Self-Organising Fuzzy Logic Classifier

Xiaowei Gu[1] and Plamen P. Angelov[1,2*]
[1]School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK
[2]Technical University, Sofia, 1000, Bulgaria (Honorary Professor)
e-mail: {x.gu3, p.angelov}@lancaster.ac.uk

Abstract- In this paper, we present a self-organising nonparametric fuzzy rule-based classifier. The proposed approach identifies prototypes from the observed data through an offline training process and uses them to build a 0-order AnYa type fuzzy rule-based system for classification. Once primed offline, it is able to continuously learn from the streaming data afterwards to follow the changing data pattern by updating the system structure and meta-parameters recursively. The meta-parameters of the proposed approach are derived from data directly. By changing the level of granularity, the proposed approach can make a trade-off between performance and computational efficiency, and, thus, the classifier is able to address a wide variety of problems with specific needs. The classifier also supports different types of distance measures. Numerical examples based on benchmark datasets demonstrate the high performance of the proposed approach and its ability of handling high-dimensional, complex, large-scale problems.
Keywords- classification, fuzzy rule-based systems, self-organising, recursive

1. Introduction

Classification is one of the hotly studied problems in machine learning [24]. Till now, various classification algorithms have been successfully developed and widely used in different areas i.e. remote sensing [46],[47], face recognition [10],[25], handwritten digits recognition [13],[21], etc.

Current classification approaches have different architectures. In general, considering their operating mechanisms, the existing approaches can be categorised into two major types: *1)* offline [13],[14],[27] and *2)* online [6],[8],[21],[31],[34],[38],[39],[44]. The offline approaches are trained with static datasets and once the training process is finished, the classifiers stop learning and allow no further modification to their structure. The majority of the offline approaches were developed during the time that data was not considered to be in large-scale, streaming and dynamically evolving. Nowadays, as we are living in the era of the so-called "Big Data", these approaches become less applicable. There are two types of online classification approaches, namely, *1)* incremental [31],[34],[44] and *2)* evolving [6],[8],[21],[38],[39]. Online approaches can be of "one-pass" type, which means that they are able to consistently learn from newly arrived data samples and only store the key information in memory, meanwhile, discard all the processed training samples. The evolving approaches [6],[8],[21],[38],[39], as the more advanced branch of online approaches, further address the problem of changing data pattern in nonstationary environments by continuously evolving system structure and recursively updating meta-parameters. Compared with the other types, evolving approaches are more memory- and computation- efficient and, thus, are more frequently used in real-world applications. On the other hand, the performance of the online approaches, including the evolving ones, is sensitive to the order of data samples.

Very often in real situations, a part of the data is available in a static form, while the rest is observed sequentially in a streaming form. Offline approaches ignore the fact that the data pattern may change with more data available. However, it is also unnecessary for an approach to learn online from the very beginning of the data stream because initialising the system with the available static data in an offline manner can guarantee a more robust performance.

Furthermore, many existing approaches also rely heavily on *1) prior* assumptions, which usually impose models with parameters which depend on the data generation model, i.e. Gaussian distribution [29], and *2)* user inputs, which are defined based on *prior* knowledge of the problem, i.e. radius [15],[21],[50], learning rate [38]/decay rates [39], size of the network [13],[23], etc. In real cases, such *prior* assumptions are often too strong to be held and user inputs are often hard to define due to the insufficient *prior* knowledge. In addition, in online scenarios, non-stationary data streams may also invalidate the *prior* assumptions and user inputs that were established at the initial stage.

In this paper, a new self-organising fuzzy logic (SOF) approach is proposed for classification. The SOF approach is grounded at the recently introduced Empirical Data Analytics (EDA) computational framework

*Corresponding Author

[4],[5] and the autonomous data-driven clustering techniques [19]. The SOF classifier has two training stages, *1)* offline and *2)* online. During the offline stage, it learns from the static data to establish a stable 0-order AnYa type fuzzy rule-based (FRB) system [7]. During the online training stage, the FRB system identified through the offline training process will be updated subsequently with the streaming data to follow the possible *drifts* and/or *shifts* in the data pattern. The SOF classifier only keeps the key meta-parameters in memory and is of "one-pass" type during its online training stage; therefore, it is very suitable for large-scale streaming data processing.

Most importantly, the proposed SOF classifier is nonparametric in the sense that no parameters or models are imposed for the data generation model. Employing the EDA quantities as described in section 2.2, the SOF classifier is able to objectively disclose the ensemble properties and mutual distributions of the streaming data based on the empirical observations and all the meta-parameters of the classifier are directly derived from the data without any *prior* knowledge [4],[5].

The proposed SOF classifier keeps the advantage of objectiveness of the data-driven approaches, and, at the same time, puts users "in the driving seat" by letting users to decide the level of granularity and the type of distance/dissimilarity measure for it. The idea of "granularity" is introduced and defined in [35],[36],[49]. It is well known that a problem can be approached at different levels of specificity (detail) depending on the complexity of the original problem, available computing resources, and particular needs [49]. The level of granularity in the proposed approach is aligned with this concept. However, it has to be stressed that there is no requirement for *prior* knowledge to decide the level of granularity and it can be given merely based on the preferences of the users. Higher level of granularity leads to a classifier with fine details, and at the same time, results in a risk of overfitting. A lower level of granularity, instead, gives users a classifier trained coarsely but with higher computational efficiency, generalisation and less memory requirement. The SOF classifier is always guaranteed to be meaningful due to its data-driven nature. The choice of the type of distance/dissimilarity measure further gives more freedom to the users and also makes the proposed SOF approach highly adaptive to various applications, e.g. natural language processing. In addition, the SOF classifier can also provide the default level of granularity and distance measure option for the less experienced users.

The remainder of this paper is organised as follows. The theoretical basis of the SOF classifier is summarised in section 2. Section 3 describes the offline training, online training and validation processes of the proposed approach. Section 4 presents how the level of granularity can influence the performance and efficiency of the SOF classifier. Numerical examples serving as a proof of concept are given in section 5, discussions on the convergence and local optimality of the proposed approach are also provided in the same section. Section 6 concludes this paper and gives the direction for future works.

2. Theoretical Basis

In this section, the theoretical basis of the self-organising fuzzy logic (SOF) classifier will be briefly summarised.

2.1. 0-order AnYa Fuzzy Rule-based Systems

AnYa type FRB system was introduced in [7] as an alternative approach to the widely used FRB systems of Takagi-Sugeno [45] or Mamdani [30] types. Comparing with the two predecessors, the antecedent (IF) part of AnYa type fuzzy rules is simplified to a more compact, objective and nonparametric vector form without the need of defining *ad hoc* membership functions. A 0-order AnYa type fuzzy rule has the following form:

$$IF\left(\boldsymbol{x}\sim\boldsymbol{p}_1\right)OR\left(\boldsymbol{x}\sim\boldsymbol{p}_2\right)OR...OR\left(\boldsymbol{x}\sim\boldsymbol{p}_N\right)\ \ THEN\left(class\right) \tag{1}$$

where $\boldsymbol{x}$ is the input vector; "~" denotes similarity, which can also be seen as a fuzzy degree of satisfaction/membership [7]; $\boldsymbol{p}_i$ ($i=1,2,...,N$) is the i^{th} prototype of the class; N is the number of prototypes identified from the data samples of this class. For a specific data sample, its label can be decided following different strategies, i.e. "winner-takes-all", "few-winners-take-all", "fuzzily weighted average", etc. In this paper, we use the first one, and the details are given in section 3.3.

2.2. Empirical Data Analytics Operators

As stated in section 1, the SOF classifier employs the nonparametric EDA quantities for objectively disclosing the ensemble properties and mutual distribution of the data. In this subsection, three EDA quantities, *1) cumulative proximity*, *2) unimodal density* and *3) multimodal density*, which are used in the proposed

approach will be described. Their recursive calculation forms for streaming data processing will be given as well.

First of all, let us assume a data set/stream within the real data space $\mathbf{R}^M$ (M is the dimensionality of the space) observed at the K^{th} time instance denoted by $\{\boldsymbol{x}\}_K = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_K\}$, where $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, ..., x_{i,M}] \in \mathbf{R}^M$, the subscript i denotes the time instance at which the i^{th} data sample, $\boldsymbol{x}_i$ arrived. To be more general, we assume that some data samples repeat more than once, namely, $\exists \boldsymbol{x}_i = \boldsymbol{x}_j, \quad i \neq j$. The set of sorted unique data samples is denoted as $\{\boldsymbol{u}\}_{U_K} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_{U_K}\}$ ($\boldsymbol{u}_i = [u_{i,1}, u_{i,2}, ..., u_{i,M}]$, $\{\boldsymbol{u}\}_{U_K} \subseteq \{\boldsymbol{x}\}_K$, $U_K \leq K$, U_K is the number of unique data samples) and the corresponding repeating times (frequency of occurrence) are $\{f\}_{U_K} = \{f_1, f_2, ..., f_{U_K}\}$ ($\sum_{i=1}^{U_K} f_i = K$). If no specific declaration is made, all the derivations are conducted at the K^{th} time instance as a default.

1) Cumulative Proximity

Cumulative proximity, π introduced earlier [2],[4] is derived empirically from the observed data without *prior* knowledge or *prior* assumptions, and can be seen as a square form of the farness. The *cumulative proximity* of data sample $\boldsymbol{x}_i$ is expressed as:

$$\pi_K(\boldsymbol{x}_i) = \sum_{j=1}^{K} d^2(\boldsymbol{x}_i, \boldsymbol{x}_j); \quad i = 1, 2, ..., K \tag{2}$$

where $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ denotes the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, which can be any type of distance/dissimilarity measure. It is also worth to be noticed that the average square distance between any two data samples within $\{\boldsymbol{x}\}_K$ can be expressed as: $\bar{d}_K = \frac{1}{K^2} \sum_{i=1}^{K} \pi_K(\boldsymbol{x}_i)$.

