the PSO algorithms can effective update the positions of the identified prototypes in the data space by minimising the values of the cost function (namely, equation (22)), they did not significantly improve the classification accuracy of SONFIS compared with the proposed PO algorithm. In addition, PSO and GLPSO consume more computational resources. Therefore, one can conclude that the proposed PO algorithm is more suitable for SONFIS optimisation than PSO algorithms.

For better evaluation, the performance of the LO-SONFIS and SONFIS on WI, OD, OR and PR datasets is compared with the following state-of-the-art algorithms:

1) SVM classifier [10];

2) Decision tree (DT) classifier [37];

3) KNN classifier [35];

4) SOM classifier [30];

5) Back-propagation neural network (BPNN);

6) LVQ [23];

7) Long short-term memory (LSTM) network [14].

8) ESAFIS classifier [36];

9) eClass0 classifier [6];

10) Simpl_eClass0 classifier [7], and;

11) ALMMo0 classifier [3].

In the following numerical examples, SVM uses Gaussian kernel; k is equal to 10 for KNN; SOM classifier applies "winner takes all" principle for decision-making and the net size is 6×6; BPNN has three hidden layers and each hidden layer has 20 neurons; LVQ has one hidden layer, which is composed of 32 neurons; LSTM has three hidden layers and each hidden layer has 20 neurons. It has to be stressed that eClass0, Simpl_eClass0 and ALMMo0 classifiers are also prototype-based neuro-fuzzy systems and are of the same type as SONFIS. For fair comparison, SONFIS is trained offline and the level of granularity is set to be $G = 5$ to avoid overfitting. The statistic performance of the involved classification algorithms (accuracy, *ACC* and execution time, t_{exe} in sec) is reported in Table 6 in the form of $mean \pm standard\ deviation$ after 10 times Monte-Carlo experiments by randomly scrambling the order of training samples.

Table 6. Performance comparison on WI, OD, OR and PR datasets

Dataset	Algorithm	*ACC*	t_{exe}
WI	LO-SONFIS	0.8040±0.0000	1.57±0.05
	SONFIS	0.7960±0.0000	1.12±0.04
	SVM	0.6280±0.0000	1.83±0.90
	DT	0.8140±0.0000	0.04±0.06
	KNN	0.6920±0.0000	0.04±0.07
	SOM	0.6260±0.0000	2.73±1.83
	BPNN	0.6316±0.0171	0.65±0.51
	LVQ	0.6260±0.0000	179.12±2.48
	LSTM	0.6260±0.0000	5.51±0.78
	ESAFIS	0.6266±0.0025	5.53±3.09
	eClass0	0.4604±0.0025	0.25±0.04
	Simpl_eClass0	0.5428±0.0054	0.28±0.05
	ALMMo0	0.7640±0.0000	0.39±0.08
OD	LO-SONFIS	0.9410±0.0000	2.95±0.09
	SONFIS	0.9382±0.0000	2.51±0.07
	SVM	0.7607±0.0000	4.25±0.43
	DT	0.9314±0.0000	0.04±0.05
	KNN	0.9664±0.0000	0.04±0.04
	SOM	0.9651±0.0065	4.62±1.33
	BPNN	0.9259±0.0468	0.81±0.23
	LVQ	0.8859±0.0008	336.85±8.96
	LSTM	0.9634±0.0125	6.59±0.65

	ESAFIS	0.9617±0.0157	16.12±9.80
	eClass0	0.8858±0.0003	1.02±0.20
	Simpl_eClass0	0.9471±0.0006	1.04±0.11
	ALMMo0	0.9394±0.0000	0.63±0.11
OR	LO-SONFIS	0.9777±0.0000	0.15±0.07
	SONFIS	0.9766±0.0000	0.09±0.06
	SVM	0.1013±0.0000	2.74±0.70
	DT	0.8525±0.0000	0.06±0.03
	KNN	0.9766±0.0000	0.02±0.04
	SOM	0.9297±0.0075	8.83±1.49
	BPNN	0.9243±0.0077	0.70±0.24
	LVQ	0.8380±0.0275	174.27±14.81
	LSTM	0.7726±0.0544	5.61±0. 91
	ESAFIS	0.9538±0.0067	32.24±10.36
	eClass0	0.8937±0.0000	0.75±0.07
	Simpl_eClass0	0.9081±0.0002	1.63±0.21
	ALMMo0	0.9789±0.0000	0.33±0.09
PR	LO-SONFIS	0.9746±0.0000	0.40±0.09
	SONFIS	0.9743±0.0000	0.29±0.08
	SVM	0.1038±0.0000	8.90±0.91
	DT	0.9125±0.0000	0.06±0.06
	KNN	0.9748±0.0000	0.03±0.06
	SOM	0.8664±0.0022	5.27±1.39
	BPNN	0.9188±0.0097	1.05±0.24
	LVQ	0.8225±0.0035	326.29±14.28
	LSTM	0.8147±0.0410	6.88±1.09
	ESAFIS	0.9155±0.0131	30.00±1.69
	eClass0	0.8274±0.0002	0.56±0.06
	Simpl_eClass0	0.8768±0.0001	0.81±0.03
	ALMMo0	0.9706±0.0000	0.43±0.10

Furthermore, the performance of the involved algorithms in the previous numerical example is compared on MF, ES, LR and CM datasets. In the following numerical example, for MF, ES and LR datasets, all the data samples are firstly split into 10 folds evenly. Then, we randomly select five of the 10 folds to train the algorithms and use the remaining for validation. For CM dataset, the order of the training samples is randomly scrambled. The same experiment is repeated for 10 times with the statistical results reported in Table 7.

