Elsevier Editorial System(tm) for Information Sciences

Manuscript Draft

Manuscript Number: INS-D-19-663R2

Title: A Hierarchical Prototype-Based Approach for Classification

Article Type: Full length article

Keywords: prototype-based; hierarchical structure; classification; multimodal distribution

Corresponding Author: Dr. Xiaowei Gu,

Corresponding Author's Institution: Infolab21, Lancaster University

First Author: Xiaowei Gu

Order of Authors: Xiaowei Gu; Weiping Ding

Abstract: In this paper, a novel hierarchical prototype-based approach for classification is proposed. This approach is able to perceive the data space and derive the multimodal distributions from streaming data at different levels of granularity in an online manner, based on which it further identifies meaningful prototypes to self-organize and self-evolve its hierarchical structure for classification. Thanks to the prototype-based nature, the system structure of the proposed classifier is highly transparent, and its learning process is of "one pass" type and computationally lean. Its decision-making process follows the "nearest prototype" principle and is fully explainable. The proposed approach is capable of presenting the learned knowledge from data in an easy-to-interpret prototype-based hierarchical form to users, and is an attractive tool for solving large-scale, complex real-world problems. Numerical examples based on various benchmark problems justify the validity and effectiveness of the proposed concept and general principles.

Research Data Related to this Submission

There are no linked research data sets for this submission. The following reason is given:
Data will be made available on request

Revision Note

Dear Editor-in-Chief

Firstly, we would like to thank you for handling our manuscript entitled: A Hierarchical Prototype-Based Approach for Classification (INS-D-19-663R1). We would also like to express our appreciation to the anonymous referees for their valuable efforts and very kind approvals.

We have revised this manuscript carefully to improve the language quality and reduced the number of references to 50.

The changes have been highlighted in yellow in this manuscript for your convenience.

Thank you again!

Sincerely,

Authors

A Hierarchical Prototype-Based Approach for Classification

Xiaowei Gu[1] and Weiping Ding[2]

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

[2]School of Information Science and Technology, Nantong University, Nantong, 226019, China

Email: x.gu3@lancaster.ac.uk; ding.wp@ntu.edu.cn

Abstract: In this paper, a novel hierarchical prototype-based approach for classification is proposed. This approach is able to perceive the data space and derive the multimodal distributions from streaming data at different levels of granularity in an online manner, based on which it further identifies meaningful prototypes to self-organize and self-evolve its hierarchical structure for classification. Thanks to the prototype-based nature, the system structure of the proposed classifier is highly transparent, and its learning process is of "one pass" type and computationally lean. Its decision-making process follows the "nearest prototype" principle and is fully explainable. The proposed approach is capable of presenting the learned knowledge from data in an easy-to-interpret prototype-based hierarchical form to users, and is an attractive tool for solving large-scale, complex real-world problems. Numerical examples based on various benchmark problems justify the validity and effectiveness of the proposed concept and general principles.

Keywords: prototype-based, hierarchical structure, classification, multimodal distribution

1. Introduction

Classification is a supervised machine learning technique used for predicting class labels of new observations [30]. It is a hotly studied problem in the machine learning and statistics domain. Till now, a variety of classification algorithms have been proposed and successfully applied to different areas, e.g., computer vision [26],[38], biomedical sciences [12], remote sensing [42],[49], finance [3], etc.

Nowadays, with the rapid development of artificial intelligence (AI) techniques, the explainability and transparency of the decisions made by machine learning algorithms are becoming increasingly important [20], especially when these approaches started to be implemented for safety-critical applications, such as autonomous vehicles [43] and medical diagnose [13], etc. However, mainstream classification algorithms, for example, deep neural networks (DNNs) [27] and support vector machines (SVMs) [9] are typical type of "black box" models. Other popular classification approaches, for example, K-nearest neighbour (KNN) [10], decision tree (DT)/random forests [39], and rule-based models [1],[17],[23] can be extremely hard to interpret when dealing with high-dimensional, large-scale, and complex problems. The lack of transparency and human-interpretability in the state-of-the-art machine learning approaches is a critical issue while not yet solved. There is a high demand in developing alternative approaches that can provide high levels of transparency, human-interpretability and, at the same time, perform learning lifelong in a computationally lean manner with high precision comparable to, and even surpassing humans.

In this paper, a novel hierarchical prototype-based (HP) approach with such characteristics is proposed for classification. The HP classifier decomposes complex problems into a series of simpler local models with different levels of granularity and represents them with meaningful prototypes aggregated naturally in a pyramidal hierarchy form. These prototypes are objectively identified from streaming data through an online, autonomous and highly computationally efficient process with different levels of granularity from low to high, which naturally results in a multi-layer structure. They represent the local peaks of multimodal distributions derived from data at different levels of specificity without *prior* knowledge of the problems. The prototype identification process of the HP classifier is of "one pass" type, highly computationally efficient and entirely data-driven. It is also non-parametric in the sense that no assumptions on data generation model with user- and problem- specific parameters are required to be made. Instead of being a "black box" model, the system structure of the proposed HP classifier is highly transparent and, its learning and decision-making processes are fully explainable and interpretable for human. In addition, the knowledge base of the HP classifier can be visualized in a human-understandable hierarchical form thanks to its prototype-based nature. The upper layers of the hierarchies are very useful for human users to quickly understand the general picture of the problem, and they can be used for efficient coarse classification. Meanwhile, the lower layers contain lots of fine details

(which may take time to interpret) and can be used for performing classification with high accuracy. Numerical examples presented in this paper validate the proposed concept and general principles, and further demonstrate the strong advantages of the HP classifier as an attractive tool for solving large-scale complex classification problems.

The key contributions of this paper are as follows: (1) a new approach that can self-learn and self-evolve highly transparent pyramidal hierarchies from streaming data through an "one pass" learning process; (2) the capability to perceive complex problems with different levels of specificity and mine valuable information in a highly computationally efficient manner; (3) the abilities of presenting the gained knowledge from data in a human-interpretable prototype-based hierarchical form and clearly explaining the rationales behind any decisions.

The remainder of this paper is organized as follows. Section 2 provides a review of related works. The algorithmic procedure of the HP classifier is detailed in section 3. Numerical experiments and discussions are given in section 4. This paper is concluded by section 5.

2. Related Works

Classification is a hotly researched topic, and there have been a wide variety of successful algorithms existing in the literature. Since it is practically impossible to cover all of them due to the limited space of this paper, the review of related works in this paper is concentrated on well-known classification approaches of the following three types: (1) multi-layer; (2) fuzzy rule-based and (3) prototype-based.

DNNs (or artificial neural networks, ANNs) [27], which include feedforward neural networks (FNNs) [12], convolutional neural networks (CNNs) [26],[38] and recurrent neural networks (RNNs) [18], are the best-known classification approaches with multi-layer architecture. Currently, DNNs are the state-of-the-art for classification, and they have demonstrated very high performance on a number of benchmark problems as well as real-world applications [27]. Nonetheless, DNNs suffer from several deficiencies, which include [19] (1) the training and decision-making processes lack explainability and human-interpretability, and the inner structure is opaque; (2) the training process is data- and computational resource-hungry and limited to offline; (3) the performance is fragile to new data patterns. In addition, some recent researches also suggest that DNNs can be easily fooled as they can produce high confidence predictions for unrecognizable images [31]. These shortcomings significantly influence the applicability of DNNs in real-world application scenarios [19],[20].

Evolving intelligent systems (EISs) [28],[40] have demonstrated success in many real-world applications, e.g., classification, prediction, control, anomaly detection, and are becoming increasingly popular thanks to their simpler system structure and explainable learning and decision-making processes. EISs, generally, can be implemented in the forms of multi-layer neuro-fuzzy systems [24] or rule-based models [1],[3],[17]. The majority of EISs are designed for processing streaming data "on the fly", and their learning processes are, usually, of "one pass" type [28]. EISs are capable of continuously self-evolving based on new observations from data streams and extending the knowledge base to incorporate new data patterns in real time. In contrast with DNNs, EISs usually offer much higher transparency and human-interpretability. Nonetheless, it is often observed that EISs can be unfavourably obese and uninterpretable for high-dimensional and large-scale problems.

Prototypes play an instrumental role in prototype-based classification algorithms. Different prototype-based approaches use different ways to identify prototypes from data, and this creates the differences in performance, computational efficiency, system transparency and interpretability. KNN algorithms [10] conduct classification using the "nearest neighbours" principle by comparing new observations with a pre-defined number (k) of labelled data samples. Therefore, KNN can be viewed as a type of prototype-based approaches where all the data samples are treated as prototypes. KNN requires all training samples to be stored in the memory, and the transparency and human-interpretability of the system structure are very low. SVMs are also a type of prototype-based classifiers since they perform classification based on support vectors (prototypes), which are derived from training data [9]. Nonetheless, the operating mechanism of SVMs is much more sophisticated, which involves an iterative optimization process to identify the maximum-margin hyperplane in the data space, and, thus, SVMs are the typical type of "black box" models like ANNs. Zero-order EISs [1],[2],[17] are, in general, based on prototypes, and they have simpler, more flexible and transparent system structure than other types of EISs, e.g. first-order [24] and higher-order ones [3]. They are more computationally efficient and capable of handling multi-class classification tasks. Nonetheless, the transparency and explainability of zero-

order EISs are very limited for high-dimensional, large-scale, complex problems. Other well-known prototype-based classifiers include learning vector quantization (LVQ) [25], self-organising map (SOM) [25], both of which are ANNs and typical "black box" type models.

Compared with alternative approaches, the proposed HP classifier is highly transparent thanks to its prototype-based nature and hierarchical system structure. The pyramidal hierarchies of the HP classifier can be clearly visualized with meaningful links between prototypes of successive layers. The upper layer of a hierarchy is composed of a smaller amount of highly abstract and more representative prototypes, and the bottom layer has a larger number of prototypes with finer details. The learning process of the HP classifier is non-iterative, non-parametric, recursive and highly interpretable for human. The proposed approach does not involve any weight or parameter optimization operation, and it can be trained in an extremely lean manner. Moreover, its system structure is dynamically evolving, and its meta-parameters are self-updating with new data, which largely strengthens the capability of handling nonstationary data streams. More importantly, the rationales behind any decisions made by the proposed approach can be explained clearly because the decision-making process strictly follows the "nearest prototype" principle.