2) Unimodal Density

Unimodal density, D [4] is used as an indicator of the main data pattern within the EDA framework. The *unimodal density* at $\boldsymbol{x}_i$ is expressed as:

$$D_K(\boldsymbol{x}_i) = \frac{\sum_{l=1}^{K} \pi_K(\boldsymbol{x}_l)}{2K\pi_K(\boldsymbol{x}_i)} = \frac{\sum_{l=1}^{K}\sum_{j=1}^{K} d^2(\boldsymbol{x}_l, \boldsymbol{x}_j)}{2K\sum_{j=1}^{K} d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)}; \quad i = 1, 2, ..., K \tag{3}$$

3) Multimodal Density

Multimodal density, D^{MM} [4],[5] is estimated at the unique data sample $\boldsymbol{u}_i$ as the weighted sum of its *unimodal density* by its repeating times of occurrence expressed as:

$$D_K^{MM}(\boldsymbol{u}_i) = f_i D_K(\boldsymbol{u}_i) = f_i \frac{\sum_{l=1}^{K} \pi_K(\boldsymbol{x}_l)}{2K\pi_K(\boldsymbol{u}_i)}; \quad i = 1, 2, ..., U_K \tag{4}$$

4) Recursive Calculation Form

The recursive calculation forms of the nonparametric EDA quantities play a significant role in streaming data processing. They ensure the processing techniques to be memory- and computation- efficient. If the Euclidean distance, Mahalanobis distance, the cosine dissimilarity or some other types of distances/dissimilarity are used, one can have elegant recursive calculation forms, with which the EDA quantities can be updated in a more efficient way by keeping only the key meta-parameters in memory. In this paper, we give an example of recursive calculation expressions using Mahalanobis distance. The recursive calculation forms of the EDA quantities with other types of distance metric can be found in the previous works [4],[18].

With Mahalanobis distance used, denoted by $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)\boldsymbol{\Sigma}_K^{-1}(\boldsymbol{x}_i - \boldsymbol{x}_j)^{\mathrm{T}}}$ ($i, j = 1, 2, ..., K$), the recursive calculation expression is given as:

$$\pi_K(\boldsymbol{x}_i) = K\left((\boldsymbol{x}_i - \boldsymbol{\mu}_K)\boldsymbol{\Sigma}_K^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_K)^{\mathrm{T}} + X_K - \boldsymbol{\mu}_K\boldsymbol{\Sigma}_K^{-1}\boldsymbol{\mu}_K^{\mathrm{T}}\right) \tag{5}$$

where $\mathbf{\Sigma}_K$ is the covariance matrix, $\mathbf{\Sigma}_K = \frac{1}{K-1}\sum_{l=1}^{K}\left(\boldsymbol{x}_l - \boldsymbol{\mu}_K\right)^{\mathrm{T}}\left(\boldsymbol{x}_l - \boldsymbol{\mu}_K\right)$; $X_K = \frac{1}{K}\sum_{l=1}^{K}\boldsymbol{x}_l\mathbf{\Sigma}_K^{-1}\boldsymbol{x}_l^{\mathrm{T}}$; $\boldsymbol{\mu}_K = \frac{1}{K}\sum_{l=1}^{K}\boldsymbol{x}_l$.

The covariance matrix, $\mathbf{\Sigma}_K$ and the global mean, $\boldsymbol{\mu}_k$ can be updated recursively as:

$$\boldsymbol{\mu}_K = \frac{K-1}{K}\boldsymbol{\mu}_{K-1} + \frac{1}{K}\boldsymbol{x}_K; \quad \boldsymbol{\mu}_1 = \boldsymbol{x}_1 \tag{6}$$

$$\mathbf{X}_K = \frac{K-1}{K}\mathbf{X}_{K-1} + \frac{1}{K}\boldsymbol{x}_K^{\mathrm{T}}\boldsymbol{x}_K; \quad \mathbf{X}_1 = \boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{x}_1 \tag{7}$$

$$\mathbf{\Sigma}_K = \frac{K}{K-1}\left(\mathbf{X}_K - \boldsymbol{\mu}_K^{\mathrm{T}}\boldsymbol{\mu}_K\right) \tag{8}$$

The sum of *cumulative proximities* of all the existing data samples is given as [21]:

$$\sum_{l=1}^{K}\pi_K\left(\boldsymbol{x}_l\right) = 2K^2\left(X_K - \boldsymbol{\mu}_K\mathbf{\Sigma}_K^{-1}\boldsymbol{\mu}_K^{\mathrm{T}}\right) = 2K^2M \tag{9}$$

and, accordingly, the *unimodal density* at $\boldsymbol{x}_i$ is calculated recursively as:

$$D_K\left(\boldsymbol{x}_i\right) = \frac{2K^2\left(X_K - \boldsymbol{\mu}_K\mathbf{\Sigma}_K^{-1}\boldsymbol{\mu}_K^{\mathrm{T}}\right)}{2K\cdot K\left(\left(\boldsymbol{x}_i - \boldsymbol{\mu}_K\right)\mathbf{\Sigma}_K^{-1}\left(\boldsymbol{x}_i - \boldsymbol{\mu}_K\right)^{\mathrm{T}} + X_K - \boldsymbol{\mu}_K\mathbf{\Sigma}_K^{-1}\boldsymbol{\mu}_K^{\mathrm{T}}\right)} = \frac{1}{1+\frac{\left(\boldsymbol{x}_i - \boldsymbol{\mu}_K\right)\mathbf{\Sigma}_K^{-1}\left(\boldsymbol{x}_i - \boldsymbol{\mu}_K\right)^{\mathrm{T}}}{M}} \tag{10}$$

From the equations (6)-(10) one can see that, if the Mahalanobis distance is used, one can recursively calculate the *cumulative proximity* and *density* of new data samples by only keeping $\boldsymbol{\mu}_K$ and $\mathbf{X}_K$ in the memory.

However, we have to admit that not all kinds of distance/dissimilarity measures support such an elegant form of recursive calculation, and for these types of measure, the following general recursive calculation expressions still hold [2]:

$$\pi_K\left(\boldsymbol{x}_i\right) = \pi_{K-1}\left(\boldsymbol{x}_i\right) + d^2\left(\boldsymbol{x}_i, \boldsymbol{x}_K\right) \tag{11a}$$

$$\sum_{j=1}^{K}\pi_K\left(\boldsymbol{x}_j\right) = \sum_{j=1}^{K-1}\pi_{K-1}\left(\boldsymbol{x}_j\right) + 2\pi_K\left(\boldsymbol{x}_K\right) \tag{11b}$$

$$D_K\left(\boldsymbol{x}_i\right) = \frac{\sum_{j=1}^{K-1}\pi_{K-1}\left(\boldsymbol{x}_j\right) + 2\pi_K\left(\boldsymbol{x}_K\right)}{2K\left(\pi_{K-1}\left(\boldsymbol{x}_i\right) + d^2\left(\boldsymbol{x}_i, \boldsymbol{x}_K\right)\right)} \tag{11c}$$

Generally, Euclidean distance is the most widely used distance metric, and its effectiveness and validity as the distance measure, in most cases, are guaranteed. If the data generation model follows a Gaussian distribution or some similar distributions, Mahalanobis distance would be a good choice. While in high dimensional problems, cosine dissimilarity is free from the "curse of dimensionality" and thus, is more effective and more frequently used [1],[9],[42].

On the other hand, we have to stress that the most suitable choice of distance/dissimilarity measure is always problem-specific, and one can use the current knowledge in the problem domain to choose the desired measure for a reasonable approximation and a desired classification result. However, this is out of the range of this paper. In this paper, we only consider the general cases without using *prior* knowledge.

3. SOF classifier

In this section, the offline training, online training and validation stages of the SOF classifier will be described in detail. For a better demonstration, the architecture of the proposed approach is given in Fig. 1.