Table 7. Performance comparison on MF, ES and LR datasets

Dataset	Algorithm	*ACC*	t_{exe}
MF	LO-SONFIS	0.9210±0.0093	0.12±0.07
	SONFIS	0.9192±0.0090	0.08±0.06
	SVM	0.1027±0.0013	0.74±0.32
	DT	0.9210±0.0121	0.12±0.04
	KNN	0.9122±0.0094	0.02±0.03
	SOM	0.8192±0.0146	14.45±0.82
	BPNN	0.8648±0.0331	0.53±0.25
	LVQ	0.6529±0.0116	64.46±2.90
	LSTM	0.2054±0.0283	4.95±0.72
	ESAFIS	0.5938±0.1896	278.62±58.83
	eClass0	0.7990±0.0113	2.22±0.27
	Simpl_eClass0	0.8417±0.0118	4.53±0.89
	ALMMo0	0.9347±0.0050	0.12±0.04
ES	LO-SONFIS	0.9023±0.0077	6.98±0.72
	SONFIS	0.8884±0.0140	2.24±0.07

	SVM	0.8005±0.0042	4.35±0.84
	DT	0.9353±0.0038	0.57±0.08
	KNN	0.9032±0.0059	0.02±0.03
	SOM	0.9127±0.0073	29.65±5.55
	BPNN	0.9570±0.0045	1.02±0.40
	LVQ	0.8891±0.0073	324.10±52.76
	LSTM	0.8142±0.0042	7.98±5.91
	ESAFIS	0.2270±0.0278	349.11±65.37
	eClass0	0.8504±0.0260	2.63±0.10
	Simpl_eClass0	0.8427±0.0228	7.24±0.47
	ALMMo0	0.8936±0.0023	25.77±3.19
LR	LO-SONFIS	0.9240±0.0049	0.37±0.16
	SONFIS	0.9223±0.0052	0.27±0.15
	SVM	0.3799±0.0373	14.21±1.23
	DT	0.8235±0.0068	0.12±0.05
	KNN	0.9201±0.0032	0.04±0.04
	SOM	0.3231±0.0074	14.60±1.50
	BPNN	0.4799±0.0328	1.34±0.23
	LVQ	0.0379±0.0015	762.20±46.92
	LSTM	0.4727±0.0290	24.42±11.00
	ESAFIS	0.4394±0.0116	5.90±0.53
	eClass0	0.4833±0.0087	1.26±0.13
	Simpl_eClass0	0.5736±0.0065	1.68±0.07
	ALMMo0	0.9179±0.0029	0.76±0.19
CM	LO-SONFIS	0.6433±0.0000	4.38±0.29
	SONFIS	0.6500±0.0000	3.52±0.27
	SVM	0.2600±0.0000	30.74±3.81
	DT	0.5767±0.0000	0.16±0.06
	KNN	0.6267±0.0000	0.03±0.05
	SOM	0.4447±0.0149	10.03±1.58
	BPNN	0.4757±0.0352	0.76±0.20
	LVQ	0.4317±0.0069	457.30±26.83
	LSTM	0.2940±0.0203	29.83±6.46
	ESAFIS	0.5503±0.0135	63.33±9.45
	eClass0	0.3480±0.0063	1.41±0.20
	Simpl_eClass0	0.3333±0.0000	2.46±0.11
	ALMMo0	0.5740±0.0216	5.79±0.83

Since the proposed PO algorithm is a generic approach and can be used for optimising other learning algorithms with similar operating mechanisms, in the following example, we use the PO algorithm for optimising the eClass0, Simpl_eClass0 and ALMMo0 classifiers. The same experiments presented in Tables 6 and 7 are repeated under the same experimental protocols and the numerical results are reported in Table 8. The optimised classifiers by the proposed PO algorithm are re-denoted as LO-eClass0, LO-Simpl_eClass0 and LO-ALMMo0, respectively. The original results obtained by the three classifiers are reported as baseline. The results of the LO-SONFIS and SONFIS are also given for better illustration.

Table 8. Performance of the locally optimised eClass0, Simpl_eClass0 and ALMMo-0 classifiers

Dataset	Algorithm	*NP*	*ACC*	t_{exe}
WI	LO-SONFIS	110.00±0.00	0.8040±0.0000	1.57±0.05
	SONFIS		0.7960±0.0000	1.12±0.04
	LO-eClass0	6.80±0.63	0.6704±0.0358	0.21±0.04
	eClass0		0.4604±0.0025	0.19±0.04
	LO-Simpl_eClass0	13.00±0.00	0.6938±0.0321	0.20±0.03
	Simpl_eClass0		0.5428±0.0054	0.18±0.03