In the following section, the algorithmic details of the HP classifier will be described.

3. The HP Classifier

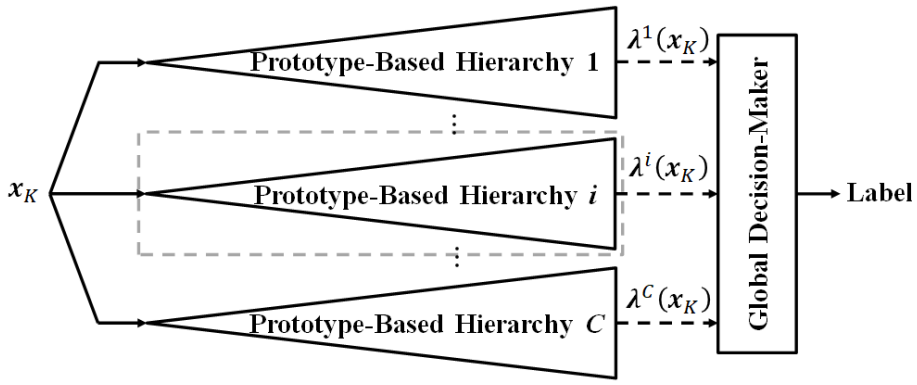
In this section, the general architecture, learning and decision-making processes of the proposed HP classifier are presented in detail. The computational complexity of the proposed approach is also analysed at the end of this section.

First of all, let $\{\boldsymbol{x}\} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k, \dots, \boldsymbol{x}_K, \dots\}$ be a particular data stream in the N-dimensional real data space, $\mathbf{R}^N$, where $\boldsymbol{x}_k = \left[x_{k,1}, x_{k,2}, \dots, x_{k,N}\right]^T \in \mathbf{R}^N$, and the subscript k denotes the time instance at which $\boldsymbol{x}_k$ is observed. The data stream, $\{\boldsymbol{x}\}$ is composed of data samples of C different classes and, thus, at the K^{th} time instance, the observed data samples from this stream can be divided into C subsets based on their class labels: $\{\boldsymbol{x}\}_K^i = \left\{\boldsymbol{x}_1^i, \boldsymbol{x}_2^i, \dots, \boldsymbol{x}_{K^i}^i\right\}$ (the superscript i denotes the i^{th} class; $i = 1,2, \dots, C$) and there is $K = \sum_{i=1}^{C} K^i$. In the remainder of this paper, all the mathematical derivations are conducted at the K^{th} time instance by default unless specifically declared otherwise.

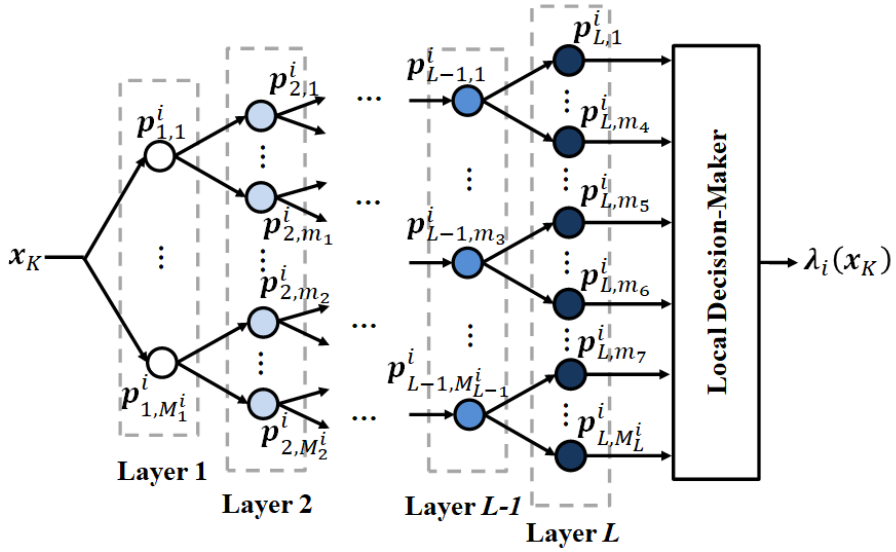
3.1. General Architecture

The general architecture of the HP classifier is depicted in Fig. 1. As one can see from Fig. 1(a), the HP classifier consists of C different prototype-based pyramidal hierarchies, which correspond to the C available classes in the data stream (one hierarchy per class). During the learning process, each hierarchy is trained in parallel using data samples of the corresponding class in a self-organizing, "one-pass" manner. The zoom-in structure of the i^{th} hierarchy is given in Fig. 1(b), where $\boldsymbol{p}_{l,j}^i$ denotes the j^{th} prototype at the l^{th} layer of the i^{th} hierarchy, which is derived from data samples of the i^{th} class of the data stream; $l = 1,2, \dots, L$; L is the layer number; $j = 1,2, \dots, M_l^i$; M_l^i is the number of prototypes at the l^{th} layer and there are $M_1^i \leq M_2^i \leq \cdots \leq M_{L-1}^i \leq M_L^i$; $1 \leq m_1 \leq m_2 \leq M_2^i$; $1 \leq m_2 \leq M_{L-1}^i$; $1 \leq m_4 \leq m_5 \leq m_6 \leq m_7 \leq M_{L-1}^i$. In this paper, without loss of generality, all prototype-based hierarchies of the HP classifier have the same layer number, namely, L.

Prototypes in the C pyramidal hierarchies are derived directly from data in an "one pass", top-down manner. These prototypes at upper layers of the hierarchies contain highly generalized information and they are more abstract and representative. Meanwhile, prototypes at the lower layers contain finer details and they are closer to the originally observed data samples in the data space. Each prototype at a particular layer (except for apex prototypes at the top layer) is linked with one immediate superior prototype at the layer above, and it represents a local peak of the multimodal distribution of data associated with its immediate superior. At the same time, each prototype (except for leaf prototypes at the bottom layer) is linked with one or more immediate subordinate prototypes at the layer below. For example, in Fig. 1(b), $\boldsymbol{p}_{1,1}^i$ is the immediate superior of $\boldsymbol{p}_{2,1}^i, \dots, \boldsymbol{p}_{2,m_1}^i$, and $\boldsymbol{p}_{L,1}^i, \dots, \boldsymbol{p}_{L,m_4}^i$ are the immediate subordinates of $\boldsymbol{p}_{L-1,1}^i$.



(a) System structure



(b) Zoom-in structure of the i^{th} prototype-based hierarchy

Fig.1. The general architecture of the HP classifier

In the following two subsections, the learning and the decision-making processes of the HP classifier are presented in detail. The computational complexity analysis of the proposed approach is also given at the end of this section.

3.2. Learning Process

In this subsection, the algorithmic procedure for autonomously identifying a hierarchical prototype-based system structure is detailed as follows. As the proposed HP approach identifies prototypes and self-organizes a top-down pyramidal hierarchy from training samples of each class separately, only the learning process of the i^{th} prototype-based hierarchy is presented ($i = 1,2,\dots,C$). The same principles can be applied to all other hierarchies in the system. By default, for each observed data sample of the i^{th} class, $\boldsymbol{x}_k^i$ ($k = 1,2,\dots,K^i,\dots$), it is firstly normalized by its Euclidean norm:

$$\boldsymbol{x}_k^i \leftarrow \frac{\boldsymbol{x}_k^i}{\|\boldsymbol{x}_k^i\|} \tag{1}$$

where $\|\boldsymbol{x}_k^i\| = \sqrt{\sum_{j=1}^{N}\left(x_{k,j}^i\right)^2}$. This type of normalization can convert the Euclidean distance between data samples into a cosine dissimilarity-based distance measure and enhances the ability of the HP classifier for handling high-dimensional complex problems [17].

Step 0. System Initialization

The first observed data sample of the i^{th} class, $\boldsymbol{x}_{K^i}^i$ ($K^i = 1$) is used for initializing the hierarchy and is treated as the first prototype at each layer ($l = 1,2,\dots,L$):

$$M_l^i \leftarrow 1; \quad \boldsymbol{p}_{l,M_l^i}^i \leftarrow \boldsymbol{x}_{K^i}^i; \quad S_{l,M_l^i}^i \leftarrow 1 \tag{2}$$

where $S_{l,M_l^i}^i$ is the number of data samples associated with $\boldsymbol{p}_{l,M_l^i}^i$.

The links (subordinate relationships) between these prototypes of successive layers are, then, established to form a hierarchical structure. Firstly, the collection of apex prototypes is defined as:

$$\boldsymbol{\mathcal{L}}_0^i \leftarrow \left\{\boldsymbol{p}_{1,M_1^i}^i\right\} \tag{3a}$$

and, for the prototype at the l^{th} layer, $\boldsymbol{p}_{l,M_l^i}^i$ ($l = 1,2,\dots,L-1$), the collection of its immediate subordinates is initialized by equation (3b):

$$\boldsymbol{\mathcal{L}}_{l,M_l^i}^i \leftarrow \left\{\boldsymbol{p}_{l+1,M_{l+1}^i}^i\right\} \tag{3b}$$

Through equation (3), the i^{th} hierarchy is established in its initial form resembling a chain with $\boldsymbol{p}_{1,M_1^i}^i$ as the starting node and $\boldsymbol{p}_{L,M_L^i}^i$ as the ending node.