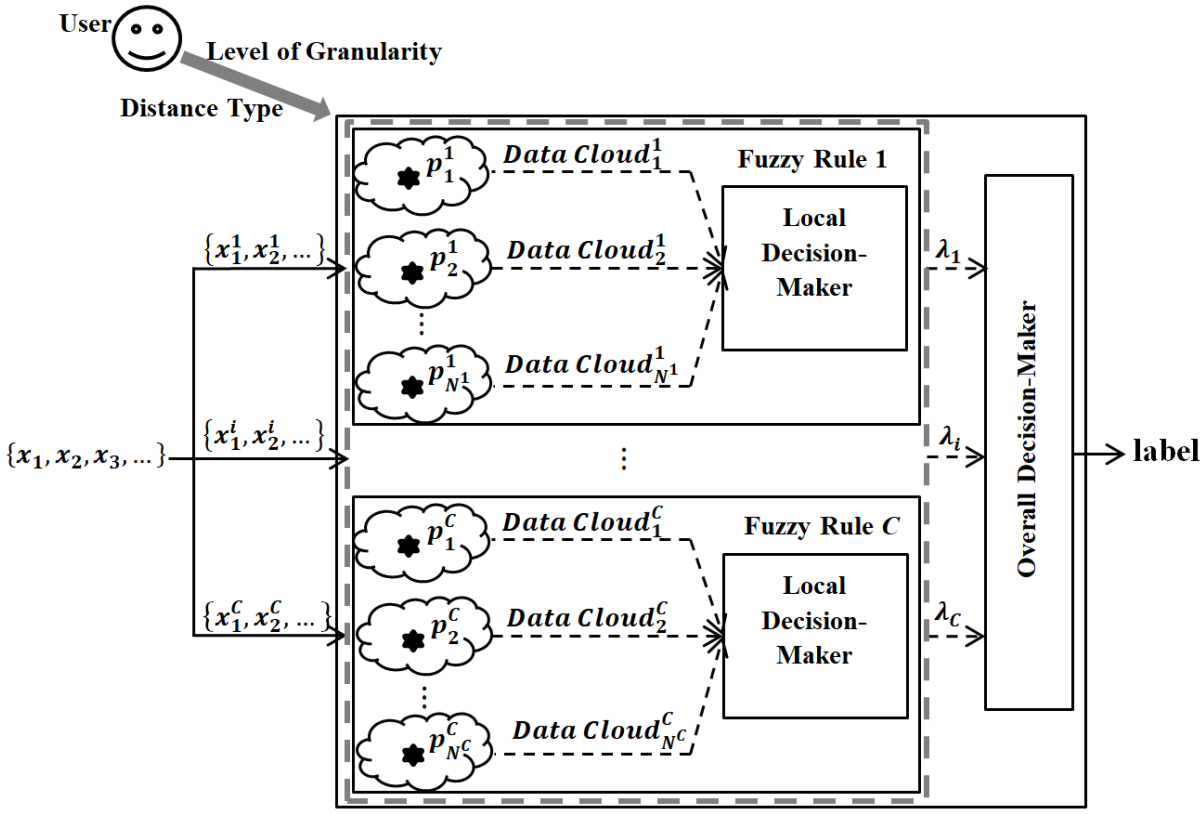


Fig. 1. Architecture of the Self-Organising Fuzzy Logic (SOF) classifier

3.1. Offline Training

The offline training process of the SOF classifier is category-wise as shown in Fig. 1, the classifier will identify prototypes from each class separately and form a 0-order AnYa type fuzzy rule based on the identified prototypes per class (in the form of equation (1)). The training processes of the fuzzy rules of different classes will not influence each other. In the rest of this subsection, we assume that the training process is conducted on data samples of the c^{th} class ($c=1,2,\ldots,C$) denoted by $\{\boldsymbol{x}\}_{K^c}^c=\{\boldsymbol{x}_1^c,\boldsymbol{x}_2^c,...,\boldsymbol{x}_{K^c}^c\}$ ($\{\boldsymbol{x}\}_{K^c}^c\subset\{\boldsymbol{x}\}_K$), and the corresponding unique data sample set and frequencies of occurrence are denoted, respectively, by $\{\boldsymbol{u}\}_{U_K^c}^c=\{\boldsymbol{u}_1^c,\boldsymbol{u}_2^c,...,\boldsymbol{u}_{U_K^c}^c\}$ and $\{f\}_{U_K^c}^c=\{f_1^c,f_2^c,...,f_{U_K^c}^c\}$, where K^c is the number of data samples with $\{\boldsymbol{x}\}_{K^c}^c$, U_K^c is the number of unique data samples of the c^{th} class. Considering all the classes, we have $\sum_{c=1}^{C}K^c=K$ and $\sum_{c=1}^{C}U_K^c=U_K$.

In the proposed approach, prototypes are identified based on the *densities* and the mutual distributions of the data samples. Firstly, *multimodal densities* $D_{K^c}^{MM}\left(\boldsymbol{u}_i^c\right)=f_i^c\dfrac{\sum_{l=1}^{K^c}\sum_{j=1}^{K^c}d^2\left(\boldsymbol{x}_l^c,\boldsymbol{x}_j^c\right)}{2K^c\sum_{j=1}^{K^c}d^2\left(\boldsymbol{u}_i^c,\boldsymbol{x}_j^c\right)}$ ($i=1,2,..,U_K^c$) [4],[5] at all the unique data samples within $\{\boldsymbol{u}\}_{U_K^c}^c$ are calculated using equation (4). Then, the data samples are ranked in a list denoted by $\{\boldsymbol{r}\}$ in terms of their mutual distances and values of *multimodal density*.

By finding out the data sample with the highest *multimodal density*, $\boldsymbol{r}_1=\arg\max_{i=1,2,...,U_K^c}\left(D_{K^c}^{MM}\left(\boldsymbol{u}_i^c\right)\right)$, the first element, $\boldsymbol{r}_1$ of the list $\{\boldsymbol{r}\}$ is identified. Then, the second element, $\boldsymbol{r}_2$ is identified as the data sample with the

minimum distance to $\boldsymbol{r}_1$: $\boldsymbol{r}_2 = \underset{i=1,2,...,U_K^c-1}{\arg\min}\left(d\left(\boldsymbol{r}_1, \boldsymbol{u}_i^c\right)\right)$. The third element of $\{\boldsymbol{r}\}$ denoted by $\boldsymbol{r}_3$ is identified based on the minimum distance to $\boldsymbol{r}_2$. By repeating the process and until all the data samples have been selected, the full list $\{\boldsymbol{r}\}$ is built and the *multimodal densities* of $\{\boldsymbol{u}\}_{U_K^c}^c$ are ranked accordingly with the list, denoted by $\left\{D_{K^c}^{MM}\left(\boldsymbol{r}\right)\right\}$ [19]. It needs to be stressed that once a data sample is selected into $\{\boldsymbol{r}\}$, it cannot be selected for a second time.

Prototypes, denoted by $\{\boldsymbol{p}\}_0$, are then identified as the local maxima of the ranked *multimodal densities*, $\left\{D_{K^c}^{MM}\left(\boldsymbol{r}\right)\right\}$ using **Condition 1** [19]:

$$\textbf{Condition 1}: \; IF\left(D_{K^c}^{MM}\left(\boldsymbol{r}_i\right) > D_{K^c}^{MM}\left(\boldsymbol{r}_{i+1}\right)\right) AND \left(D_{K^c}^{MM}\left(\boldsymbol{r}_i\right) > D_{K^c}^{MM}\left(\boldsymbol{r}_{i-1}\right)\right) \;\; THEN\left(\boldsymbol{r}_i \in \{\boldsymbol{p}\}_0\right) \tag{12}$$

Once all the prototypes are identified using equation (12), one may notice some less representative ones within $\{\boldsymbol{p}\}_0$, therefore, it is necessary to conduct a filtering operation to remove them from $\{\boldsymbol{p}\}_0$.

Before the filtering operation starts, we firstly use the prototypes to attract nearby data samples to form data clouds [7] resembling Voronoi tessellation [32]:

$$winning\ prototype = \underset{\boldsymbol{p}\in\{\boldsymbol{p}\}_0}{\arg\min}\left(d\left(\boldsymbol{x}_i, \boldsymbol{p}\right)\right); \quad \boldsymbol{x}_i \in \{\boldsymbol{x}\}_{K^c}^c \tag{13}$$

After all the data clouds are formed around the existing prototypes $\{\boldsymbol{p}\}_0$, one can obtain the centres of the data clouds denoted by $\{\boldsymbol{\varphi}\}_0$ and the *multimodal densities* at the centres are calculated using equation (4) as $D_{K^c}^{MM}\left(\boldsymbol{\varphi}_i\right) = S_i D_{K^c}\left(\boldsymbol{\varphi}_i\right)$, where $\boldsymbol{\varphi}_i \in \{\boldsymbol{\varphi}\}_0$; S_i is the support (number of members) of the i^{th} data cloud.

Then, for each data cloud, assuming the i^{th} one ($\boldsymbol{\varphi}_i \in \{\boldsymbol{\varphi}\}_0$), the collection of the centres of its neighbouring data clouds, denoted by $\{\boldsymbol{\varphi}\}_i^{neighbouring}$ are identified using the following principle:

$$\textbf{Condition 2}: \; IF\left(d^2\left(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j\right) \le G_{K^c}^{c,L}\right) \;\; THEN\left(\boldsymbol{\varphi}_j \in \{\boldsymbol{\varphi}\}_i^{neighbouring}\right) \tag{14}$$

where $\boldsymbol{\varphi}_j \in \{\boldsymbol{\varphi}\}_0$, $\boldsymbol{\varphi}_j \neq \boldsymbol{\varphi}_i$; $G_{K^c}^{c,L}$ is defined as the average radius of local influential area around each data sample, which is corresponding to the L^{th} ($L = 1, 2, 3, ...$) level of granularity and is derived from the data of the c^{th} class based on the users' choice in an offline way. Section 4 will explain how to derive $G_{K^c}^{c,L}$ in detail.

Finally, the most representative prototypes of the c^{th} class, denoted by $\{\boldsymbol{p}\}^c$, are selected out from the centres of the existing data clouds satisfying **Condition 3** [19]:

$$\textbf{Condition 3}: IF\left(D_{K^c}^{MM}\left(\boldsymbol{\varphi}_i\right) > \max_{\boldsymbol{\varphi}\in\{\boldsymbol{\varphi}\}_i^{neighbouring}}\left(D_{K^c}^{MM}\left(\boldsymbol{\varphi}\right)\right)\right) \;\; THEN\left(\boldsymbol{\varphi}_i \in \{\boldsymbol{p}\}^c\right) \tag{15}$$

After all the representative prototypes of the c^{th} class $\{\boldsymbol{p}\}^c$ are identified, one can build the AnYa type fuzzy rule in the following form, where N^c is the number of prototypes in $\{\boldsymbol{p}\}^c$.

$$IF\left(\boldsymbol{x} \sim \boldsymbol{p}_1^c\right) OR \left(\boldsymbol{x} \sim \boldsymbol{p}_2^c\right) OR ... OR \left(\boldsymbol{x} \sim \boldsymbol{p}_{N^c}^c\right) \;\; THEN\left(class\, c\right) \tag{16}$$

The main procedure of the offline training process of the proposed SOF classifier is summarised in in the following pseudo code.

Offline training process of the SOF classifier

i. Calculate D^{MM} at $\{\boldsymbol{u}\}_{U_K^c}^c$;

ii. Find $\boldsymbol{r}_1 = \underset{i=1,2,...,U_K^c}{\arg\max}\left(D_{K^c}^{MM}\left(\boldsymbol{u}_i^c\right)\right)$ and exclude $\boldsymbol{r}_1$ from $\{\boldsymbol{u}\}_{U_K^c}^c$;

iii. $k \leftarrow 1$; $\{\boldsymbol{r}\} \leftarrow \boldsymbol{r}_1$; $\left\{D_{K^c}^{MM}\left(\boldsymbol{r}\right)\right\} \leftarrow D_{K^c}^{MM}\left(\boldsymbol{r}_1\right)$;;

iv. **While** $U_K^c - k \neq 0$

* $k \leftarrow k+1$;

* Find $\boldsymbol{r}_k = \arg\min_{\boldsymbol{u}_i^c \in \{\boldsymbol{u}\}_{U_K^c}^c} \left(d\left(\boldsymbol{r}_{k-1}, \boldsymbol{u}_i^c\right)\right)$ and exclude $\boldsymbol{r}_k$ from $\{\boldsymbol{u}\}_{U_K^c}^c$;

* $\{\boldsymbol{r}\} \leftarrow \{\boldsymbol{r}\} + \boldsymbol{r}_k$; $\left\{D_{K^c}^{MM}(\boldsymbol{r})\right\} \leftarrow \left\{D_{K^c}^{MM}(\boldsymbol{r})\right\} + D_{K^c}^{MM}(\boldsymbol{r}_k)$;

v. **End While**

vi. Identify $\{\boldsymbol{p}\}_0$ using **Condition 1**;

vii. Form data clouds around $\{\boldsymbol{p}\}_0$;

viii. Identify $\{\boldsymbol{\varphi}\}_0$ from the data clouds;

ix. Calculate D^{MM} at $\{\boldsymbol{\varphi}\}_0$;

x. Identify $\{\boldsymbol{\varphi}\}^{neighbouring}$ using **Condition 2**;

xi. Identify $\{\boldsymbol{p}\}^c$ using **Condition 3**;

xii. Create the c^{th} fuzzy rule with $\{\boldsymbol{p}\}^c$.