	LO-ALMMo0	462.50±9.50	0.7748±0.0224	0.96±0.10
	ALMMo0		0.7640±0.0000	0.39±0.08
OD	LO-SONFIS	201.00±0.00	0.9410±0.0000	2.95±0.09
	SONFIS		0.9382±0.0000	2.51±0.07
	LO-eClass0	17.00±0.00	0.9507±0.0002	0.57±0.20
	eClass0		0.8858±0.0003	0.53±0.21
	LO-Simpl_eClass0	27.00±0.00	0.9658±0.0006	1.71±0.09
	Simpl_eClass0		0.9471±0.0006	0.39±0.02
	LO-ALMMo0	432.70±19.02	0.9423±0.0058	1.23±0.19
	ALMMo0		0.9394±0.0000	0.63±0.11
OR	LO-SONFIS	409.00±0.00	0.9777±0.0000	0.15±0.07
	SONFIS		0.9766±0.0000	0.09±0.06
	LO-eClass0	75.00±0.00	0.9563±0.0013	0.79±0.07
	eClass0		0.8937±0.0000	0.72±0.05
	LO-Simpl_eClass0	142.00±0.00	0.9613±0.0004	1.45±0.03
	Simpl_eClass0		0.9081±0.0002	1.37±0.04
	LO-ALMMo0	1573.40±13.78	0.9787±0.0017	0.37±0.10
	ALMMo0		0.9789±0.0000	0.33±0.09
PR	LO-SONFIS	747.00±0.00	0.9746±0.0000	0.40±0.09
	SONFIS		0.9743±0.0000	0.29±0.08
	LO-eClass0	62.90±0.32	0.9000±0.0016	0.58±0.07
	eClass0		0.8274±0.0002	0.52±0.06
	LO-Simpl_eClass0	139.90±0.32	0.9376±0.0016	0.96±0.11
	Simpl_eClass0		0.8768±0.0001	0.89±0.11
	LO-ALMMo0	1565.70±13.82	0.9759±0.0022	0.63±0.13
	ALMMo0		0.9706±0.0000	0.43±0.10
MF	LO-SONFIS	179.70±4.99	0.9210±0.0093	0.12±0.07
	SONFIS		0.9192±0.0090	0.08±0.06
	LO-eClass0	70.80±2.97	0.9104±0.0067	1.92±0.08
	eClass0		0.7990±0.0113	1.84±0.10
	LO-Simpl_eClass0	100.20±4.52	0.9247±0.0055	2.94±0.25
	Simpl_eClass0		0.8417±0.0118	2.87±0.26
	LO-ALMMo0	220.90±5.74	0.9358±0.0038	0.17±0.04
	ALMMo0		0.9347±0.0050	0.12±0.04
ES	LO-SONFIS	944.70±13.47	0.9023±0.0077	6.98±0.72
	SONFIS		0.8884±0.0140	2.24±0.07
	LO-eClass0	6.00±0.00	0.8669±0.0091	2.48±0.11
	eClass0		0.8504±0.0260	2.26±0.07
	LO-Simpl_eClass0	24.00±0.00	0.8508±0.0202	5.48±0.28
	Simpl_eClass0		0.8427±0.0228	5.04±0.10
	LO-ALMMo0	5369.90±10.01	0.8935±0.0023	33.54±1.17
	ALMMo0		0.8936±0.0023	25.77±3.19
LR	LO-SONFIS	1510.50±25.79	0.9240±0.0049	0.37±0.16
	SONFIS		0.9223±0.0052	0.27±0.15
	LO-eClass0	153.60±1.51	0.7079±0.0287	1.04±0.04
	eClass0		0.4833±0.0087	0.94±0.04
	LO-Simpl_eClass0	332.70±7.63	0.8012±0.0123	0.73±0.06
	Simpl_eClass0		0.5736±0.0065	0.64±0.04
	LO-ALMMo0	2036.30±36.16	0.9252±0.0024	1.00±0.15
	ALMMo0		0.9179±0.0029	0.76±0.19
CM	LO-SONFIS	509.00±0.00	0.6433±0.0000	4.38±0.29
	SONFIS		0.6500±0.0000	3.52±0.27
	LO-eClass0	53.90±9.22	0.5543±0.0127	2.19±0.47
	eClass0		0.3480±0.0063	1.41±0.20
	LO-Simpl_eClass0	112.30±29.13	0.5560±0.0107	5.25±1.28

	Simpl_eClass0		0.3333±0.0000	2.46±0.11
	LO-ALMMo0	3397.60±21.33	0.5717±0.0178	3.15±0.60
	ALMMo0		0.5740±0.0216	5.79±0.83

Finally, we conduct numerical experiments on FC dataset to further evaluate the effectiveness of the proposed PO algorithm on improving the classification performance of SONFIS, eClass0, Simpl_eClass0 and ALMMo0. We follow the same experimental protocol as used in the numerical examples presented in Table 7 by evenly splitting all the data samples into 10 folds and randomly selecting five of the 10 folds to train the algorithms and using the remaining for validation. LVQ, LSTM, SOM and ESAFIS algorithms are not involved for comparison because their computational efficiency is significantly low on large-scale datasets. In this experiment, the SONFIS is primed with 10% training samples in an offline manner and continuously updated with the remaining data on a sample-by-sample basis; the level of granularity of SONFIS is set as: $G = 12$ due to the larger scale and more complex structure of the problem. Classification performance of the involved algorithms in terms of accuracy (*ACC*) is reported in Table 9.

Table 9. Performance comparison on FC dataset

Algorithm	*ACC*
LO-SONFIS	0.9259±0.0005
SONFIS	0.9245±0.0007
LO-eClass0	0.5330±0.0011
eClass0	0.4281±0.0006
LO-Simpl_eClass0	0.4878±0.0103
Simpl_eClass0	0.3479±0.0005
LO-ALMMo0	0.9046±0.0003
ALMMo0	0.8934±0.0006
SVM	0.7981±0.0003
DT	0.9181±0.0010
KNN	0.9107±0.0006
BPNN	0.7227±0.0063

7.2. Experiments on Benchmark Image Sets

In this subsection, we further use the following benchmark image sets to justify the validity and effectiveness of the proposed PO algorithm on image classification problems:

1) MNIST image set[11];

2) Fashion MNIST image set [12];

3) Singapore image set[13];

4) RSSCN7 image set[14];

5) Caltech101 image set[15], and;

6) Caltech256 image set[16].