Step 1. System Dynamically Evolving

After initialization, the HP classifier is able to continuously self-evolve its system structure and self-update meta-parameters with steaming data. When a new data sample, $\boldsymbol{x}_{K^i}^i$ ($K^i \leftarrow K^i + 1$) is observed, the system updating process starts from the top layer ($l = 1$) in a top-down manner. Firstly, the nearest prototype at the l^{th} layer to $\boldsymbol{x}_{K^i}^i$ is identified using the following equation:

$$n_l^* = \begin{cases} \operatorname{argmin}_{\boldsymbol{p}\in\boldsymbol{\mathcal{L}}_0^i}\left(\left\|\boldsymbol{x}_{K^i}^i - \boldsymbol{p}\right\|\right) & if\ l = 1 \\ \operatorname{argmin}_{\boldsymbol{p}\in\boldsymbol{\mathcal{L}}_{l-1,n_{l-1}^*}^i}\left(\left\|\boldsymbol{x}_{K^i}^i - \boldsymbol{p}\right\|\right) & if\ l = 2,3,\dots,L \end{cases} \tag{4}$$

It is well-known that for a prototype-based approach, identifying a huge number of prototypes during the learning process on large-scale, complex problems is often inevitable. Equation (4) significantly improves the computational efficiency of the nearest prototype searching process by reducing the searching range from the whole data space to a small group of neighbouring prototypes. This can effectively avoid wasting computational resources because the majority of the identified prototypes in the data space are actually far away from $\boldsymbol{x}_{K^i}^i$. This searching technique allows the proposed HP approach to identify the nearest prototype in an extremely efficient manner compared with alternative approaches that are also based on the "nearest prototype" principle [2],[17].

Once the nearest prototype $\boldsymbol{p}_{l,n_l^*}^i$ is identified, the following condition is checked to see whether $\boldsymbol{x}_{K^i}^i$ has the potential to become a new prototype of the l^{th} layer:

$$\boldsymbol{Condition\ 1}: \quad \begin{array}{c} IF\ \left(\left\|\boldsymbol{x}_{K^i}^i - \boldsymbol{p}_{l,n_l^*}^i\right\|^2 > r_l\right) \\ THEN\ \left(\boldsymbol{x}_{K^i}^i\ is\ a\ new\ prototype\ at\ the\ l^{th}\ layer\right) \end{array} \tag{5}$$

where r_l is the radius of the influence area around each prototype at the l^{th} layer of the hierarchy, which defines a particular degree of closeness that is interesting and distinguishable under the l^{th} level of granularity [17]. If the distance between $\boldsymbol{x}_{K^i}^i$ and $\boldsymbol{p}_{l,n_l^*}^i$ is larger than r_l, it means that $\boldsymbol{x}_{K^i}^i$ represents a novel data pattern that no prototypes at the l^{th} layer can describe, Thus, $\boldsymbol{x}_{K^i}^i$ is added to the system as a new prototype. In fact, **Condition 1** (equation (5)) is a simplified combination of Conditions 4 and 5 in [17].

In this paper, by default, the value of r_l ($l = 1,2,\dots,L$) is derived by the following equation [2]:

$$r_l = 2\left(1 - cos\left(\frac{\theta_o}{2^{l-1}}\right)\right) \tag{6}$$

where $\theta_o = \frac{\pi}{2}$. The decreasing values of the radii of the successive layers allow the HP classifier to learn from data and partition the data space with different levels of granularity. The upper layers of the pyramidal hierarchies are capable of seeing the general picture of the problems and the lower layers can see more details. However, it has to be stressed that the radii of influence areas, r_l $(l = 1,2, \dots, L)$ are adjustable, and their values are only required to follow the simple principle: $r_1 > r_2 > \cdots > r_l > \cdots > r_L$, which means that prototypes at the upper layer have a larger influence area than prototypes at the lower layer. Furthermore, the radii, r_l $(l = 1,2, \dots, L)$ are not problem- and user-specific parameters and require no *prior* knowledge to be determined. On the contrary, their values can be decided based on users' preferences. One may explore different methods to define the values of r_l $(l = 1,2, \dots, L)$ depending on specific needs of the problems. Moreover, there is an alternative approach introduced in [17] for directly deriving r_l $(l = 1,2, \dots, L)$ from empirically observed data samples based on their mutual distances. One may also consider determining the layer number for each hierarchy depending on the complexity of the problems and the availability of computational resources. Nonetheless, the main purpose of this paper is to deliver the concept and general principles and, thus, only the general settings are implemented.

If **Condition 1** is unsatisfied, $\boldsymbol{x}^i_{K^i}$ is used for updating the meta-parameters of the nearest prototype at the l^{th} layer by the following formulas:

$$\boldsymbol{p}^i_{l,n^*_l} \leftarrow \frac{S^i_{l,n^*_l}}{S^i_{l,n^*_l}+1}\boldsymbol{p}^i_{l,n^*_l} + \frac{1}{S^i_{l,n^*_l}+1}\boldsymbol{x}^i_{K^i} \tag{7a}$$

$$\boldsymbol{p}^i_{l,n^*_l} \leftarrow \frac{\boldsymbol{p}^i_{l,n^*_l}}{\left\|\boldsymbol{p}^i_{l,n^*_l}\right\|} \tag{7b}$$

$$S^i_{l,n^*_l} \leftarrow S^i_{l,n^*_l} + 1 \tag{7c}$$

The main reason for using equation (7b) to re-normalize $\boldsymbol{p}^i_{l,n^*_l}$ after equation (7a) is that, according to the triangle inequality theorem, the Euclidean norm of the updated $\boldsymbol{p}^i_{l,n^*_l}$ by $\boldsymbol{x}^i_{K^i}$ using equation (7a) is smaller than 1 unless $\boldsymbol{p}^i_{l,n^*_l} = \boldsymbol{x}^i_{K^i}$:

$$\left\|\frac{S^i_{l,n^*_l}}{S^i_{l,n^*_l}+1}\boldsymbol{p}^i_{l,n^*_l} + \frac{1}{S^i_{l,n^*_l}+1}\boldsymbol{x}^i_{K^i}\right\| \leq \frac{S^i_{l,n^*_l}}{S^i_{l,n^*_l}+1}\left\|\boldsymbol{p}^i_{l,n^*_l}\right\| + \frac{1}{S^i_{l,n^*_l}+1}\left\|\boldsymbol{x}^i_{K^i}\right\| = 1 \tag{8}$$

Therefore, a re-normalization is needed to guarantee that the algorithm can correctly recognize the true nearest prototypes all the time by equation (4).

Then, $\boldsymbol{x}^i_{K^i}$ is passed to the next layer $(l \leftarrow l+1)$, and the same procedure (starting from equation (4)) is repeated to update the next layer of the hierarchy until the bottom layer is updated or being interrupted when **Condition 1** is met.

If **Condition 1** is satisfied, it means that $\boldsymbol{x}^i_{K^i}$ is distinctive from all existing prototypes at the l^{th} layer as well as the prototypes at lower layers. In such cases, $\boldsymbol{x}^i_{K^i}$ becomes a new prototype of the l^{th} layer with $\boldsymbol{p}^i_{l-1,n^*_{l-1}}$ as its immediate superior (if there is any). A new prototype is added to the l^{th} layer as well as the successive layers with meta-parameters initialized by equation (9) $(j = l, l+1, l+2, \dots, L)$:

$$M^i_j \leftarrow M^i_j + 1; \quad \boldsymbol{p}^i_{j,M^i_j} \leftarrow \boldsymbol{x}^i_{K^i}; \quad S^i_{j,M^i_j} \leftarrow 1 \tag{9}$$

After new prototypes have been added to the hierarchy, the system structure is, then, updated by creating links between these newly added prototypes and their immediate superiors. This results in a new branch adding into the hierarchy. Firstly, the starting node of this new branch is identified. If $\boldsymbol{x}^i_{K^i}$ is recognized as an apex prototype (namely, $l = 1$), $\boldsymbol{x}^i_{K^i}$ itself is the starting node and $\boldsymbol{\mathcal{L}}^i_0$ is updated as follows:

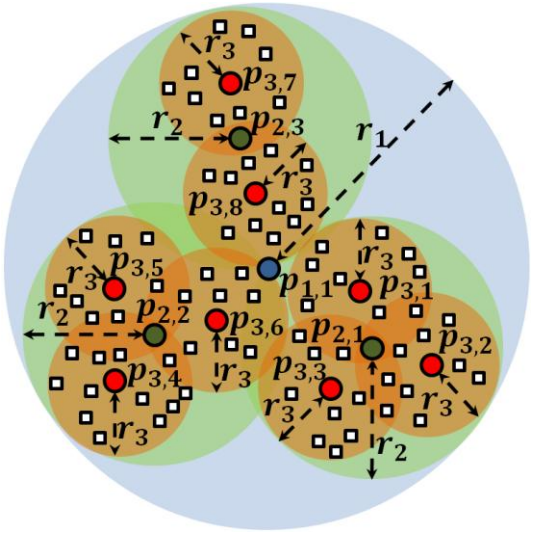
$$\boldsymbol{\mathcal{L}}^i_0 \leftarrow \boldsymbol{\mathcal{L}}^i_0 \cup \left\{\boldsymbol{p}^i_{1,M^i_1}\right\} \tag{10a}$$

Otherwise, $\boldsymbol{p}^i_{l-1,n^*_{l-1}}$ is recognized as the starting node of the new branch, and $\boldsymbol{\mathcal{L}}^i_{l-1,n^*_{l-1}}$ is updated by equation (10b):

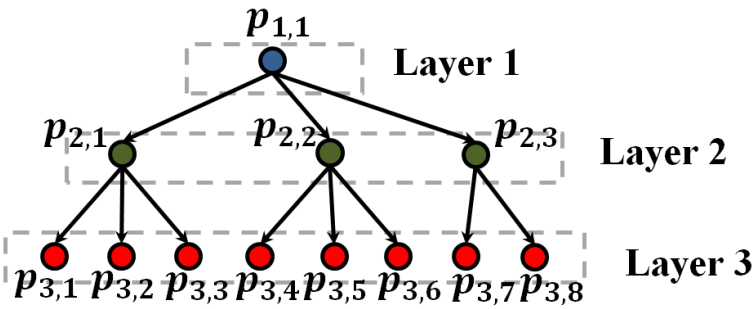
$$\boldsymbol{\mathcal{L}}^i_{l-1,n^*_{l-1}} \leftarrow \boldsymbol{\mathcal{L}}^i_{l-1,n^*_{l-1}} \cup \left\{\boldsymbol{p}^i_{l,M^i_l}\right\} \tag{10b}$$

After the starting node of this new branch is defined, the links between these newly added prototypes, namely, $\boldsymbol{p}^i_{l,M^i_l}, \boldsymbol{p}^i_{l+1,M^i_{l+1}}, \ldots, \boldsymbol{p}^i_{L,M^i_L}$ are created using equation (3b). Once all the new links have been established, the structure updating process is completed. The system goes back to **Step 1** ready for processing the next available data sample.