3.2. Online Self-Evolving Training

During the online training stage, the SOF classifier continues to update its system parameters and structure with the streaming data on a sample-by-sample basis. Furthermore, because the EDA quantities employed by the SOF classifier can be updated recursively, it can be of "one-pass" type, and its computation- and memory-efficiency is also guaranteed. In this subsection, we assume that the training process of the SOF classifier with the static dataset $\{\boldsymbol{x}\}_K$ has been finished and new data samples start to arrive in a data stream form. Similar to the offline training stage, during the online training stage, the fuzzy rules of different classes are updated separately. During the online stage, recursive calculation expressions of the EDA quantities with Mahalanobis distance are used. Nonetheless, we want to stress again that the proposed approach can use various types of distance/dissimilarity measures (see subsection 2.2).

Assuming at K+1$^{\text{th}}$ instance, a new data sample of the c^{th} class, denoted as $\boldsymbol{x}_{K^c+1}^c$, arrives, the SOF classifier, firstly, updates the meta-parameters $\boldsymbol{\mu}_{K^c}^c$, $\mathbf{X}_{K^c}^c$, $\boldsymbol{\Sigma}_{K^c}^c$ to $\boldsymbol{\mu}_{K^c+1}^c$, $\mathbf{X}_{K^c+1}^c$, $\boldsymbol{\Sigma}_{K^c+1}^c$ using equations (6)-(8). The average radius of local areas of influence, $G_{K^c}^{c,L}$ is updated afterwards in a recursive way based on the ratio between $\bar{d}_{K^c}^c$ and $G_{K^c}^{c,L}$:

$$G_{K^c+1}^{c,L} = \frac{\bar{d}_{K^c+1}^c}{\bar{d}_{K^c}^c} G_{K^c}^{c,L} = \frac{\frac{1}{\left(K^c+1\right)^2}\sum_{l=1}^{K^c+1}\pi_{K^c+1}\left(\boldsymbol{x}_l^c\right)}{\frac{1}{\left(K^c\right)^2}\sum_{l=1}^{K^c}\pi_{K^c}\left(\boldsymbol{x}_l^c\right)} G_{K^c}^{c,L} \tag{17}$$

where $\bar{d}_{K^c}^c$ and $\bar{d}_{K^c+1}^c$ denote the average square distances between any two data samples within $\{\boldsymbol{x}\}_{K^c}^c$ and $\{\boldsymbol{x}\}_{K^c+1}^c$, respectively.

As a special case, for Mahalanobis distance (equation (9)), $G_{K^c+1}^{c,L} = G_{K^c}^{c,L}$. As we can see from equation (17), instead of deriving $G_{K^c+1}^{c,L}$ in an offline way, which will be described in section 4 in detail, equation (17) largely reduces the computational complexity and memory requirement, and further largely improves the efficiency of the SOF classifier.

Then, $\boldsymbol{x}_{K^c+1}^c$ is checked by the following condition to evaluate its potential to be a new prototype [2],[8]:

Condition 4: $$IF\left(D_{K^c+1}\left(\boldsymbol{x}^c_{K^c+1}\right) > \max_{\boldsymbol{p}\in\{\boldsymbol{p}\}^c}\left(D_{K^c+1}(\boldsymbol{p})\right)\right) OR \left(D_{K^c+1}\left(\boldsymbol{x}^c_{K^c+1}\right) < \min_{\boldsymbol{p}\in\{\boldsymbol{p}\}^c}\left(D_{K^c+1}(\boldsymbol{p})\right)\right) \quad THEN\left(\boldsymbol{x}^c_{K^c+1}\in\{\boldsymbol{p}\}^c\right) \tag{18}$$

where equation (10) is used for calculating $D_{K^c+1}\left(\boldsymbol{x}^c_{K^c+1}\right)$ and $D_{K^c+1}(\boldsymbol{p})$ ($\boldsymbol{p}\in\{\boldsymbol{p}\}^c$). If $\boldsymbol{x}^c_{K^c+1}$ meets **Condition 4**, a new prototype is added to the fuzzy rule of the c^{th} class (equation (16)) and the meta-parameters of the SOF classifier are updated as follows:

$$N^c \leftarrow N^c+1;\quad \boldsymbol{p}^c_{N^c} \leftarrow \boldsymbol{x}^c_{K^c+1};\quad S^c_{N^c}\leftarrow 1;\quad \{\boldsymbol{p}\}^c \leftarrow \{\boldsymbol{p}\}^c + \boldsymbol{p}^c_{N^c} \tag{19}$$

If **Condition 4** is unsatisfied, we still need to check whether $\boldsymbol{x}^c_{K^c+1}$ is very close to an existing prototype by using **Condition 5** [19].

Condition 5: $$IF\left(\min_{\boldsymbol{p}\in\{\boldsymbol{p}\}^c}\left(d^2\left(\boldsymbol{x}^c_{K^c+1},\boldsymbol{p}\right)\right) > G^{c,L}_{K^c+1}\right)\quad THEN\left(\boldsymbol{x}^c_{K^c+1}\in\{\boldsymbol{p}\}^c\right) \tag{20}$$

If **Condition 5** is met, a new prototype is added to the fuzzy rule of the c^{th} class ($\{\boldsymbol{p}\}^c \leftarrow \{\boldsymbol{p}\}^c + \boldsymbol{p}^c_{N^c}$) and the corresponding new data cloud with meta-parameters initialised by equation (19) is added to the SOF classifier.

If **Conditions 4 and 5** are both unsatisfied, $\boldsymbol{x}^c_{K^c+1}$ is assigned to the nearest prototype $\boldsymbol{p}^c_{n^*} = \arg\min_{\boldsymbol{p}\in\{\boldsymbol{p}\}^c}\left(d\left(\boldsymbol{x}^c_{K^c+1},\boldsymbol{p}\right)\right)$ and the meta-parameters of the corresponding data cloud are updated as follows [2]:

$$\boldsymbol{p}^c_{n^*} \leftarrow \frac{S^c_{n^*}}{S^c_{n^*}+1}\boldsymbol{p}^c_{n^*} + \frac{1}{S^c_{n^*}+1}\boldsymbol{x}^c_{K^c+1};\quad S^c_{n^*}\leftarrow S^c_{n^*}+1 \tag{21}$$

After the meta-parameters of the classifier are updated, the AnYa type fuzzy rule (equation (16)) will be updated accordingly and the SOF classifier is ready for processing the next data sample or conducting classification.

The main procedure of the online training process of the proposed SOF classifier is summarised in the following pseudo code.

Online training process of the SOF classifier

While a new data sample of the c^{th} class $\boldsymbol{x}^c_{K^c+1}$ is available (or until interrupted)

i. Update $\boldsymbol{\mu}^c_{K^c}$, $\mathbf{X}^c_{K^c}$, $\boldsymbol{\Sigma}^c_{K^c}$, $G^{c,L}_{K^c}$ to $\boldsymbol{\mu}^c_{K^c+1}$, $\mathbf{X}^c_{K^c+1}$, $\boldsymbol{\Sigma}^c_{K^c+1}$, $G^{c,L}_{K^c+1}$;

ii. Calculate D at $\boldsymbol{x}^c_{K^c+1}$ and $\{\boldsymbol{p}\}^c$;

iii. **If** (**Condition 4** is met) **Or** (**Condition 5** is met) **Then**

* $N^c \leftarrow N^c+1;\quad \boldsymbol{p}^c_{N^c} \leftarrow \boldsymbol{x}^c_{K^c+1};\quad S^c_{N^c}\leftarrow 1;\quad \{\boldsymbol{p}\}^c \leftarrow \{\boldsymbol{p}\}^c + \boldsymbol{p}^c_{N^c}$

iv. **Else**

* Find $\boldsymbol{p}^c_{n^*}$;

* $\boldsymbol{p}^c_{n^*} \leftarrow \frac{S^c_{n^*}}{S^c_{n^*}+1}\boldsymbol{p}^c_{n^*} + \frac{1}{S^c_{n^*}+1}\boldsymbol{x}^c_{K^c+1};\quad S^c_{n^*}\leftarrow S^c_{n^*}+1;$

v. **End If**

vi. $K^c \leftarrow K^c+1;$

vii. Update the fuzzy rule;

End While

3.3. Validation

In this subsection, the procedure of the SOF classifier for decision-making is described. As it is shown in Fig. 1, during the validation stage, for a particular testing data sample, denoted by $\boldsymbol{x}$, each AnYa type fuzzy rule will have a firing strength given by the local decision-maker, denoted by $\lambda^c(\boldsymbol{x})$ ($c=1,2,...,C$), which is determined as follows:

$$\lambda^c(\boldsymbol{x}) = \max_{\boldsymbol{p}\in\{\boldsymbol{p}\}^c}\left(e^{-d^2(\boldsymbol{x},\boldsymbol{p})}\right); \quad c=1,2,...,C \tag{22}$$

Based on the C firing strengths of the C fuzzy rules correspondingly (one per rule), the label of $\boldsymbol{x}$ is decided by the overall decision-maker using the "winner-takes-all" principle as follows:

$$label = \arg\max_{c=1,2,...,C}\left(\lambda^c(\boldsymbol{x})\right) \tag{23}$$

4. Classification under Different Levels of Granularity

Since the SOF classifier is a prototype-based approach, it is of paramount importance to define a suitable local area of influence for each prototype in order to increase the descriptive ability of the fuzzy rules and at the same time, avoid overlap. There are two commonly adopted ways to define this. The first one is to define a radius based on *prior* knowledge [15]. The second one is to derive it from data following hard-coded principles [29],[38]. However, in most cases, *prior* knowledge is unavailable, while the hard-coded principles are too sensitive to the nature of the data. The performance of the two approaches is often not guaranteed. In the following part of this section, we will demonstrate how to define the local areas around prototypes based on the data and the level of granularity.