Detailed descriptions of the six image sets are as follows.

[11] Available from http://yann.lecun.com/exdb/mnist/
[12] Available from https://github.com/zalandoresearch/fashion-mnist
[13] Available from http://icn.bjtu.edu.cn/Visint/resources/Scenesig.aspx
[14] Available from https://sites.google.com/view/zhouwx/dataset
[15] Available from http://www.vision.caltech.edu/Image_Datasets/Caltech101/
[16] Available from http://www.vision.caltech.edu/Image_Datasets/Caltech256/

1) MNIST dataset

MNIST dataset is a famous benchmark database for handwritten digit recognition. This dataset contains 70000 greyscale images of handwritten digits from "0" to "9", 60000 of which are used for training and the remaining 10000 images are for validation/testing. The amounts of training and validation images of the 10 classes are more or less balanced. The size of both training and validation images is 28×28 pixels.

2) Fashion MNIST dataset

Fashion MNIST dataset is a new dataset composed of 70000 greyscale images of different fashion product from 10 classes: *1)* T-shirt; *2)* trouser; *3)* pullover; *4)* dress; *5)* coat; *6)* sandals; *7)* shirt; *8)* sneaker; *9)* bag and *10)* ankle boots. Each category has 7000 images with the 28×28 pixel size, 6000 of them are used for training, and the other 1000 images are used for testing.

3) Singapore dataset

Singapore dataset is a recently introduced benchmark image set for remote sensing scene classification. This dataset consists of 1086 images of 256×256 pixels size. These images belong to nine land-use categories: *i)* airplane; *ii)* forest; *iii)* harbour; *iv)* industry; *v)* meadow; *vi)* overpass; *vii)* residential; *viii)* river and *ix)* runway. The numbers of images of the nine land-use categories are imbalanced varying from 42 to 179.

4) RSSCN7 dataset

RSSCN7 dataset is collected from Google Earth (Google Inc.). This dataset has seven land-use categories including: *i)* grassland; *ii)* forest; *iii)* farmland; *iv)* parking lot; *v)* resident; *vi)* industry and *vii)* river and lake. Each land-use category contains 400 images of size 600×600 pixels. The images of each land-use category are sampled at four different scales (100 images per scale) with different angles and, thus, classifying images of this dataset is very challenging.

5) Caltech101 dataset and 6) Caltech256 dataset

Caltech101 dataset has more than 8677 images belonging to 101 classes. There are 31 to 800 images for each class, and the size of each image is roughly 200×300 pixels. Caltech256 dataset is the extended dataset of Caltech101, which has 256 classes. The minimum number of images per class for the Caltech256 dataset is 80, and in total, there are 29780 images. Caltech101 and Caltech256 datasets both contain classes corresponding to rigid object (like bikes and cars) and classes corresponding to non-rigid object (like animals and flowers) with various backgrounds, and, thus, they are very challenging problems. Example images of the six benchmark datasets are given in Fig. 11(a)-(f).

For MNIST and Fashion MNIST image sets, images are firstly converted into 784×1 dimensional vectors and, then, directly used for training and testing the classification algorithms. For Singapore, RSSCN7, Caltech101 and Caltech256 datasets, a high-level ensemble descriptor using the pre-trained AlexNet [24] and VGG-VD-16 [39] deep learning neural networks is created for feature extraction. The feature extraction process for converting a particular image $\mathbf{I}$ into a feature vector $\boldsymbol{x}$ is expressed as:

$$\boldsymbol{x} = \mathrm{F}(\mathbf{I}) = \left[\frac{\mathrm{AN}(\mathbf{I})}{\|\mathrm{AN}(\mathbf{I})\|}, \frac{\mathrm{VN}(\mathbf{I})}{\|\mathrm{VN}(\mathbf{I})\|}\right]^T \tag{35}$$

where $\mathrm{F}(\mathbf{I})$ represents 9192×1 dimensional representation extracted from $\mathbf{I}$ by the ensemble feature descriptor; $\mathrm{AN}(\mathbf{I})$ and $\mathrm{VN}(\mathbf{I})$ are the 1×4096 dimensional feature vectors extracted from the first fully connected layer of the AlexNet and VGG-VD-16 models, respectively. In addition, for Singapore and RSSCN7 datasets, we adopt the commonly used "centre, four corners and horizontal flipping" data augmentation process and use the mean of feature vectors of the 10 sub-images created from each remote sensing image as its corresponding feature vector [24]. In the numerical examples presented in this subsection, SONFIS and LO-SONFIS use cosine dissimilarity for classification [18], and the level of granularity is set to be $G = 12$.

Firstly, the effectiveness and validity of the proposed PO algorithm on image classification problems are justified on MNIST and Fashion MNIST datasets. In the following numerical example, SONFIS is primed offline with 10000 training images, and then, uses 20000, 30000, 40000 and 50000 training images for online learning. Thus, in total, there are 30000, 40000, 50000 and 60000 training images used for experiments,

respectively. After the online learning process, the identified prototypes of SONFIS are optimised by the PO algorithm with the images involved during the overall learning process. After being optimised, the performance of LO-SONFIS is, then, evaluated on the testing images. The average accuracy (*ACC*) of the classification results by LO-SONFIS after 10 times Monte-Carlo experiments are tabulated in Table 10. The results by SONFIS under the same experimental protocol are reported as the baseline. Furthermore, the following algorithms are involved for comparison:

1) SVM classifier [10];

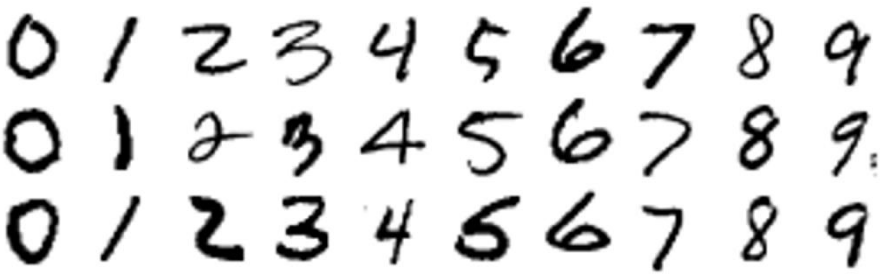
2) KNN classifier [35];

3) eClass0 classifier [6];

4) Simpl_eClass0 classifier [7], and;

5) ALMMo0 classifier [3].