For better illustration, a three-layer prototype-based hierarchy derived from data samples observed in a 2D data space is depicted in Fig. 2 as an example. Fig. 2(a) demonstrates how the identified prototypes partition the data space with different levels of granularity; Fig. 2(b) depicts the three-layer hierarchy with meaningful links between prototypes of successive layers corresponding to Fig. 2(a). In Fig.2(a), white squares "□" are the empirically observed data samples in the data space; the large blue dot "·" is the apex prototype ($\boldsymbol{p}_{1,1}$) of the hierarchy and it is the most representative one; large green dots "·" are the prototypes ($\boldsymbol{p}_{2,1}, \boldsymbol{p}_{2,2}$ and $\boldsymbol{p}_{2,3}$) at the second layer and they correspond to the local peaks of the multimodal distribution of the data samples; large red dots "·" are the leaf prototypes ($\boldsymbol{p}_{3,1}, \boldsymbol{p}_{3,2}, \ldots, \boldsymbol{p}_{3,8}$) at the bottom layer. Each leaf prototype represents one of the local peaks of the multimodal distribution derived from data samples associated with its immediate superior. The blue, green and orange shadows indicate the respective areas of influence around prototypes of the top, middle and bottom layers, where r_1, r_2 and r_3 are the respective radii, and there is $r_1 > r_2 > r_3$. The links between these prototypes are presented in equation (11a):



(a) Partitioning the data space in three levels of granularity



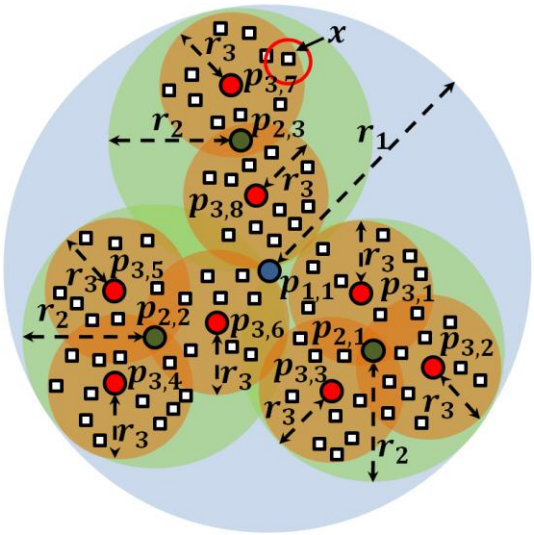
(b) The corresponding three-layer hierarchy

Fig. 2. Illustration of a three-layer prototype-based hierarchy derived from data

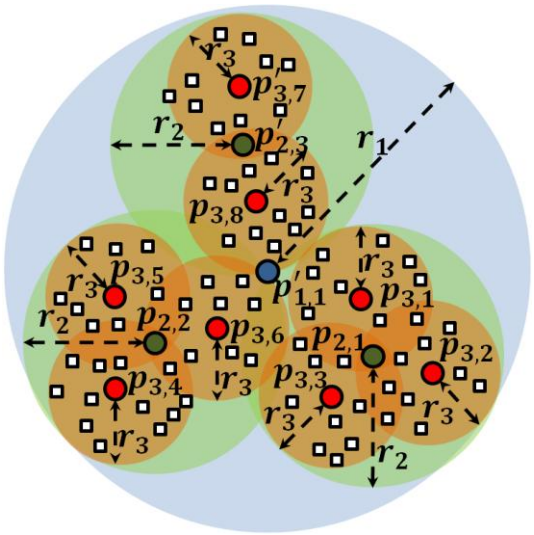
$$\begin{cases} \mathcal{L}_0 = \{\boldsymbol{p}_{1,1}\} \\ \mathcal{L}_{1,1} = \{\boldsymbol{p}_{2,1}, \boldsymbol{p}_{2,2}, \boldsymbol{p}_{2,3}\} \\ \mathcal{L}_{2,1} = \{\boldsymbol{p}_{3,1}, \boldsymbol{p}_{3,2}, \boldsymbol{p}_{3,3}\} \\ \mathcal{L}_{2,2} = \{\boldsymbol{p}_{3,4}, \boldsymbol{p}_{3,5}, \boldsymbol{p}_{3,6}\} \\ \mathcal{L}_{2,3} = \{\boldsymbol{p}_{3,7}, \boldsymbol{p}_{3,8}\} \end{cases} \tag{11a}$$

When a new data sample, $\boldsymbol{x}$ that represents a familiar data pattern is observed (see Fig. 3(a) for example), the meta-parameters of the nearest prototypes to $\boldsymbol{x}$ at each layer, namely, $\boldsymbol{p}_{1,1}$, $\boldsymbol{p}_{2,3}$ and $\boldsymbol{p}_{3,7}$ are updated using equation (7). The three prototypes, $\boldsymbol{p}_{1,1}$, $\boldsymbol{p}_{2,3}$ and $\boldsymbol{p}_{3,7}$ will be slightly shifted towards $\boldsymbol{x}$, and the updated prototypes are re-denoted by $\boldsymbol{p}'_{1,1}$, $\boldsymbol{p}'_{2,3}$ and $\boldsymbol{p}'_{3,7}$, respectively. The updated data space partitioning result is given in Fig. 3(b), and the corresponding three-layer hierarchy is depicted in Fig. 3(c). The links between prototypes of the updated hierarchy in Fig. 3(c) are given as:

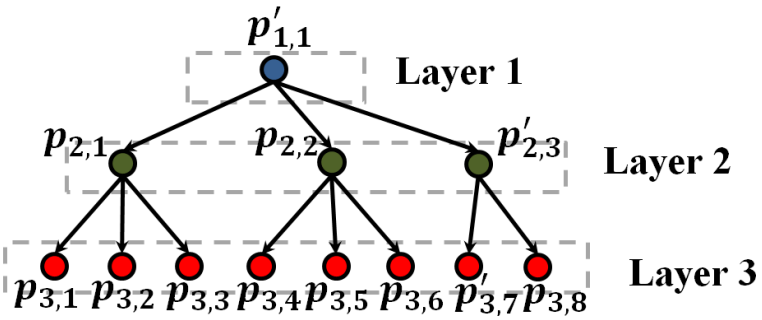
$$\begin{cases} \mathcal{L}_0 = \{\boldsymbol{p}'_{1,1}\} \\ \mathcal{L}_{1,1} = \{\boldsymbol{p}_{2,1}, \boldsymbol{p}_{2,2}, \boldsymbol{p}'_{2,3}\} \\ \mathcal{L}_{2,1} = \{\boldsymbol{p}_{3,1}, \boldsymbol{p}_{3,2}, \boldsymbol{p}_{3,3}\} \\ \mathcal{L}_{2,2} = \{\boldsymbol{p}_{3,4}, \boldsymbol{p}_{3,5}, \boldsymbol{p}_{3,6}\} \\ \mathcal{L}_{2,3} = \{\boldsymbol{p}'_{3,7}, \boldsymbol{p}_{3,8}\} \end{cases} \tag{11b}$$



(a) New observation in the data space



(b) The updated data partitioning results

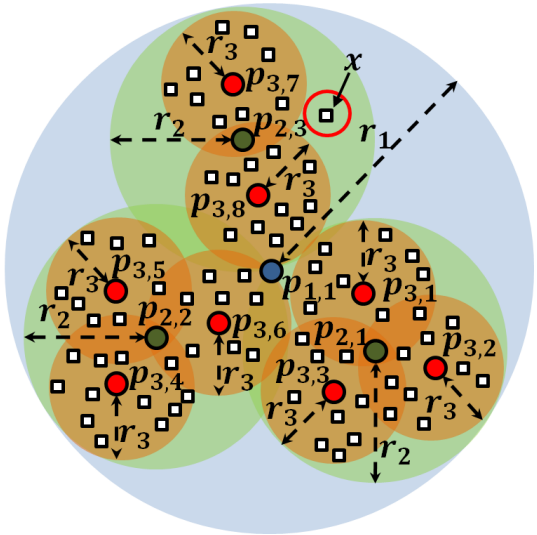


(c) The updated three-layer hierarchy

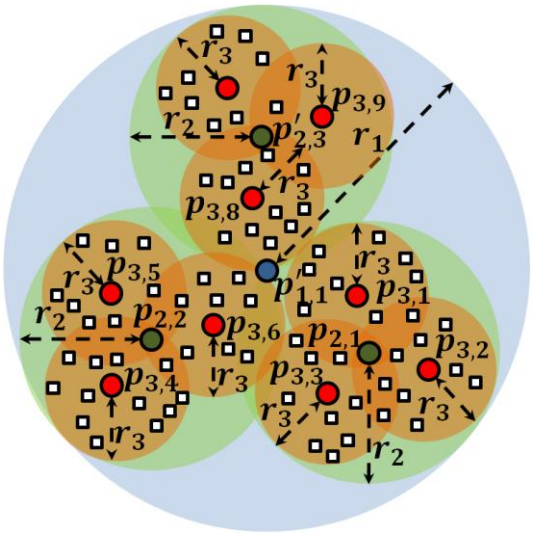
Fig. 3. Illustration of updating the existing prototypes of the three-layer hierarchy