Under the 1st level of granularity (L=1), the average radius of local influential area around each prototype of the c^{th} class, denoted by $G_{K^c}^{c,1}$, is defined as follows:

$$G_{K^c}^{c,1} = \frac{\sum_{\boldsymbol{x},\boldsymbol{y}\in\{\boldsymbol{x}\}_{K^c}^c,\ \boldsymbol{x}\neq\boldsymbol{y}, d^2(\boldsymbol{x},\boldsymbol{y})\leq\bar{d}_{K^c}^c} d^2(\boldsymbol{x},\boldsymbol{y})}{Q_{K^c}^{c,1}} \tag{24}$$

where $Q_{K^c}^{c,1}$ is the number of the pairs of data samples within $\{\boldsymbol{x}\}_{K^c}^c$ between which the distance is smaller than the average distance, $\bar{d}_{K^c}^c$.

From level 2 to an arbitrary level of granularity ($L=2,3,...,$), one can calculate the average radius iteratively using the following equation:

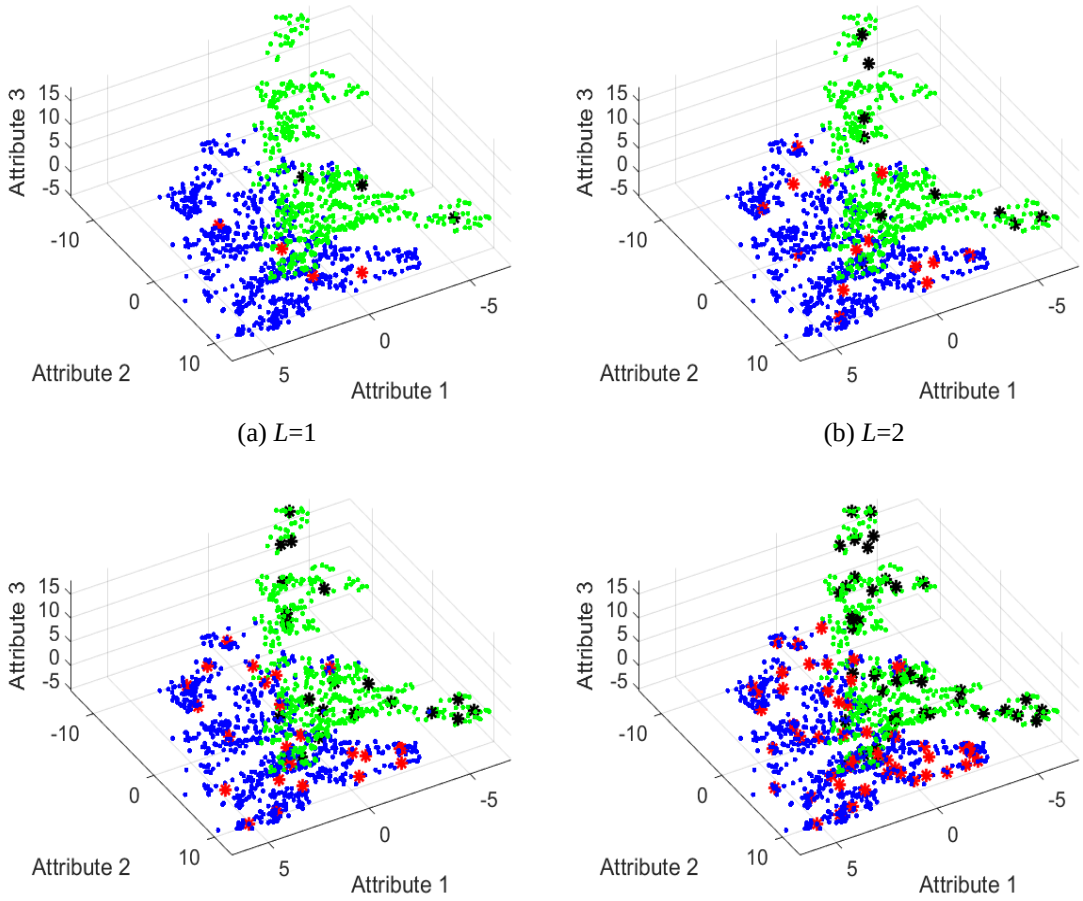
$$G_{K^c}^{c,L} = \frac{\sum_{\boldsymbol{x},\boldsymbol{y}\in\{\boldsymbol{x}\}_{K^c}^c,\ \boldsymbol{x}\neq\boldsymbol{y}, d^2(\boldsymbol{x},\boldsymbol{y})\leq G_{K^c}^{c,L-1}} d^2(\boldsymbol{x},\boldsymbol{y})}{Q_{K^c}^{c,L}} \tag{25}$$

where $G_{K^c}^{c,L-1}$ is the average radius corresponding to (L-1)th level of granularity; $Q_{K^c}^{c,L}$ is the number of the pairs of data samples between which the distance is smaller than $G_{K^c}^{c,L-1}$.

Compared with the traditional approaches, there are strong advantages in deriving local information in this way. Firstly, $G_{K^c}^{c,L}$ is guaranteed to be valid all the time. Defining the threshold or hard-coded mathematical principles in advance may suffer from various problems, which have been discussed at the beginning of this section. While $G_{K^c}^{c,L}$ is derived from the data directly and is always meaningful. There is no need for *prior* knowledge of data sets/streams, and the level of granularity used by the SOF classifier can be decided merely based on the preferences of the users. Moreover, users are allowed to have freedom to make choices, but at the same time, are not overloaded. Finally, one can always adapt the classifier by changing the level of granularity based on the specific needs. Some problems rely heavily on fine details, while others may need generality only.

In general, the higher level of granularity is chosen, the more fine details (more prototypes) the SOF classifier extracts from the data, and the classifier achieves a higher performance. At the same time, the SOF classifier may consume more computational and memory resources, and overfitting may also appear. On the contrary, with low level of granularity, the SOF classifier only learns the coarse information from training. Although, the classifier will be more computationally efficient, its performance may be influenced due to the loss of fine information from the data. An illustrative example based on the UCI benchmark dataset named Banknote Authentication[1] is given in Fig. 2 with different levels of granularity. In this visual example, the SOF classifier is trained offline using Mahalanobis distance.

[1] available at: https://archive.ics.uci.edu/ml/datasets/banknote+authentication



(a) L=1 (b) L=2

(c) L=3 (d) L=4

Fig. 2. Prototypes identified under different levels of granularity based on Banknote Authentication dataset (dots and asterisks in different colours denote data samples and prototypes of different classes).

5. Numerical Examples and Discussions

In this section, numerical examples are provided as a proof of concept. The experiments are conducted using MATLAB R2017a on a PC within Windows 10 operating system, 3.6 GHz dual core Intel 7 processor and 16 GB RAM. The links to the source codes (MATLAB and Python versions) of the proposed approach can be found at: http://www.empiricaldataanalytics.org/downloads.html.

We, firstly, consider the following challenging UCI benchmark datasets for evaluating the performance the SOF classifier:

1) Occupancy detection dataset[2];

2) Optical recognition of handwritten digits dataset[3];

3) Multiple features dataset[4], and

4) Letter recognition dataset[5].

The details of the four benchmark datasets are tabulated in Table 1. The occupancy detection dataset contains one training set with 8143 data samples and two testing sets with 2665 and 9752 data samples in each, respectively. In this paper, we combine the two testing sets into one for clarity, and the time stamp of this dataset has been removed. The optical recognition dataset consists of one training set with 3823 data samples and one testing set with 1797 data samples.

[2] available at: https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+

[3] available at: https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

[4] available at: https://archive.ics.uci.edu/ml/datasets/Multiple+Features

[5] available at: https://archive.ics.uci.edu/ml/datasets/Letter+Recognition

Table 1 Details of the benchmark datasets used for numerical example

Dataset		Number of Classes	Number of Samples	Number of Attributes
Occupancy detection	Training set	2	8143	5+1 label
	Testing set		12417	
Optical recognition	Training set	10	3823	64+1 label
	Testing set		1797	
Letter recognition		26	20000	16+1 label
Multiple features		10	2000	649+1 label

Firstly, the influence of different levels of granularity on the classification results of the proposed SOF approach is studied, and the occupancy detection and optical recognition datasets are used in this experiment. In this example, we consider the offline scenario only and vary the level of granularity, L from 1 to 12. The classification results are tabulated in Table 2 and the performance is measured in terms of classification accuracy, denoted by *Acc*, the number of identified prototypes, denoted by N ($N = \sum_{c=1}^{C} N^c$) and the training time consumption in seconds, denoted by *t*. Here, the Mahalanobis distance, Euclidean distance and cosine dissimilarity are used. However, for the optical recognition dataset, as the co-variance matrix of the data is not always positive definite, we consider the results obtained using the Euclidean distance and cosine dissimilarity only. The results tabulated are the average of 10 Monte Carlo experiments by randomly descrambling the order of the training samples.

From Table 2 one can see that, in general, the higher level of granularity is chosen, the higher accuracy the SOF classifier can exhibit during classification, but the more prototypes the classifier identifies, which can lower down the computation- and memory-efficiency. It is worth to notice that the proposed approach produced the same result in 10 Monte Carlo experiments, which demonstrates that the SOF classifier is invariant to the changes in the order of data samples during the offline training.

One may also notice from Table 2 that the type of distance/dissimilarity measure used also influences the performance of the proposed approach. As the proposed approach accommodates various types of distance/dissimilarity measures, one can use the current knowledge of the problem domain to choose the appropriate distance measure.

Secondly, the classification performance of the proposed SOF classifier with different amounts of offline training samples is investigated. In this example, we use the letter recognition and multiple features datasets. As the two datasets are both highly complex, we choose the 12^{th} level of granularity to ensure the SOF classifier can learn sufficient details. The percentage of offline training samples is changed from 10% to 50% and the classification is conducted on the remaining 50% of the data in an offline scenario. The results are tabulated in Table 3, which are the averages of 10 Monte Carlo experiments by randomly selecting the training set and testing set. The corresponding average training time consumption (in seconds) is depicted in Fig. 3.