In the experiments, SVM uses linear kernel; k is equal to 1 for KNN. The optimised eClass0, Simpl_eClass0 and ALMMo0 by using the proposed PO algorithm, namely, LO-eClass0, LO-Simpl_eClass0 and LO-ALMMo0 are also involved.



(a) MNIST

(b) Fashion MNIST

(c) Singapore

(d) RSSCN7

(e) Caltech101

(f) Caltech256

Fig. 11. Examples of the benchmark image sets

Table 10. Performance comparison on MNSIT and Fashion MNIST datasets

Dataset	Algorithm	*30000*	*40000*	*50000*	*60000*
MNSIT	LO-SONFIS	0.9617±0.0017	0.9650±0.0013	0.9665±0.0008	0.9686±0.0010
	SONFIS	0.9621±0.0017	0.9646±0.0013	0.9662±0.0010	0.9681±0.0011
	LO-eClass0	0.9231±0.0034	0.9255±0.0012	0.9240±0.0014	0.9250±0.0045
	eClass0	0.7557±0.0024	0.7565±0.0000	0.7569±0.0003	0.7354±0.0000
	LO-Simpl_eClass0	0.9340±0.0014	0.9361±0.0015	0.9384±0.0019	0.9362±0.0009
	Simpl_eClass0	0.7719±0.0004	0.7717±0.0000	0.7743±0.0009	0.7528±0.0000
	LO-ALMMo0	0.9624±0.0014	0.9651±0.0014	0.9678±0.0014	0.9690±0.0013
	ALMMo0	0.9621±0.0018	0.9649±0.0015	0.9672±0.0016	0.9683±0.0013
	SVM	0.9370±0.0016	0.9403±0.0012	0.9424±0.0016	0.9438±0.0000
	KNN	0.9632±0.0011	0.9661±0.0013	0.9672±0.0010	0.9691±0.0000
Fashion MNIST	LO-SONFIS	0.8478±0.0032	0.8523±0.0028	0.8575±0.0015	0.8610±0.0021
	SONFIS	0.8483±0.0027	0.8537±0.0020	0.8583±0.0016	0.8610±0.0015
	LO-eClass0	0.7790±0.0037	0.7785±0.0022	0.7785±0.0053	0.7798±0.0068
	eClass0	0.6535±0.0012	0.6539±0.0000	0.6539±0.0000	0.6539±0.0000
	LO-Simpl_eClass0	0.7825±0.0028	0.7852±0.0020	0.7943±0.0043	0.7942±0.0032

	Simpl_eClass0	0.6624±0.0009	0.6618±0.0000	0.6618±0.0000	0.6618±0.0000
	LO-ALMMo0	0.8429±0.0017	0.8503±0.0018	0.8547±0.0014	0.8597±0.0012
	ALMMo0	0.8432±0.0021	0.8498±0.0023	0.8543±0.0017	0.8589±0.0015
	SVM	0.8417±0.0016	0.8457±0.0011	0.8486±0.0018	0.8498±0.0001
	KNN	0.8349±0.0020	0.8384±0.0020	0.8444±0.0017	0.8497±0.0000

Then, we use Singapore and RSSCN7 datasets to further evaluate the effectiveness of the proposed PO algorithm on improving the classification accuracy of zero-order EISs, namely, SONFIS, eClass0, Simpl_eClass0 and ALMMo0. The same SVM and KNN algorithms used in the previous numerical example are also involved. Following the commonly used experimental protocols [12],[44], for Singapore dataset, 20% images per class are randomly selected out for training and the remaining images are used for validation. For RSSCN7 dataset, 20% and 50% images per class are randomly selected out for training, respectively, and the remaining images are used for validation. The average classification accuracy rates by the classification algorithms on the two datasets are reported in Tables 11 and 12, respectively, after 10 times Monte-Carlo experiments. Furthermore, selected state-of-the-art results in the literature are reported in the two tables for informed comparison.

Table 11. Numerical results on Singapore dataset

Algorithm	*ACC*
LO-SONFIS	0.9718±0.0051
SONFIS	0.9713±0.0054
LO-eClass0	0.9685±0.0051
eClass0	0.9379±0.0070
LO-Simpl_eClass0	0.9713±0.0054
Simpl_eClass0	0.9528±0.0061
LO-ALMMo0	0.9685±0.0060
ALMMo0	0.9684±0.0060
SVM	0.9726±0.0053
KNN	0.9700±0.0066
TLFP [12]	0.9094
BoVW [45]	0.8741
VLAD [21]	0.8878
SPM [26]	0.8285