On the other hand, if $\boldsymbol{x}$ represents an emerging data pattern (see Fig. 4(a)), the structure of the three-layer hierarchy will self-evolve to incorporate the new pattern. Firstly, the meta-parameters of $\boldsymbol{p}_{1,1}$ and $\boldsymbol{p}_{2,3}$ are updated because $\boldsymbol{x}$ falls into their areas of influence, and the updated prototypes are re-denoted by $\boldsymbol{p}'_{1,1}$ and $\boldsymbol{p}'_{2,3}$ respectively. Then, $\boldsymbol{x}$ will be recognized as a new leaf prototype (denoted by $\boldsymbol{p}_{3,9}$) at the bottom layer according to **Condition 1**, and a new branch of the three-layer hierarchy is established. The updated data space partitioning result and the new three-layer hierarchical structure are given in Figs. 4(b) and 4(c), respectively. The links between the prototypes of the updated hierarchy are given by equation (11c):

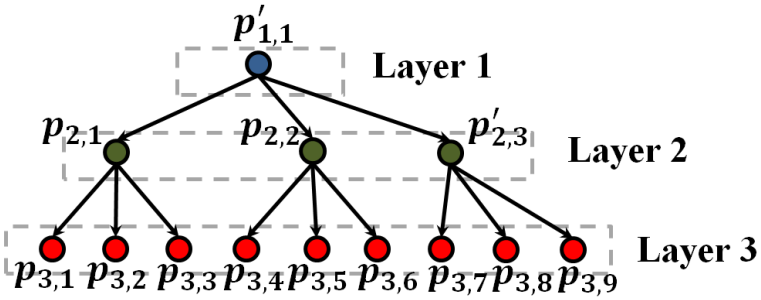
$$\begin{cases} \mathcal{L}_0 = \{\boldsymbol{p}'_{1,1}\} \\ \mathcal{L}_{1,1} = \{\boldsymbol{p}_{2,1}, \boldsymbol{p}_{2,2}, \boldsymbol{p}'_{2,3}\} \\ \mathcal{L}_{2,1} = \{\boldsymbol{p}_{3,1}, \boldsymbol{p}_{3,2}, \boldsymbol{p}_{3,3}\} \\ \mathcal{L}_{2,2} = \{\boldsymbol{p}_{3,4}, \boldsymbol{p}_{3,5}, \boldsymbol{p}_{3,6}\} \\ \mathcal{L}_{2,3} = \{\boldsymbol{p}_{3,7}, \boldsymbol{p}_{3,8}, \boldsymbol{p}_{3,9}\} \end{cases} \tag{11c}$$



(a) New observation in the data space



(b) The updated data partitioning results



(c) The updated three-layer hierarchy

Fig. 4. Illustration of adding a new prototype to the three-layer hierarchy

The learning process of the i^{th} prototype-based hierarchy is summarized by the following pseudo code:

Input: the data stream, $\{\boldsymbol{x}\}_K^i$
Algorithm begins

While (new data sample $\boldsymbol{x}_{K^i}^i$ is available or until interrupted)

a. Normalize $\boldsymbol{x}_{K^i}^i$ by its Euclidean norm: $\boldsymbol{x}_{K^i}^i \leftarrow \boldsymbol{x}_{K^i}^i / \|\boldsymbol{x}_{K^i}^i\|$;

b. **If** $(K^i = 1)$ **then**

i. **For** $l = 1$ to L **do**

1. $M_l^i \leftarrow 1$; $\quad \boldsymbol{p}_{l,M_l^i}^i \leftarrow \boldsymbol{x}_{K^i}^i$; $\quad S_{l,M_l^i}^i \leftarrow 1$;

2. $\begin{cases} \mathcal{L}_0^i \leftarrow \left\{\boldsymbol{p}_{l,M_l^i}^i\right\} & if\ l = 1 \\ \mathcal{L}_{l-1,M_{l-1}^i}^i \leftarrow \left\{\boldsymbol{p}_{l,M_l^i}^i\right\} & if\ l = 2,3,\dots,L \end{cases}$;

ii. **End for**

c. **Else**

i. **For** $l = 1$ to L **do**

1. Identify the nearest prototype, $\boldsymbol{p}_{l,n_l^*}^i$ to $\boldsymbol{x}_{K^i}^i$ by equation (4);

2. **If** (**Condition 1** is unsatisfied) **then**

* Update $\boldsymbol{p}_{l,n_l^*}^i$ and $S_{l,n_l^*}^i$ by equation (7);

3. **Else**

* **For** $j = l$ to L **do**

- $M_j^i \leftarrow M_j^i + 1$; $\quad \boldsymbol{p}_{j,M_j^i}^i \leftarrow \boldsymbol{x}_{K^i}^i$; $\quad S_{j,M_j^i}^i \leftarrow 1$;

* **End for**

* $\begin{cases} \mathcal{L}_0^i \leftarrow \mathcal{L}_0^i \cup \left\{\boldsymbol{p}_{l,M_l^i}^i\right\} & if\ l = 1 \\ \mathcal{L}_{l-1,n_{l-1}^*}^i \leftarrow \mathcal{L}_{l-1,n_{l-1}^*}^i \cup \left\{\boldsymbol{p}_{l,M_l^i}^i\right\} & if\ l = 2,3,\dots,L \end{cases}$;

* **For** $j = l + 1$ to L **do**

- $\mathcal{L}_{j-1,M_{j-1}^i}^i \leftarrow \left\{\boldsymbol{p}_{j,M_j^i}^i\right\}$;

* **End for**

* **Break** the **for** loop;

4. **End if**

ii. **End for**

d. **End if**

End while

Algorithm ends

Output: the i^{th} prototype-based hierarchy

3.3. Decision-Making Process

During the validation stage, the proposed HP classifier determines the class label of a given data sample by the "winner takes all" principle. Thanks to its hierarchical structure, one can choose to use any layer of the HP classifier for classification. The upper layers have less but more representative prototypes, and they can be used for performing efficient and coarse classification. The lower layers have more prototypes with fine details, and they can be used for performing more accurate classification.

Assuming that the l^{th} ($l = 1,2,\dots,L$) layer is used, for a particular unlabelled data sample denoted by $\boldsymbol{x}_K$, the local decision-maker of each prototype-based hierarchy (Fig. 1(b)) will produce a score of confidence, $\lambda^i(\boldsymbol{x}_K)$ based on the similarity between $\boldsymbol{x}_K$ and the nearest prototype at the selected layer following the "nearest prototype" principle ($i = 1,2,\dots,C$). In this subsection, two optional nearest prototype searching methods for calculating $\lambda^i(\boldsymbol{x}_K)$ are provided.

Mode A: the first method is to search the nearest prototype to $\boldsymbol{x}_K$ at the l^{th} layer of the hierarchy directly, and the score of confidence is calculated by the following equation:

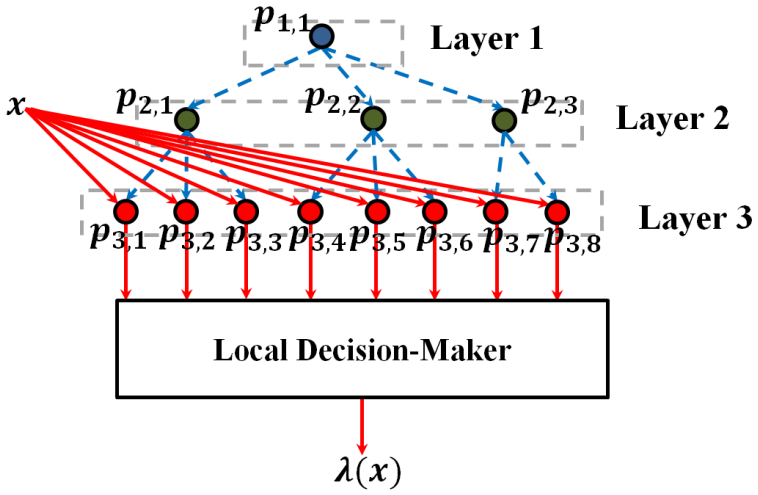
$$\lambda^i(\boldsymbol{x}_K) = \max_{\boldsymbol{p}\in\{\boldsymbol{p}\}_l^i}\left(e^{-\|\boldsymbol{p}-\boldsymbol{x}_K\|^2}\right) \tag{12}$$

where $\{\boldsymbol{p}\}_l^i = \left\{\boldsymbol{p}_{l,1}^i, \boldsymbol{p}_{l,2}^i, \dots, \boldsymbol{p}_{l,M_l^i}^i\right\}$ denotes the collection of prototypes at the l^{th} layer of the i^{th} hierarchy.