In order to investigate the performance of the proposed approach in an online scenario, we conduct an extra experiment on the two datasets. The SOF classifier is firstly trained with 15% of the data samples in an offline scenario, and then, is trained in an online scenario by using different amounts (from 5% to 35%) of data samples on a sample-by-sample basis. The classification accuracy of SOF classifier is evaluated on the remaining 50% of data samples. The average performance is tabulated in Table 4 after 10 Monte Carlo experiments by randomly selecting the offline training set, online training set and testing set. The corresponding average time consumption per data sample (in milliseconds) during the online training process is given in Fig. 4. In both Tables 3 and 4, the classification results on the multiple feature dataset using the Mahalanobis distance is not given for the same reason mentioned before.

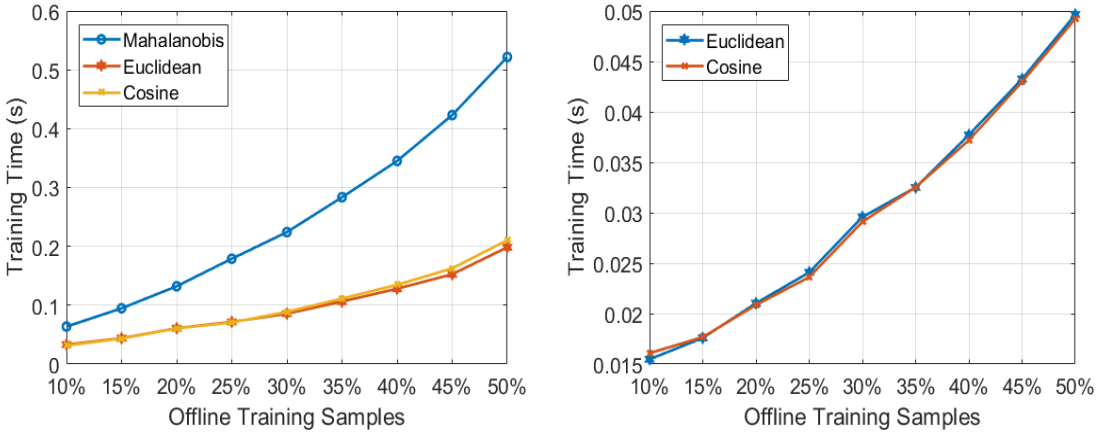
From Table 3 one can conclude that the more data samples the SOF classifier is provided with during the offline training stage, the better performance it can exhibit in the classification stage. Table 4 shows that the performance of the SOF classifier can be further improved through the online update with more training data samples after the offline training, which is one of the very strong advantages of the proposed approach. In real applications, new data is more often coming in the form of a data stream, which may exhibit *shifts* and/or *drifts* in the data pattern [28]. With the ability of self-evolving online learning, the SOF classifier is able to continuously follow the changing data pattern without full retraining, which largely enhances the efficiency and

saves the computational resources. Fig. 4 demonstrates the very high computational efficiency (less than 0.3 millisecond per data sample) for the SOF classifier to self-evolve recursively on a sample-by-sample basis.

Table 2 Influence of granularity on classification performance

Dataset	Distance	Measures	*L*					
			1	2	3	4	5	6
Occupancy detection	Mahalanobis	*Acc*	0.8942	0.8920	0.9038	0.9426	0.9494	0.9532
		N	14	31	55	116	217	339
		t	2.80	2.97	3.08	3.11	3.16	3.14
	Euclidean	*Acc*	0.8107	0.8403	0.8618	0.9112	0.9382	0.9513
		N	16	46	77	137	201	281
		t	2.15	2.31	2.47	2.55	2.59	2.65
	Cosine	*Acc*	0.8109	0.8161	0.8877	0.9261	0.9481	0.9519
		N	12	43	72	108	167	217
		t	2.13	2.31	2.55	2.56	2.63	2.72
Optical recognition	Euclidean	*Acc*	0.9160	0.9421	0.9499	0.9716	0.9766	0.9761
		N	25	48	105	214	409	643
		t	0.09	0.10	0.10	0.09	0.10	0.10
	Cosine	*Acc*	0.9087	0.9421	0.9588	0.9649	0.9699	0.9733
		N	25	50	116	238	417	655
		t	0.09	0.10	0.09	0.09	0.10	0.10

Dataset	Distance	Measures	*L*					
			7	8	9	10	11	12
Occupancy detection	Mahalanobis	*Acc*	0.9539	0.9543	0.9543	0.9543	0.9543	0.9543
		N	549	786	1029	1279	1433	1512
		t	3.33	3.16	3.26	3.29	3.36	3.32
	Euclidean	*Acc*	0.9564	0.9579	0.9584	0.9588	0.9588	0.9588
		N	395	525	663	783	939	1094
		t	2.72	2.68	2.69	2.68	2.74	2.70
	Cosine	*Acc*	0.9558	0.9557	0.9559	0.9559	0.9559	0.9559
		N	288	388	507	650	825	1007
		t	2.78	2.68	2.75	2.70	2.79	2.77
Optical recognition	Euclidean	*Acc*	0.9811	0.9833	0.9833	0.9833	0.9839	0.9839
		N	840	950	1012	1034	1046	1048
		t	0.10	0.10	0.10	0.11	0.11	0.11
	Cosine	*Acc*	0.9755	0.9761	0.9761	0.9761	0.9761	0.9761
		N	843	960	1013	1039	1039	1046
		t	0.11	0.11	0.11	0.11	0.11	0.11

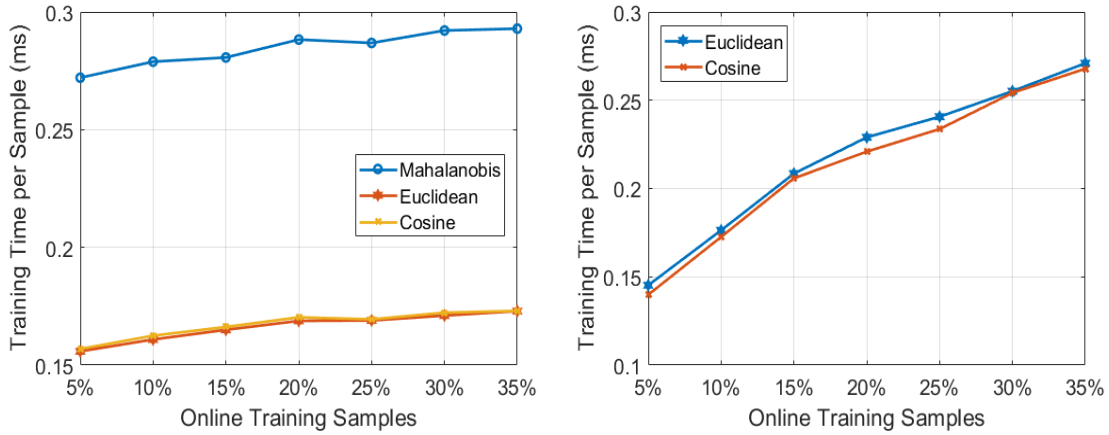


(a) Letter recognition (b) Multiple features

Fig. 3. The average training time consumption with different amounts of training samples

Table 3. Classification performance (in accuracy) with different amount of data for offline training

Dataset	Distance	Percentage for Offline Training								
		10%	15%	20%	25%	30%	35%	40%	45%	50%
Letter recognition	Mahanobis	0.8375	0.8689	0.8878	0.8983	0.9079	0.9162	0.9217	0.9241	0.9265
	Euclidean	0.7924	0.8415	0.8703	0.8863	0.9013	0.9082	0.9185	0.9244	0.9298
	Cosine	0.8013	0.8480	0.8731	0.8904	0.9026	0.9109	0.9197	0.9253	0.9296
Multiple features	Euclidean	0.8415	0.8664	0.8854	0.8924	0.9026	0.9076	0.9144	0.9203	0.9267
	Cosine	0.8703	0.8895	0.9025	0.9125	0.9194	0.9263	0.9269	0.9276	0.9366



(a) Letter recognition (b) Multiple features

Fig. 4. The average training time consumption per sample during the online training

Table 4. Classification performance (accuracy) with different amount of data for online training following the offline training with 15% of the data

Dataset	Distance	Percentage for Online Training						
		5%	10%	15%	20%	25%	30%	35%
Letter recognition	Mahanobis	0.8594	0.8836	0.9012	0.9125	0.9199	0.9279	0.9327
	Euclidean	0.8738	0.8910	0.9062	0.9162	0.9231	0.9303	0.9352
	Cosine	0.8758	0.8931	0.9070	0.9158	0.9233	0.9293	0.9350
Multiple features	Euclidean	0.8827	0.9097	0.9166	0.9205	0.9272	0.9340	0.9352
	Cosine	0.9062	0.9258	0.9335	0.9316	0.9318	0.9399	0.9409

To further evaluate the performance of the SOF classifier with $L=12$, we compare it with a number of "state-of-the-art" approaches in an offline scenario on the four benchmark datasets tabulated in Table 1:

1) Support vector machine (SVM) classifier [14];

2) K-nearest neighbour (KNN) classifier [16];

3) Decision tree (DT) classifier [40];

4) Self-organising map (SOM) classifier [37];

6) DENFIS classifier [22];

7) eClass-0 classifier [8], and

8) TEDAClass classifier [21].

During the comparison, the SVM classifier uses a linear kernel; for the KNN classifier, k is equal to 10; SOM classifier applies "winner-takes-all" principle with a net size of 9×9. As one may obtain the covariance matrices that are not positive definite from the optical recognition and multiple feature datasets, we only use the Euclidean distance and cosine dissimilarity for these two datasets during the comparison. For letter recognition and multiple features datasets, we use 50% of the data for training and the rest for testing. The performance comparison is tabulated in Table 5, where the highest classification accuracy for each benchmark problem is bolded. The reported results are the averages of 10 Monte Carlo experiments. In the experiments, the DENFIS classifier failed in both the optical recognition and multiple feature datasets because of the high dimensionality.

From Table 5 one can see that, the proposed SOF classifier can exhibit very high performance on the four benchmark problems with a very short training process.