Table 12. Numerical results on RSSCN7 datasets

Algorithm	*ACC*	
	20% Training Images	50% Training Images
LO-SONFIS	0.8741±0.0081	0.9041±0.0071
SONFIS	0.8741±0.0082	0.9042±0.0074
LO-eClass0	0.8568±0.0114	0.8732±0.0079
eClass0	0.7436±0.0137	0.7511±0.0075
LO-Simpl_eClass0	0.8778±0.0079	0.9003±0.0078
Simpl_eClass0	0.7671±0.0084	0.7770±0.0075
LO-ALMMo0	0.8681±0.0068	0.9041±0.0072
ALMMo0	0.8681±0.0069	0.9043±0.0071
SVM	0.8798±0.0085	0.9073±0.0070
KNN	0.8707±0.0075	0.9072±0.0070
CaffeNet [44]	0.8557±0.0095	0.8885±0.0062
GoogLeNet [44]	0.8398±0.0087	0.8718±0.0094
VGG-VD-16 [44]	0.8255±0.0111	0.8584±0.0092
BoVW(SIFT) [44]	0.7633±0.0088	0.8134±0.0055
VLAD(SIFT) [44]	0.8082±0.0215	0.7727±0.0058
DBNFS [50]	0.7581	0.7119

Finally, the classification performance (in terms of accuracy, *ACC*) of LO-SONFIS and SONFIS are tested on Caltech101 and Caltech256 datasets. Following the commonly used experimental protocol [13], for Caltech101 dataset, 15, 30 images per class are randomly selected out for training, respectively, and the remaining images are used for validation. For Caltech256 dataset, 15, 30, 45 and 60 images per class are randomly selected out for training, respectively, and the remaining images are used for validation. The average classification accuracy obtained by LO-SONFIS and SONFIS on the two datasets is reported in Tables 13 and 14 after 10 times Monte-Carlo experiments. The same SVM and KNN algorithms used in the previous examples are tested on the two datasets as well, and their classification accuracy rates are reported in Tables 13 and 14. We further report the selected state-of-the-art results in the literature for informed comparison. The average numbers of prototypes (*NP*) per class identified by SONFIS during the experiments are given in Fig. 12.

From the numerical examples presented in this subsection one can see that both SONFIS and LO-SONFIS produced the highly accurate classification results surpassing or on par with the state-of-the-art approaches.

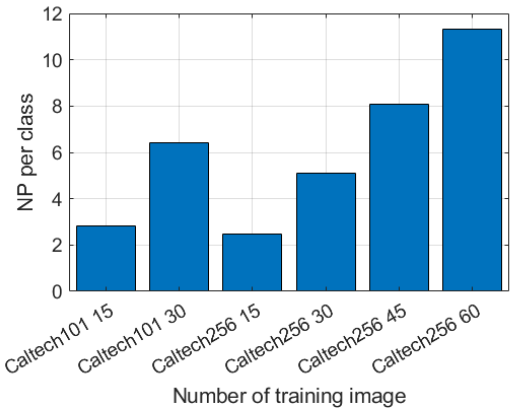


Fig. 12. The average numbers of prototypes (*NP*) per class identified by SONFIS during the experiments

Table 13. Numerical results on Caltech101 dataset

Algorithm	*ACC*	
	15 Training Images	30 Training Images
LO-SONFIS	0.8978±0.0050	0.9230±0.0034
SONFIS	0.8979±0.0048	0.9231±0.0036
SVM	0.8729±0.0104	0.9027±0.0087
KNN	0.8670±0.0063	0.8999±0.0051
ICAC [47]	0.7148±0.0056	0.7663±0.0079
CASE-LLC-SVM [42]	0.6400±0.0040	0.7140±0.0120
ScSPM [46]	0.6700±0.0045	0.7320±0.0054
DEFEATnet [13]	0.7128±0.0061	0.7760±0.0096

Table 14. Numerical results on Caltech256 dataset

Algorithm	*ACC*			
	15 Training Images	30 Training Images	45 Training Images	60 Training Images
LO-SONFIS	0.6799±0.0033	0.7113±0.0029	0.7272±0.0033	0.7416±0.0041
SONFIS	0.6798±0.0033	0.7114±0.0029	0.7270±0.0034	0.7407±0.0040
SVM	Out of System Memory			
KNN	0.6249±0.0033	0.6723±0.0026	0.6986±0.0032	0.7210±0.0027
SWSS-DeCAF [48]	0.6152±0.0039	0.6768±0.0065	0.6977±0.0053	0.7283±0.0044
SWSS-FV [48]	0.4246±0.0038	0.4985±0.0042	0.5466±0.0047	0.5652±0.0041
SC^2-CNN [49]	0.4758±0.0062	0.5542±0.0056	0.5912±0.0051	0.6174±0.0050

ScSPM [46]	0.2773±0.0051	0.3402±0.0035	0.3746±0.0055	0.4014±0.0091
DEFEATnet [13]	0.3507±0.0038	0.4206±0.0025	0.4598±0.0026	0.4852±0.0032

7.3. Discussions

Numerical examples on benchmark numerical datasets and image sets presented in this section demonstrate that the proposed PO algorithm can effectively improve the classification accuracy of SONFIS, eClass0, Simpl_eClass0 and ALMMo0 on various types of problems. The proposed PO algorithm is more computationally efficient than PSO algorithms and more effective on complex, large-scale problems. In addition, it only has slight influence on the computational efficiency of the learning algorithms and costs little extra memory resources.

However, one may notice that the performance of eClass0 and Simpl_eClass0 increases much more after being optimised by the proposed PO algorithm compared with SONFIS and ALMMo0. This is due to the differences in the operating mechanisms of the classification algorithms. eClass0 and Simpl_eClass0 usually extract a smaller number of prototypes from data samples compared with SONFIS and ALMMo0 (see Table 8), which results in a coarse partitioning of the data space and leaves more space for further improvement. As a result, the PO algorithm is able to play a more significant role in improving their classification performance.