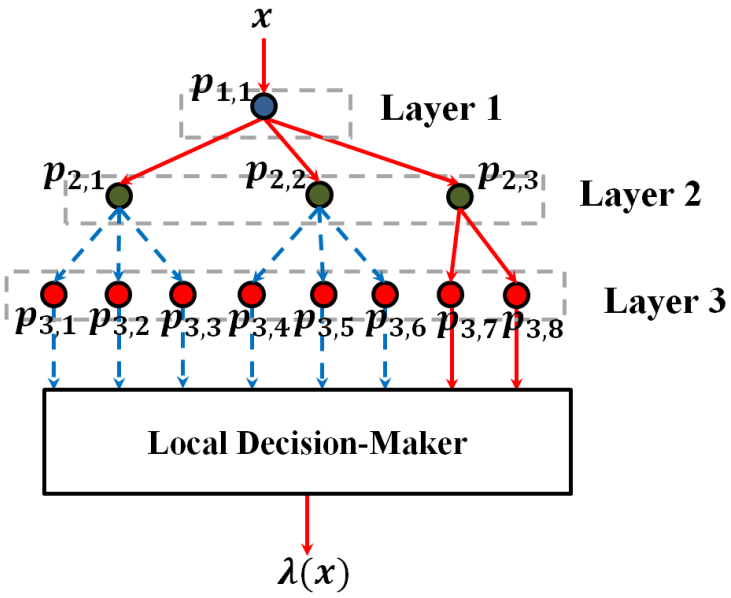
Mode B: alternatively, one can choose to search the nearest prototypes $\boldsymbol{p}_{1,n_1^*}^i, \boldsymbol{p}_{2,n_2^*}^i, \dots, \boldsymbol{p}_{l,n_{Ll}^*}^i$ layer-by-layer from the top to the l^{th} layer of the hierarchy using equation (4), and the final output, namely, the score of confidence is given by:

$$\lambda^i(\boldsymbol{x}_K) = e^{-\left\|\boldsymbol{p}_{l,n_l^*}^i - \boldsymbol{x}_K\right\|^2} \tag{13}$$

An illustrative example is given in Fig. 5 to demonstrate the differences between the two optional nearest prototype searching methods for decision-making. In this example, the bottom layer of the prototype-based hierarchy as given by Fig. 2(b) is selected for producing the score of confidence on the new observation, $\boldsymbol{x}$ as depicted in Fig. 3(a). As one can see from Fig. 5(a), if ***Mode A*** is used, the score of confidence is calculated in a more straightforward way by comparing the similarity between $\boldsymbol{x}$ and all leaf prototypes at the bottom layer, namely, $\boldsymbol{p}_{3,1}, \boldsymbol{p}_{3,2}, \dots, \boldsymbol{p}_{3,8}$ and identifying the most similar one to $\boldsymbol{x}$. In this example, the score of confidence produced by the three-layer hierarchy is obtained as $\lambda(\boldsymbol{x}) = e^{-\|\boldsymbol{p}_{3,7}-\boldsymbol{x}\|^2}$. Otherwise, if ***Mode B*** is adopted (see Fig. 5(b)), the searching algorithm firstly looks for the nearest apex prototype to the new observation, $\boldsymbol{x}$, which is $\boldsymbol{p}_{1,1}$ in this case. Then, the algorithm continues to find the nearest prototype to $\boldsymbol{x}$ from the immediate subordinates of $\boldsymbol{p}_{1,1}$ at the second layer, which include $\boldsymbol{p}_{2,1}, \boldsymbol{p}_{2,2}$ and $\boldsymbol{p}_{2,3}$, and in this example, $\boldsymbol{p}_{2,3}$ is the nearest one. Finally, the algorithm identifies the nearest prototype to $\boldsymbol{x}$ from the leaf prototypes linked with $\boldsymbol{p}_{2,3}$, namely, $\boldsymbol{p}_{3,7}$ and $\boldsymbol{p}_{3,8}$, and calculates the score of confidence accordingly.



(a) ***Mode A***



(b) ***Mode B***

Fig. 5. Illustration of two nearest prototype searching methods for decision-making (the red arrows with concrete lines are the exploited paths during the searching process; the blue arrows with dash lines are the unexploited paths)

After all C pyramidal hierarchies have produced their scores of confidence on $\boldsymbol{x}_K$ (one per class), the class label of $\boldsymbol{x}_K$ is determined following the "winner takes all" principle:

$$label(\boldsymbol{x}_K) \leftarrow class\ i^*; \quad i^* \leftarrow \text{argmax}_{i=1,2,\ldots,C}\left(\lambda^i(\boldsymbol{x}_K)\right) \tag{14}$$

It is worth to be noticed that ***Mode A*** is highly computationally efficient if the upper layers of the HP classifier are used for classification because these layers usually have a smaller number of highly generalized prototypes. If the bottom layers are selected, ***Mode A*** is able to produce more accurate classification results. However, in such cases, the computational efficiency of the HP classifier can be decreased because these layers usually are composed of a larger number of prototypes, especially for large-scale, high-dimensional problems.

In contrast, ***Mode B*** is more suitable for processing large-scale, high-dimensional problems and is highly efficient thanks to the nearest prototype searching technique introduced in this paper (equation (4)). However, it tends to produce more wrong decisions during the decision-making process. This is because that, for a particular

unlabelled sample, if any wrong decision is made during the top-down searching process, ***Mode B*** is unlikely to give a correct final decision.

3.4. Computational Complexity Analysis

In this subsection, computational complexity of the learning and decision-making processes of the proposed HP classifier is analysed. Since the system structure and meta-parameters of the HP classifier are dynamically evolving, the complexity analysis is assumed to be conducted at the K^{th} time instance.

Computational complexity analysis of the learning process

Firstly, let us assume that the data sample observed at the current time instance belongs to the i^{th} class, namely, $\boldsymbol{x}_{K^i}^i$. Because the HP classifier performs self-learning on a sample-by-sample basis and each prototype-based hierarchy is updated separately, one can reasonably expect that the computational complexity of each learning cycle varies a lot depending on the mutual distances between $\boldsymbol{x}_{K^i}^i$ and prototypes of the i^{th} hierarchy. As a result, it is practically impossible to derive an exact expression. Nonetheless, the upper and lower bounds of the computational complexity still can be calculated.

The maximum computational complexity for a particular learning cycle is reached when $\boldsymbol{x}_{K^i}^i$ represents a familiar data pattern and is associated with one of the prototypes at each layer of the hierarchy. In such cases, the HP classifier is required to search the nearest prototype to $\boldsymbol{x}_{K^i}^i$ layer-by-layer in a top-down manner, and the computational complexity of the whole searching process is $O\left(M_0^i N + P_{1,n_1^*}^i N + P_{2,n_2^*}^i N + \cdots + P_{L-1,n_{L-1}^*}^i N\right)$, where $P_{j,n_j^*}^i$ is the number of immediate subordinate prototypes at the $(j+1)^{th}$ layer linked with $\boldsymbol{p}_{j,n_j^*}^i$. In other words, $P_{j,n_j^*}^i$ is the cardinality of $\boldsymbol{\mathcal{L}}_{j,n_j^*}^i$. The computational complexity of the overall prototype updating process is $O(LN)$ (each layer has one prototype being updated). Therefore, the upper bound is $O\left(N\left(M_0^i + \sum_{j=1}^{L-1} P_{j,n_j^*}^i + L\right)\right)$. The lower bound of the computational complexity is reached when $\boldsymbol{x}_{K^i}^i$ is distinctive from the apex prototypes. The only computation is made for calculating the distances between $\boldsymbol{x}_{K^i}^i$ and $\boldsymbol{\mathcal{L}}_0^i$, and thus, the lower bound is $O\left(NM_0^i\right)$.

Therefore, the computational complexity of a particular learning cycle of the HP classifier is between $O\left(NM_0^i\right)$ and $O\left(N\left(M_0^i + \sum_{j=1}^{L-1} P_{j,n_j^*}^i + L\right)\right)$.

Computational complexity analysis of the decision-making process

Let us assume that the l^{th} layer of the HP classifier is used for classification. During the validation stage, for a given unlabelled data sample, $\boldsymbol{x}_K$, each pyramidal prototype hierarchy will produce a score of confidence based on the similarity between $\boldsymbol{x}_K$ and $\{\boldsymbol{p}\}_l^i$ $(i = 1,2,\ldots,C)$. If the HP classifier uses ***Mode A*** for producing the scores of confidence, the computational complexity is $O\left(N\sum_{i=1}^{C} M_l^i\right)$. In contrast, if ***Mode B*** is used, the computational complexity is $O\left(N\sum_{i=1}^{C}\left(M_0^i + \sum_{j=1}^{l-1} P_{j,n_j^*}^i\right)\right)$.

Based on the above computational complexity analysis, one may also conclude that ***Mode B*** is more computational efficient for high-dimensional, large-scale problems. However, for simpler, small-scale problems, ***Mode A*** might be more efficient. This conclusion is further verified via numerical examples presented in the next section.

4. Numerical examples and Discussions

In this section, numerical examples are presented to justify the validity and effectiveness of the proposed concept and method. The overall performance of the HP classifier is evaluated on well-known benchmark datasets and compared with the state-of-the-art approaches. The algorithms were developed using MATLAB2018a, and the performance was evaluated on a desktop with dual core i7 processor 3.60GHz × 2 and 16.0GB RAM. In this paper, by default, the values of r_l $(l = 1,2,\ldots,L)$ of the HP classifier is defined by equation (6) and the value of θ_o is set to be $\frac{\pi}{2}$.

4.1. Demonstration of the Proposed Approach

First of all, a well-known benchmark dataset named banknote authentication[1] is used for illustrating the concept of the proposed approach. This dataset contains 1372 samples, and each data sample has four attributes and one label (class 1 and class 2). There are 762 data samples in class 1 and 610 data samples in class 2. Thanks to its smaller scale and simpler data structure, this dataset is very suitable for demonstration.

In the following numerical example, all data samples are used for training a three-layer HP classifier. Through the training process, the HP classifier self-organizes two prototype-based hierarchies (one per class). For the hierarchy corresponding to class 1, there are two prototypes at the top layer, six prototypes at the second layer and 28 prototypes at the bottom layer. The other hierarchy, which is identified from data samples of class 2, has three prototypes at the top layer, 16 prototypes at the second layer and 47 prototypes at the bottom layer.