Table 5. Performance comparison

Dataset	Approach	*Acc*	*t* (s)	Dataset	Approach	*Acc*	*t* (s)
Occupancy detection	SOF-Mahalanobis	0.9543	3.32	Letter recognition	SOF-Mahalanobis	0.9265	0.52
	SOF-Euclidean	0.9588	2.70		SOF-Euclidean	**0.9298**	0.20
	SOF-Cosine	0.9559	2.77		SOF-Cosine	0.9296	0.21
	SVM	0.9577	103.62		SVM	0.8533	16.16
	KNN	**0.9664**	0.11		KNN	0.9180	0.05
	DT	0.9314	0.10		DT	0.8243	0.10
	SOM	0.9512	9.40		SOM	0.5363	12.85
	DENFIS	0.8909	14.28		DENFIS	0.3256	95.36
	eClass-0	0.8863	0.72		eClass-0	0.5125	0.74
	Simpl_eClass0	0.9096	0.49		Simpl_eClass0	0.5853	1.09
	TEDAClass	0.9634	416.50		TEDAClass	0.5154	2335.71
Optical recognition	SOF-Euclidean	**0.9839**	0.11	Multiple features	SOF-Euclidean	0.9267	0.05
	SOF-Cosine	0.9761	0.11		SOF-Cosine	0.9366	0.05
	SVM	0.9627	1.49		SVM	**0.9671**	15.97
	KNN	0.9766	0.08		KNN	0.9151	0.02
	DT	0.8525	0.11		DT	0.9244	0.16
	SOM	0.9577	12.19		SOM	0.8746	29.19
	DENFIS	No Valid Result			DENFIS	No Valid Result	
	eClass-0	0.8681	0.69		eClass-0	0.8264	1.59
	Simpl_eClass0	0.8883	1.51		Simpl_eClass0	0.8201	3.30
	TEDAClass	0.9120	1649.17		TEDAClass	0.8637	14011.87

In order to see the performance of the proposed SOF classifier on high-dimensional, complex problems, we further involve the following two benchmark image classification problems:

1) MNIST dataset[6], and

2) Singapore dataset[7].

MNIST dataset is a large-scale image set for hand-written digits recognition (from "0" to "9"). The training set contains 60000 images and the testing set contains 10000 images (with image size of 28×28 pixels). We use the GIST feature descriptor [33] to extract 1×512 dimensional feature vectors from the central area (22×22 pixels) of handwritten digit images [3]. Singapore dataset was constructed from a large high-resolution satellite image of Singapore. This dataset consists of 1086 remote sensing scene images (with original size of 256×256 pixels) of nine classes ("airplane", "forest", "harbour", "industry", "meadow", "overpass", "residential", "river" and "runway"). We use the 1×4096 dimensional activations of the pre-trained VGG-VD-16 convolution neural network [43] from the first fully connected layer [47] as the feature vectors of the images. The details of two large-scale benchmark problems are tabulated in Table 6. Examples of the images of both datasets are given in Fig. 5. In the following examples, the SOF classifier only uses the Euclidean distance and the cosine dissimilarity, and the 12^{th} level of granularity ($L=12$) for both problems.

For the MNIST dataset, the SOF classifier is trained in an offline scenario using the training sets with different sizes (5000, 10000, 20000, 30000, 40000, 50000 and 60000 images), and then, is tested on the testing set, and the classification results are tabulated in Table 7. Experiments in the online scenario are also conducted by firstly training the SOF classifier with 5000 images in an offline scenario, and then continuing the training in an online self-evolving manner; the performance with different amounts of training images is tabulated in Table 7 as well. All the results reported in Table 7 are average results of 10 Monte Carlo experiments. The average time consumption of the proposed approach is given in Fig. 6. For the offline scenario, we report the overall time consumption; for the online scenario, we report the time consumption for the SOF classifier to process each image. We also involve the following approaches in the comparison:

1) Support vector machine (SVM) classifier;

[6] available at: http://yann.lecun.com/exdb/mnist/

[7] available at: http://icn.bjtu.edu.cn/Visint/resources/Scenesig.aspx

2) K-nearest neighbour (KNN) classifier;

3) Decision tree (DT) classifier;

4) eClass-1 classifier [8];

5) AutoClass-1 classifier [6] and

6) TEDAClass classifier.

For Singapore dataset, we follow the commonly used experimental protocol [17] by randomly selecting 20% of the images of each class for training and using the rest for testing in an offline scenario. The average classification accuracy is reported in Table 8 after 10 Monte Carlo experiments, where the best result is bolded. We also involve the following "state-of-the-art" approaches in the area of remote sensing for comparison:

1) Spatial pyramid matching kernel (SPMK) [26];

2) Pyramid of spatial relations (PSR) [11];

3) Bag-of-visual-words (BoVW) [48];

4) SIFT-based features with sparse coding (SIFTSC) [12];

5) Two-level feature representation with sparse coding (TLFRSC) [17];

6) Vector of locally aggregated descriptors (VLAD) [20], and

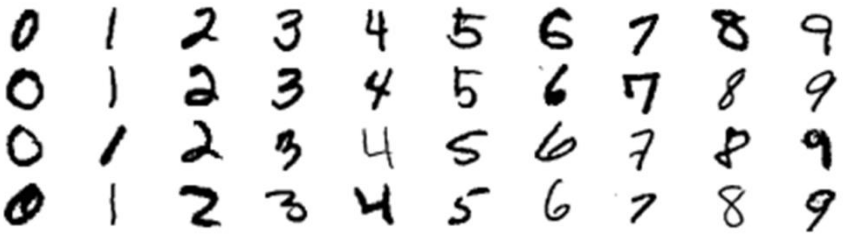
7) Two-level feature representation with selection-constrained coding (TLFRSCC) [17].

Table 6 Details of the datasets used for numerical example

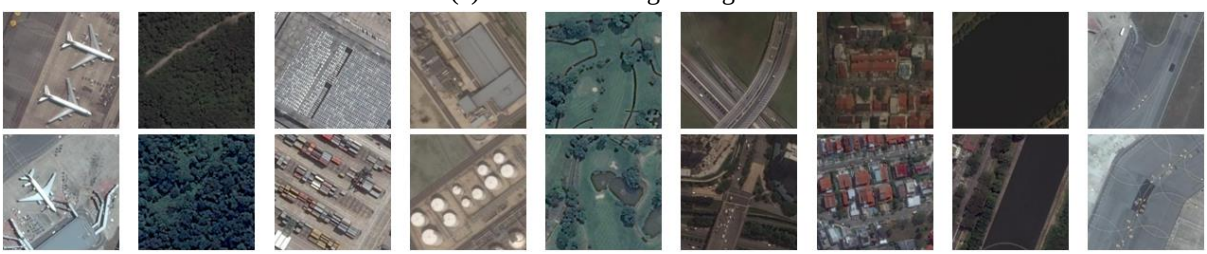
Dataset		Number of Classes	Number of Samples	Number of Attributes
MNIST	Training set	10	60000	512+1 label
	Testing set		10000	
Singapore		9	1086	4096+1 label

From Tables 7 and 8 we can see that the proposed SOF classifier can exhibit very high performance compared with the "state-of-the-art" approaches in large-scale, high-dimensional problems.

Fig. 6 shows the high efficiency of the proposed approach in both offline and online training. In addition, one may notice that, in offline scenario, the SOF classifier can be trained on 60000 images for less than 20 minutes, 2 minutes per rule, which is far more efficient than the deep learning-based approaches [13], which require several hours plus several GPUs. In the online scenario, the SOF classifier only needs less than 0.025 second to process each image and is self-evolving.



(a) Handwritten digit images



(b) Remote sensing images

Fig. 5. Example images of the two benchmark datasets.

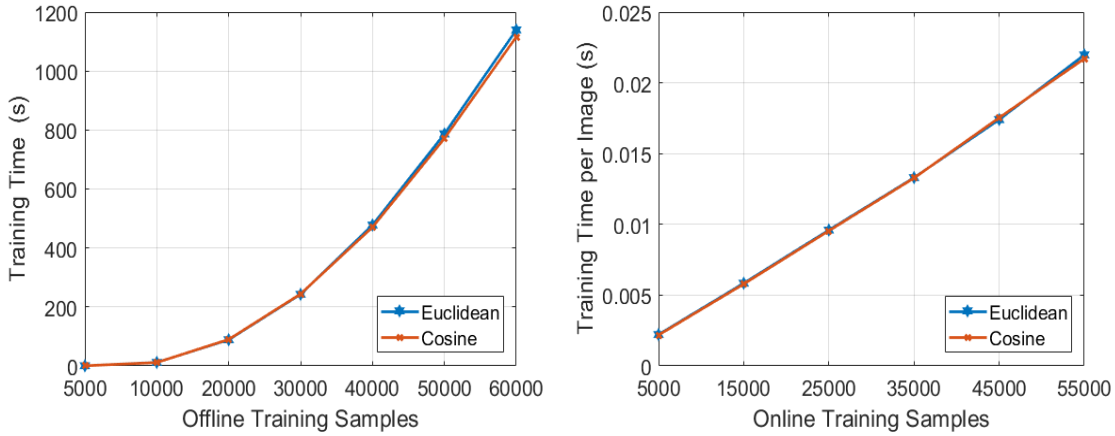
Moreover, due to the prototype-based nature of the SOF classifier, one can obtain a highly transparent, human-understandable AnYa type fuzzy rules after the training process, which is also one of the advantages of

the SOF classifier. Illustrative examples of the fuzzy rules generated from the two benchmark problems are given in Table 9. For visual clarity, we resized the images in the table.

From the above numerical examples, one can conclude that the proposed SOF classifier in this paper is a powerful alternative to the existing approaches with high accuracy, transparency and fast self-evolving learning.

Table 7. Performance comparison (in accuracy) on MNSIT dataset

Approach		Classification Performance on Different Amounts of Training Samples						
		5000	10000	20000	30000	40000	50000	60000
SOF-Euclidean	Offline	0.9672	0.9731	0.9785	0.9813	0.9828	0.9839	0.9854
	Online		0.9746	0.9802	0.9818	0.9835	0.9846	0.9850
SOF-Cosine	Offline	0.9718	0.9773	0.9818	0.9844	0.9851	0.9862	0.9855
	Online		0.9776	0.9825	0.9845	0.9862	0.9867	0.9868
SVM		0.9776	0.9810	0.9838	0.9856	0.9861	0.9866	0.9869
KNN		0.9678	0.9749	0.9798	0.9824	0.9840	0.9854	0.9861
DT		0.8174	0.8429	0.8654	0.8764	0.8818	0.8890	0.8933
eClass1		0.9685	0.9719	0.9732	0.9746	0.9745	0.9746	0.9746
AutoClass1		0.9691	0.9724	0.9738	0.9744	0.9742	0.9738	0.9742
TEDAClass		0.9716	0.9738	0.9753	0.9768	0.9766	0.9765	0.9763



(a) Offline scenario (b) Online scenario

Fig. 6. The average training time consumption with different amounts of training samples

On the other hand, we have to admit that the proposed approach does not converge to a locally optimal solution of the problem after the training process. In order to achieve the local optimality, one needs to predefine an objective function and involve an iterative process to minimise the objective function [41]. Because of the greedy type search used in the SOF classifier (multiple peaks, etc.), there is no guarantee for convergence to locally optimal solution obtained by the proposed approach. Nonetheless, by involving a similar iterative process as described in [41], the SOF classifier can also converge to the locally optimal solutions, but this is out of the scope of this paper.