It is also interesting to notice that when the training set contains many incorrect labelled samples, PO algorithm actually decreases the classification accuracy of SONFIS and ALMMo0 as they are more sensitive to noise because of the larger number of prototypes identified from data. In such cases, anomalies are highly likely to be recognised as prototypes because of their very different patterns from the majority, and they create lots of confusions during the validation stage.

8. Conclusion and Future Work

In this paper, we use the recently introduced SONFIS as an example to study the local optimality of zero-order EISs. Based on a detailed mathematical analysis, it is proven that SONFIS is not able to obtain a locally optimal solution from data through the "one pass" type learning process. Following this conclusion, we, then, propose an optimisation algorithm that enables SONFIS to self-adjust the locations of its prototypes based on historically observed data and finally achieve the local optimal solution. Numerical examples on benchmark datasets demonstrate the validity of the optimality analysis and the effectiveness of the proposed optimisation algorithm. Moreover, it is further numerically proven that the proposed concepts and general principles are also applicable to other types of zero-order EISs with similar operating mechanisms

As future work, we will extend this study to first-order EISs by further investigating the optimality of both the premise, IF and consequent, THEN parts. Since the optimality analysis conducted within this paper mostly concerns the data partitioning results obtained by zero-order EISs, we will also extend this study to clustering algorithms and other prototype-based semi-supervised classification approaches. Another interesting direction for more future works is to find an effective way to reduce the number of prototypes during the optimisation process for simplifying the system structure and, meanwhile, maintain the same level of classification accuracy.

References

[1] P. Angelov, Autonomous learning systems: from data streams to knowledge in real time. John Wiley & Sons, Ltd., 2012.

[2] P. P. Angelov and D. P. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," IEEE Trans. Syst. Man, Cybern. - Part B Cybern., vol. 34, no. 1, pp. 484–498, 2004.

[3] P. P. Angelov and X. Gu, "Autonomous learning multi-model classifier of 0-order (ALMMo-0)," in IEEE International Conference on Evolving and Autonomous Intelligent Systems, 2017, pp. 1–7.

[4] P. P. Angelov, X. Gu, and J. Principe, "A generalized methodology for data analysis," IEEE Trans. Cybern., vol. 48, no. 10, pp. 2981–2993, 2018.

[5] P. P. Angelov and R. Yager, "A new type of simplified fuzzy rule-based system," Int. J. Gen. Syst., vol. 41, no. 2, pp. 163–185, 2012.

[6] P. Angelov and X. Zhou, "Evolving fuzzy-rule based classifiers from data streams," IEEE Trans. Fuzzy

Syst., vol. 16, no. 6, pp. 1462–1474, 2008.
[7] R. D. Baruah, P. P. Angelov, and J. Andreu, “Simpl _ eClass : simplified potential-free evolving fuzzy rule-based classifiers,” in IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2011, pp. 2249–2254.
[8] O. Castillo, L. Cervantes, J. Soria, M. Sanchez, and J. R. Castro, “A generalized type-2 fuzzy granular approach with applications to aerospace,” Inf. Sci. (Ny)., vol. 354, pp. 165–177, 2016.
[9] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” IEEE Trans. Evol. Comput., vol. 6, no. 1, pp. 58–73, 2002.
[10] N. Cristianini and J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods. Cambridge: Cambridge University Press, 2000.
[11] A. Fernandez, F. Herrera, O. Cordon, M. Jose Del Jesus, and F. Marcelloni, “Evolutionary fuzzy systems for explainable artificial intelligence: why, when, what for, and where to?,” IEEE Comput. Intell. Mag., vol. 14, no. 1, pp. 69–81, 2019.
[12] J. Gan, Q. Li, Z. Zhang, and J. Wang, “Two-level feature representation for aerial scene classification,” IEEE Geosci. Remote Sens. Lett., vol. 13, no. 11, pp. 1626–1630, 2016.
[13] S. Gao, L. Duan, and I. W. Tsang, “DEFEATnet—a deep conventional image representation for image classification,” IEEE Trans. Circuits Syst. Video Technol., vol. 26, no. 3, pp. 494–505, 2016.
[14] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, “Learning Precise Timing with LSTM Recurrent Networks,” J. Mach. Learn. Res., vol. 3, no. 1, pp. 115–143, 2002.
[15] L. Gómez-Chova, G. Camps-Valls, J. Munoz-Mari, and J. Calpe, “Semisupervised image classification with Laplacian support vector machines,” IEEE Geosci. Remote Sens. Lett., vol. 5, no. 3, pp. 336–340, 2008.
[16] Y. J. Gong et al., “Genetic learning particle swarm optimization,” IEEE Trans. Cybern., vol. 46, no. 10, pp. 2277–2290, 2016.
[17] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. Crambridge, MA: MIT Press, 2016.
[18] X. Gu and P. P. Angelov, “Self-organising fuzzy logic classifier,” Inf. Sci. (Ny)., vol. 447, pp. 36–51, 2018.
[19] X. Gu and P. Angelov, “Self-boosting first-order autonomous learning neuro-fuzzy systems,” Appl. Soft Comput., vol. 77, pp. 118–134, 2019.
[20] P. Hajek, “Predicting corporate investment/non-investment grade by using interval-valued fuzzy rule-based systems—a cross-region analysis,” Appl. Soft Comput. J., vol. 62, pp. 73–85, 2018.
[21] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact representation,” IEEE Conf. Comput. Vis. Pattern Recognit., pp. 3304–3311, 2010.
[22] N. K. Kasabov and Q. Song, “DENFIS : dynamic evolving neural-fuzzy inference system and its application for time-series prediction,” IEEE Trans. Fuzzy Syst., vol. 10, no. 2, pp. 144–154, 2002.
[23] T. Kohonen, Self-organizing maps. Berlin: Springer, 1997.
[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Advances In Neural Information Processing Systems, 2012, pp. 1097–1105.
[25] H. W. Kuhn and A. W. Tucker, “Nonlinear programming,” in Proceedings of the Second Symposium on Mathematical Statistics and Probability, 1951, pp. 481–492.
[26] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features : spatial pyramid matching for recognizing natural scene categories,” in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, pp. 2169–2178.
[27] A. Lemos, W. Caminhas, and F. Gomide, “Adaptive fault detection and diagnosis using an evolving fuzzy classifier,” Inf. Sci. (Ny)., vol. 220, pp. 64–85, 2013.
[28] E. Lughofer, Evolving fuzzy systems-methodologies, advanced concepts and applications. Berlin: Springer, 2011.
[29] E. Lughofer and P. Angelov, “Handling drifts and shifts in on-line data streams with evolving fuzzy systems,” Appl. Soft Comput., vol. 11, no. 2, pp. 2057–2068, 2011.
[30] P. Płoński and K. Zaremba, “Self-organising maps for classification with metropolis-hastings algorithm for supervision,” in International Conference on Neural Information Processing, 2012, pp. 149–156.
[31] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, “PANFIS : a novel incremental learning machine,” IEEE Trans. Neural Networks Learn. Syst., vol. 25, no. 1, pp. 55–68, 2014.
[32] M. Pratama, S. G. Anavatti, and E. Lughofer, “Genefis: toward an effective localist network,” IEEE Trans. Fuzzy Syst., vol. 22, no. 3, pp. 547–562, 2014.
[33] M. Pratama, J. Lu, and G. Zhang, “Evolving type-2 fuzzy classifier,” IEEE Trans. Fuzzy Syst., vol. 24, no. 3, pp. 574–589, 2016.
[34] R. E. Precup, T. A. Teban, A. Albu, A. I. Szedlak-Stinean, and C. A. Bojan-Dragos, “Experiments in incremental online identification of fuzzy models of finger dynamics,” Rom. J. Inf. Sci. Technol., vol. 21, no. 4, pp. 358–376, 2018.
[35] S. Ramaswamy, R. Rastogi, and K. Shim, “Efficient algorithms for mining outliers from large data sets,” ACM SIGMOD Rec., pp. 427–438, 2000.