The prototypes identified from data samples of the two classes are visualized in Figs. 6(a) and 6(b), respectively. For visual clarity, the first two PCA scores of the original data are used for plotting. In the two figures, dots "·" represent the observed data samples in the data space; asterisks "*" represent the apex prototypes at the top layer of the hierarchy; squares "□" represent the second-layer prototypes; and circles "○" represent the leaf prototypes at the bottom layer. Lines in different colours stand for the links between prototypes of successive layers. The identified prototypes of the two classes are visualized together in the Fig.6(c), where only prototypes at the first two layers are presented for visual clarity. These prototypes are also tabulated in Table 1 for better illustration, and the links between them are given by equations (15a) and (15b), respectively.

$$\begin{cases} \mathcal{L}_0^1 = \{\boldsymbol{p}_{1,1}^1, \boldsymbol{p}_{1,2}^1\} \\ \mathcal{L}_{1,1}^1 = \{\boldsymbol{p}_{2,1}^1, \boldsymbol{p}_{2,2}^1, \boldsymbol{p}_{2,3}^1\} \\ \mathcal{L}_{1,2}^1 = \{\boldsymbol{p}_{2,4}^1, \boldsymbol{p}_{2,5}^1, \boldsymbol{p}_{2,6}^1\} \end{cases} \tag{15a}$$

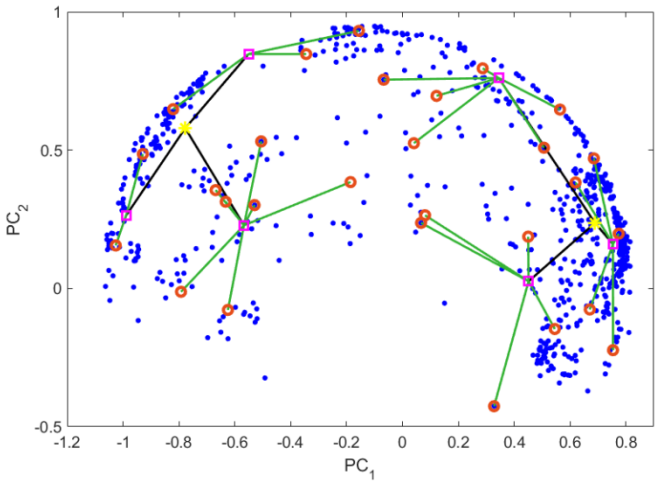
$$\begin{cases} \mathcal{L}_0^2 = \{\boldsymbol{p}_{1,1}^2, \boldsymbol{p}_{1,2}^2, \boldsymbol{p}_{1,3}^2\} \\ \mathcal{L}_{1,1}^2 = \{\boldsymbol{p}_{2,1}^2, \boldsymbol{p}_{2,2}^2, \boldsymbol{p}_{2,3}^2, \boldsymbol{p}_{2,4}^2, \boldsymbol{p}_{2,5}^2\} \\ \mathcal{L}_{1,2}^2 = \{\boldsymbol{p}_{2,6}^2, \boldsymbol{p}_{2,7}^2, \boldsymbol{p}_{2,8}^2, \boldsymbol{p}_{2,9}^2, \boldsymbol{p}_{2,10}^2, \boldsymbol{p}_{2,11}^2\} \\ \mathcal{L}_{1,3}^2 = \{\boldsymbol{p}_{2,12}^2, \boldsymbol{p}_{2,13}^2, \boldsymbol{p}_{2,14}^2, \boldsymbol{p}_{2,15}^2, \boldsymbol{p}_{2,16}^2\} \end{cases} \tag{15b}$$

Table 1. Two-layer prototypes identified through the learning process

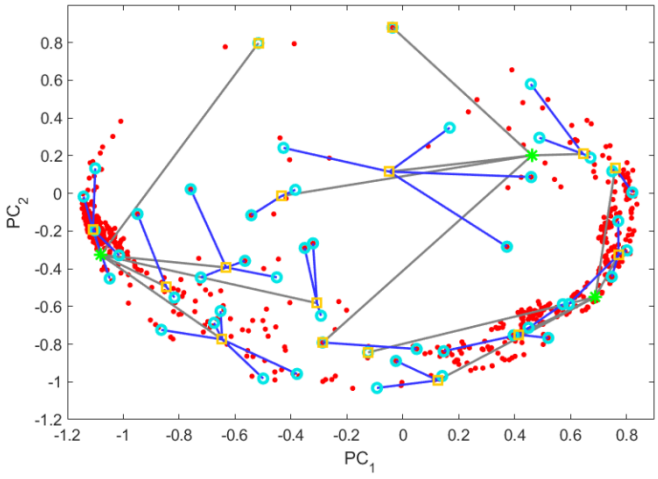
Class	First-Layer Prototypes	Second-Layer Prototypes
1	$\boldsymbol{p}_{1,1}^1 = [0.3282, 0.9237, -0.0673, -0.1856]^T$	$\boldsymbol{p}_{2,1}^1 = [0.2734, 0.9152, -0.1861, -0.2301]^T$
		$\boldsymbol{p}_{2,2}^1 = [0.0736, 0.9234, 0.3183, -0.2017]^T$
		$\boldsymbol{p}_{2,3}^1 = [0.7869, 0.6020, 0.0517, 0.1249]^T$
	$\boldsymbol{p}_{1,2}^1 = [0.5675, -0.2883, 0.7681, 0.0696]^T$	$\boldsymbol{p}_{2,4}^1 = [0.2734, 0.9152, -0.1861, -0.2301]^T$
		$\boldsymbol{p}_{2,5}^1 = [0.0736, 0.9234, 0.3183, -0.2017]^T$
		$\boldsymbol{p}_{2,6}^1 = [0.7869, 0.6020, 0.0517, 0.1249]^T$
2	$\boldsymbol{p}_{1,1}^2 = [-0.3664, 0.6767, -0.4162, -0.4843]^T$	$\boldsymbol{p}_{2,1}^2 = [-0.1083, 0.6222, -0.5949, -0.4972]^T$
		$\boldsymbol{p}_{2,2}^2 = [0.2642, 0.5401, -0.7988, -0.0189]^T$
		$\boldsymbol{p}_{2,3}^2 = [-0.6196, 0.6369, -0.0370, -0.4572]^T$
		$\boldsymbol{p}_{2,4}^2 = [-0.8475, 0.0740, -0.3883, -0.3543]^T$
		$\boldsymbol{p}_{2,5}^2 = [-0.9046, 0.0567, -0.2270, 0.3562]^T$
	$\boldsymbol{p}_{1,2}^2 = [-0.3626, -0.6178, 0.6929, 0.0819]^T$	$\boldsymbol{p}_{2,6}^2 = [0.7167, -0.3291, 0.0860, 0.6088]^T$
		$\boldsymbol{p}_{2,7}^2 = [-0.2090, -0.6682, 0.7135, 0.0285]^T$
		$\boldsymbol{p}_{2,8}^2 = [-0.8294, -0.1319, 0.5366, 0.0820]^T$
		$\boldsymbol{p}_{2,9}^2 = [-0.6622, -0.2736, -0.4052, 0.5678]^T$
		$\boldsymbol{p}_{2,10}^2 = [-0.4282, -0.8982, -0.0993, 0.0044]^T$
		$\boldsymbol{p}_{2,11}^2 = [-0.5886, -0.1169, 0.3471, 0.7207]^T$
	$\boldsymbol{p}_{1,3}^2 = [0.2454, 0.2079, -0.8751, 0.3616]^T$	$\boldsymbol{p}_{2,12}^2 = [-0.0060, -0.6663, -0.5926, 0.4525]^T$

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

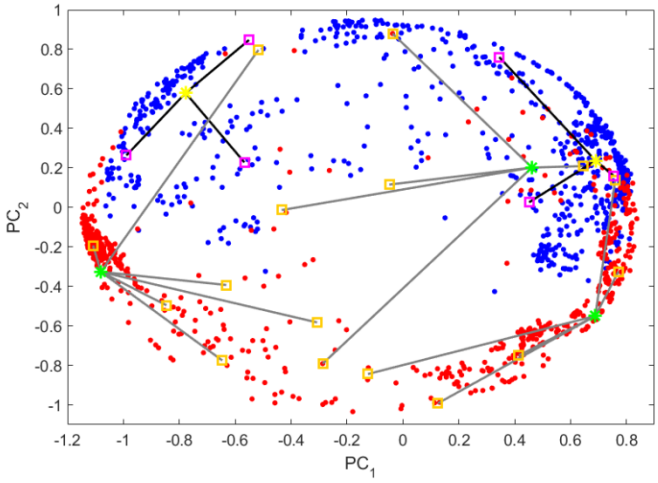
		$\boldsymbol{p}_{2,13}^{2} = [0.3092, 0.3851, -0.8593, 0.1326]^T$
		$\boldsymbol{p}_{2,14}^{2} = [-0.0413, 0.0733, -0.3990, 0.9131]^T$
		$\boldsymbol{p}_{2,15}^{2} = [-0.7843, -0.3472, -0.4515, 0.2460]^T$
		$\boldsymbol{p}_{2,16}^{2} = [0.9092, -0.0624, -0.2575, 0.3212]^T$



(a) Identified prototypes of the class 1



(b) Identified prototypes of the class 2



(c) Identified prototypes of both classes

Fig. 6. The visualization of the identified prototypes of the HP classifier

In the next two subsections, numerical examples on benchmark datasets are presented for performance evaluation. All reported results are obtained after 20 times Monte-Carlo experiments, and the bottom layer of the HP classifier is used for classification unless specifically declared otherwise.

4.2. Performance on Benchmark Numerical Datasets

In this subsection, the following popular benchmark numerical datasets are used:

(1) Multiple feature (MF) dataset[2];

(2) Pen-based handwritten digits recognition (PR) dataset[3];

(3) Epileptic seizure recognition (ES) dataset[4];

(4) Letter recognition (LR) dataset[5], and;

(5) Forest cover type (FC) dataset[6];

Details of the five benchmark datasets are summarized in Table 2.