Table 8. Performance comparison on Singapore dataset

Approach	*Acc*	Approach	*Acc*
SOF-Euclidean	0.9493	SPMK	0.8285
SOF-Cosine	**0.9711**	PSR	0.8454
SVM	0.9525	BoVW	0.8741
KNN	0.8527	SIFTSC	0.8758
DT	0.7241	TLFRSC	0.8827
eClass0	0.9181	VLAD	0.8870
Simpl_eClass0	0.9262	TLFRSCC	0.9094

Table 9. Illustrative fuzzy rules

Dataset	Fuzzy rule
MNIST	***IF*** **(I~** **)** ***AND*** **(I~** **)** ***AND*** **(I~** **)** ***AND***… ***AND*** **(I~**  **)** ***AND*** **(I~**  **)** ***THEN*** (Digit 1)
	IF **(I~**  **)** ***AND*** **(I~**  **)** ***AND*** **(I~** **)** ***AND***… ***AND*** **(I~**  **)** ***AND*** **(I~**  **)** ***THEN*** (Digit 5)
	IF **(I~** **)** ***AND*** **(I~** **)** ***AND*** **(I~** **)** ***AND*** … ***AND*** **(I~** **)** ***AND*** **(I~** **)** ***THEN*** (Digit 9)
Singapore	***IF*** **(I~**  **)** ***AND*** **(I~**  **)** ***AND*** **(I~**  **)** ***AND*** … ***AND*** **(I~**  **)** ***AND*** **(I~**  **)** ***THEN*** (Airplane)
	IF **(I~**  **)** ***AND*** **(I~**  **)** ***AND*** **(I~**  **)** ***AND***… ***AND*** **(I~**  **)** ***AND*** **(I~**  **)** ***THEN*** (Meadow)
	IF **(I~**  **)** ***AND*** **(I~**  **)** ***AND*** **(I~**  **)** ***AND***… ***AND*** **(I~**  **)** ***AND*** **(I~**  **)** ***THEN*** (Highway)

6. Conclusions and Future Works

In this paper, a new type of self-organising fuzzy logic (SOF) classifier is proposed on the basis of the AnYa type fuzzy system and Empirical Data Analytics computational framework. The proposed SOF classifier is free from predefined parameters or *prior* assumptions about the data generation model and it is driven by the empirically observed data. The proposed classifier can identify prototypes from the offline training data in a highly efficient way and continue to learn from the streaming data recursively. It can support various types of distances and/or dissimilarity measures and can also conduct classification under different levels of granularity, which makes it a powerful alternative to the "state-of-the-art" approaches and gives its strong potential in real applications. Numerical examples on benchmark problems demonstrate the high performance of the SOF approach and show its ability in handling different sorts of problems.

As future work, we will apply the proposed approach to different complex problems and further improve its performance. The local optimality of the classifier will be further investigated. We will also use first order fuzzy rules in the SOF classifier to increase the degrees of freedom and thus, allow a higher performance.

References

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in International Conference on Database Theory, 2001, pp. 420–434.

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

[3] P. Angelov and X. Gu, "A cascade of deep learning fuzzy rule-based image classifier and SVM," in International Conference on Systems, Man and Cybernetics, 2017, pp. 1–8.

[4] P. Angelov, X. Gu, and D. Kangin, "Empirical data analytics," Int. J. Intell. Syst., vol. 32, no. 12, pp. 1261–1284, 2017.

[5] P. Angelov, X. Gu, J. Principe, "A generalized methodology for data analysis", IEEE Transactions on Cybernetics, DOI: 10.1109/TCYB.2017.2753880, 2017.

[6] P. Angelov, D. Kangin, and D. Kolev, "Symbol recognition with a new autonomously evolving classifier AutoClass," in International Joint Conference on Neural Networks, 2015, pp. 1–8.

[7] 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, 2011.

[8] 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.

[9] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is ‘nearest neighbors’ meaningful?,” in International Conference on Database Theoryheory, 1999, pp. 217–235.
[10] M. A. Borgi, D. Labate, M. El Arbi, and C. Ben Amar, “Regularized shearlet network for face recognition using single sample per person,” in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 514–518.
[11] S. Chen and Y. Tian, “Pyramid of spatial relatons for scene-level land use classification,” IEEE Trans. Geosci. Remote Sens., vol. 53, no. 4, pp. 1947–1957, 2015.
[12] A. M. Cheriyadat, “Unsupervised feature learning for aerial scene classification,” IEEE Trans. Geosci. Remote Sens., vol. 52, no. 1, pp. 439–451, 2014.
[13] D. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in Conference on Computer Vision and Pattern Recognition, 2012, pp. 3642–3649.
[14] N. Cristianini and J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods. Cambridge: Cambridge University Press, 2000.
[15] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 5, pp. 603–619, 2002.
[16] P. Cunningham and S. J. Delany, “K-nearest neighbour classifiers,” Mult. Classif. Syst., vol. 34, pp. 1–17, 2007.
[17] 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.
[18] X. Gu, P. Angelov, D. Kangin, J. Principe, “Self-organised direction aware data partitioning algorithm”, Inf. Sci., vol. 423, pp. 80-95, 2018.
[19] X. Gu, P. P. Angelov and J. C. Principe, “A method for autonomous data partitioning,” Inf. Sci., submitted for publication, 2017.
[20] 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.
[21] D. Kangin, P. Angelov, and J. A. Iglesias, “Autonomously evolving classifier TEDAClass,” Inf. Sci. (Ny)., vol. 366, pp. 1–11, 2016.
[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] L. Kuncheva, Combining pattern classifiers: methods and algorithms. Hoboken, New Jersey: John Wiley & Sons, 2004.
[25] T. Larrain, J. S. J. Bernhard, D. Mery, and K. W. Bowyer, “Face recognition using sparse fingerprint classification algorithm,” IEEE Trans. Inf. Forensics Secur., vol. 12, no. 7, pp. 1646–1657, 2017.
[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] L. Lu, L. Di, and Y. Ye, “A decision-tree classifier for extracting transparent plastic-mulched Landcover from landsat-5 TM images,” IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 7, no. 11, pp. 4548–4558, 2014.
[28] 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.
[29] E. Lughofer, R. Richter, U. Neissl, W. Heidl, C. Eitzinger, and T. Radauer, “Explaining classifier decisions linguistically for stimulating and improving operators labeling behavior,” Inf. Sci. (Ny)., 2017.
[30] E. H. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” Int. J. Man. Mach. Stud., vol. 7, no. 1, pp. 1–13, 1975.
[31] F. Noorbehbahani, A. Fanian, R. Mousavi, and H. Hasannejad, “An incremental intrusion detection system using a new semi-supervised stream classification method,” Int. J. Commun. Syst., vol. 30, no. 4, pp. 1–26, 2017.
[32] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, Spatial tessellations: concepts and applications of Voronoi diagrams, 2nd ed. Chichester, England: John Wiley & Sons., 1999.

[33] A. Oliva and A. Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope,” Int. J. Comput. Vis., vol. 42, no. 3, pp. 145–175, 2001.
[34] N. Passalis and A. Tefas, “Neural bag-of-features learning,” Pattern Recognit., vol. 64, pp. 277–294, 2017.
[35] W. Pedrycz, Granular computing: analysis and design of intelligent systems. CRC press, 2013.
[36] W. Pedrycz and A. Bargiela, “Granular clustering: a granular signature of data,” IEEE Trans. Syst. Man, Cybern. Part B Cybern., vol. 32, no. 2, pp. 212–224, 2002.
[37] 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.
[38] 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.
[39] H. J. Rong, N. Sundararajan, G. Bin Huang, and P. Saratchandran, “Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction,” Fuzzy Sets Syst., vol. 157, no. 9, pp. 1260–1275, 2006.
[40] S. R. Safavian and D. Landgrebe, “A survey of decsion tree clasifier methodology,” IEEE Trans. Syst. Man. Cybern., vol. 21, no. 3, pp. 660–674, 1990.
[41] 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.
[42] M. Senoussaoui, P. Kenny, P. Dumouchel, and T. Stafylakis, “Efficient iterative mean shift based cosine dissimilarity for multi-recording speaker clustering,” IEEE Int. Conf. Acoust. Speech Signal Process., pp. 7712–7715, 2013.
[43] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in International Conference on Learning Representations, 2015, pp. 1–14.
[44] A. L. Suárez-Cetrulo and A. Cervantes, “An online classification algorithm for large scale data streams: iGNGSVM,” Neurocomputing, vol. 262, pp. 67–76, 2017.
[45] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” IEEE Trans. Syst. Man. Cybern., vol. 15, no. 1, pp. 116–132, 1985.
[46] Q. Weng, Z. Mao, J. Lin, and W. Guo, “Land-use classification via extreme learning classifier based on deep convolutional features,” IEEE Geosci. Remote Sens. Lett., vol. 14, no. 5, pp. 704–708, 2017.
[47] G. S. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, and L. Zhang, “AID: A benchmark dataset for performance evaluation of aerial scene classification,” IEEE Trans. Geosci. Remote Sens., vol. 55, no. 7, pp. 3965–3981, 2017.
[48] 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.
[49] J. Yao, A. V Vasilakos, and W. Pedrycz, “Granular computing: perspectives and challenges.,” IEEE Trans. Cybern., vol. 43, no. 6, pp. 1977–89, 2013.
[50] X. T. Yuan, B. G. Hu, and R. He, “Agglomerative mean-shift clustering,” IEEE Trans. Knowl. Data Eng., vol. 24, no. 2, pp. 209–219, 2012.