[36] H. J. Rong, N. Sundararajan, G. Bin Huang, and G. S. Zhao, “Extended sequential adaptive fuzzy inference system for classification problems,” Evol. Syst., vol. 2, no. 2, pp. 71–82, 2011.
[37] S. R. Safavian and D. Landgrebe, “A survey of decision tree classifier methodology,” IEEE Trans. Syst. Man. Cybern., vol. 21, no. 3, pp. 660–674, 1990.
[38] S. Z. Selim and M. A. Ismail, “K-means-type algorithms: a generalized convergence theorem and characterization of local optimality,” IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-6, no. 1, pp. 81–87, 1984.
[39] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in International Conference on Learning Representations, 2015, pp. 1–14.
[40] I. Škrjanc, J. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide, “Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey,” Inf. Sci. (Ny)., vol. 490, pp. 344–368, 2019.
[41] J. Soto, P. Melin, and O. Castillo, “A new approach for time series prediction using ensembles of IT2FNN models with optimization of fuzzy integrators,” Int. J. Fuzzy Syst., vol. 20, no. 3, pp. 701–728, 2018.
[42] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, “Locality-constrained linear coding for image classification,” in IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 3360–3367.
[43] R. E. Wendell and A. P. Hurter Jr, “Minimization of a non-separable objective function subject to disjoint constraints,” Oper. Res., vol. 24, no. 4, pp. 643–657, 1976.
[44] G. Xia et al., “AID: a benchmark dataset for performance evaluation of aerial scene classification,” IEEE Trans. Geosci. Remote Sens., vol. 55, no. 7, pp. 3965–3981, 2017.
[45] Y. Yang and S. Newsam, “Bag-of-visual-words and spatial extensions for land-use classification,” in International Conference on Advances in Geographic Information Systems, 2010, pp. 270–279.
[46] J. Yang, K. Yu, Y. Gong, and T. Huang, “Linear spatial pyramid matching using sparse coding for image classification,” in IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1794–1801.
[47] C. Zhang, J. Cheng, and Q. Tian, “Incremental codebook adaptation for visual representation and categorization,” IEEE Trans. Cybern., vol. 48, no. 7, pp. 2012–2023, 2018.
[48] C. Zhang, J. Cheng, and Q. Tian, “Structured weak semantic space construction for visual categorization,” IEEE Trans. Neural Networks Learn. Syst., vol. 29, no. 8, pp. 3442–3451, 2018.
[49] C. Zhang, C. Li, D. Lu, J. Cheng, and Q. Tian, “Birds of a feather flock together: visual representation with scale and class consistency,” Inf. Sci. (Ny)., vol. 460–461, pp. 115–127, 2018.
[50] Q. Zou, L. Ni, T. Zhang, and Q. Wang, “Deep learning based feature selection for remote sensing scene classification,” IEEE Geosci. Remote Sens. Lett., vol. 12, no. 11, pp. 2321–2325, 2015.