Table 2. Details of the benchmark numerical datasets for evaluation

Dataset	# Samples	# Attributes	# Classes
MF	2000	649+1 label	10
PR	10996	16+1 label	10
ES	11500	178+1 label	5
LR	20000	16+1 label	26
FC	581012	54+1 label	7

Firstly, the influence of the layer number, namely, L on the performance and system complexity of the HP classifier is investigated. In this numerical example, the four benchmark datasets, namely, MF, PR, ES and LR are used. For each dataset, 50% of the data samples are randomly selected as the training set and the remaining ones are used for validation. Owing to its non-iterative, "one-pass" learning behaviour, there is no randomness

[2] Available at: https://archive.ics.uci.edu/ml/datasets/Multiple+Features
[3] Available at: https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits
[4] Available at: https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition
[5] Available at: https://archive.ics.uci.edu/ml/datasets/Letter+Recognition
[6] Available at: https://archive.ics.uci.edu/ml/datasets/covertype

existing during the training process of the HP classifier. It can be easily concluded that the system structure and meta-parameters of a L_1-layer HP classifier are exactly the same as the top L_1 layers of a L_2-layer HP classifier ($L_1 < L_2$) given the same training samples. In the following numerical example, a six-layer HP classifier ($L = 6$) is trained on the four benchmark datasets. Classification on validation sets is performed by using the porotypes at the first, second, third, fourth, fifth and sixth layers of the HP classifier, individually, which is equivalent to conducting classification with the bottom layer of a one-layer, two-layer, three-layer, four-layer, five-layer or six-layer HP classifier. The classification accuracy and time consumption (in seconds) of the validation process (the two types of nearest prototype searching modes for decision-making as presented in subsection 3.3 are both involved) are reported in Table 3 in the form of $mean \pm standard\ deviation$. The number of prototypes (per class) at each layer of the HP classifier is also tabulated in Table 3.

Table 3. Number of identified prototypes per class during the training process

Dataset	Layer	# Prototypes	***Mode***	Classification Accuracy	Testing Time, s
MF	1	1.00±0.00	***A***	0.7606±0.0101	0.41±0.01
			B		
	2	1.07±0.05	***A***	0.7523±0.0187	0.45±0.02
			B	0.7523±0.0153	1.04±0.05
	3	1.44±0.09	***A***	0.7455±0.0175	0.48±0.04
			B	0.7455±0.0166	1.67±0.05
	4	3.27±0.30	***A***	0.8266±0.0177	0.55±0.09
			B	0.8256±0.0201	2.25±0.18
	5	14.87±0.52	***A***	0.9199±0.0087	0.65±0.05
			B	0.9113±0.0091	2.89±0.16
	6	68.39±1.20	***A***	0.9484±0.0064	1.34±0.19
			B	0.9329±0.0072	3.53±0.27
PR	1	1.00±0.00	***A***	0.7796±0.0008	1.13±0.03
			B		
	2	1.72±0.06	***A***	0.8341±0.0140	1.43±0.10
			B	0.8341±0.0140	3.40±0.17
	3	8.32±0.34	***A***	0.9303±0.0068	1.50±0.09
			B	0.9263±0.0077	5.22±0.24
	4	68.24±1.62	***A***	0.9656±0.0033	1.50±0.02
			B	0.9583±0.0079	6.45±0.19
	5	375.07±3.54	***A***	0.9764±0.0012	2.17±0.22
			B	0.9648±0.0083	8.56±0.55
	6	713.42±1.14	***A***	0.9782±0.0001	2.38±0.07
			B	0.9649±0.0084	9.83±0.20
ES	1	4.68±0.32	***A***	0.3581±0.0086	1.17±0.01
			B		
	2	817.76±6.33	***A***	0.5206±0.0056	24.19±0.62
			B	0.4809±0.0078	4.89±0.11
	3	1113.43±1.94	***A***	0.5429±0.0046	41.10±0.99
			B	0.4874±0.0072	6.53±0.18
	4	1147.47±0.50	***A***	0.5391±0.0047	41.98±0.70
			B	0.4830±0.0068	7.93±0.15
	5	1149.84±0.15	***A***	0.5389±0.0047	41.98±0.70
			B	0.4830±0.0067	9.34±0.17
	6	1150.00±0.00	***A***	0.5389±0.0047	41.88±0.64
			B	0.4830±0.0067	9.34±0.17
LE	1	1.00±0.00	***A***	0.5605±0.0051	7.92±0.04
			B		
	2	1.01±0.01	***A***	0.5604±0.0051	8.04±0.22
			B	0.5604±0.0051	19.73±0.27
	3	3.25±0.11	***A***	0.6057±0.0117	9.42±0.04
			B	0.6056±0.0117	33.60±0.08

	4	28.86±0.79	***A***	0.8862±0.0034	10.27±0.06
			B	0.8760±0.0043	45.52±0.19
	5	183.14±1.15	***A***	0.9401±0.0016	12.15±0.15
			B	0.9180±0.0030	57.83±0.21
	6	348.05±1.22	***A***	0.9414±0.0011	14.19±0.47
			B	0.9153±0.0033	77.50±4.44

As one can see from Table 3, the HP classifier is able to perceive the data space and identify prototypes from data at different levels of granularity. Each layer of the prototype-based hierarchies corresponds to a particular level of granularity. In general, an upper layer of the hierarchy usually contains less but more abstract prototypes, and it is able to perform more efficient but coarser classification. In contrast, a lower layer contains more prototypes and partitions the data space in a finer way. As a result, it can be used for conducting classification with a higher precision. The larger L is, the more detailed data partitioning results the HP classifier can achieve. Nonetheless, if L is too large, the HP classifier will achieve the finest data partitioning, namely, all the data samples are recognized as prototypes at the lower layers of the prototype-based hierarchies. In such cases, the HP classifier reduces to a KNN classifier with k=1. It is also noticeable in Table 3 that the proposed HP classifier demonstrates better classification performance when ***Mode A*** is used during the validation stage. At the same time, for high-dimensional, large-scale problems (e.g. ES dataset), using ***Mode B*** can result in a highly computationally efficient decision-making process. In the rest of this section, the proposed HP classifier uses ***Mode A*** by default because this method performs better in terms of classification accuracy.

To evaluate the influence of the layer number, L on computational efficiency, a HP classifier is trained on the previously used four benchmark datasets with L varying from 1 to 6. The relationship between the training time consumption and the layer number is depicted in Fig. 7. From this figure one can see that, the computational efficiency of the HP classifier is (almost) linearly correlated to the layer number. The more layers the HP classifier has, the more time is consumed during the learning process. However, it is worth to be noticed that if the highest level of granularity for data partitioning has been reached and all the data samples have been recognized as prototypes at the bottom layers, adding extra layers to the HP classifier will not significantly increase the training time consumption.

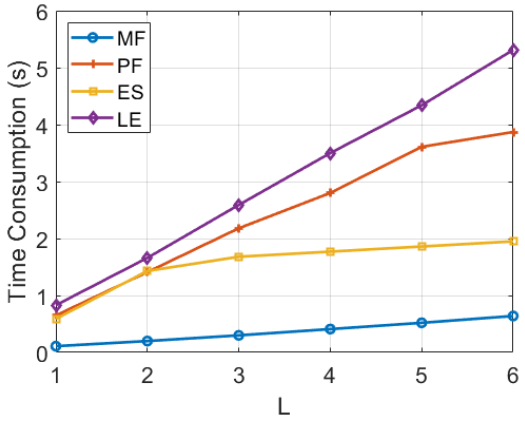


Fig. 7. The influence of layer number on the computational efficiency of the HP classifier

In order to better evaluate the performance of the proposed HP classifier, the following state-of-the-art approaches are involved for comparison:

(1) KNN classifier using Euclidean distance (KNN-E) with $k = 1$ [10];

(2) KNN classifier using cosine dissimilarity (KNN-C) with $k = 1$ [10];

(3) SVM with linear kernel [9];

(4) DT classifier [39];

(5) FNN with three hidden layers, each hidden layer consists of 20 neurons [21];

(6) Long short-term memory (LSTM) with three hidden layers, each hidden layer consists of 20 neurons [16];

(7) LVQ with one hidden layer consisting of 32 neurons [25];

(8) SOM classifier using "winner takes all" principle with the net size of 9×9 [34];

(9) Extended sequential adaptive fuzzy inference system (ESAFIS) [35];

(10) Zero-order autonomous learning multiple-model (ALMMo0) classifier [2];

(11) Self-organizing fuzzy (SOF) classifier using Euclidean distance with the level of granularity $G = 12$ [17], and;

(12) eClass0 classifier [1].

Note that both, KNN and SVM are two main generic classifiers used by the pre-trained DNN-based approaches [33], and they are capable of producing highly accurate classification results. ALMMo0, SOF and eClass0 classifiers are all zero-order EISs, and they have a prototype-based nature same as the proposed HP classifier. In these experiments, a six-layer HP classifier ($L = 6$) is used. The same experimental protocol as used in the previous numerical examples are adopted. The performance of these classification approaches in terms of accuracy and training time consumption (in seconds) on the four benchmark problems is presented in Table 4, where the training time consumptions of the KNN classifiers are not reported because they literally require no training.

Table 4. Performance comparison on benchmark numerical datasets

Dataset	Algorithm	Classification Accuracy	Training Time, s
MF	HP	0.9484±0.0064	0.64±0.05
	KNN-E	0.9434±0.0051	
	KNN-C	0.9489±0.0056	
	SVM	**0.9704±0.0062**	20.35±4.69
	DT	0.9263±0.0107	**0.14±0.02**
	FNN	0.8658±0.0380	0.49±0.18
	LSTM	0.2042±0.0471	6.43±1.75
	LVQ	0.6533±0.0209	69.80±2.35
	SOM	0.8790±0.0079	40.62±1.63
	ESAFIS	0.5324±0.1383	360.31±74.51
	ALMMo0	0.9343±0.0055	0.11±0.01
	SOF	0.9226±0.0092	0.06±0.02
	eClass0	0.7985±0.0060	2.54±0.11
PR	HP	**0.9782±0.0001**	3.87±0.07
	KNN-E	0.9774±0.0000	
	KNN-C	0.9780±0.0000	
	SVM	0.9551±0.0000	58.65±0.90
	DT	0.9122±0.0003	**0.04±0.03**
	FNN	0.9177±0.0180	0.92±0.17
	LSTM	0.8218±0.0280	34.29±7.82
	LVQ	0.8217±0.0018	338.83±7.10
	SOM	0.9126±0.0067	9.76±0.58
	ESAFIS	0.9197±0.0134	38.80±5.93
	ALMMo0	0.9752±0.0023	0.56±0.05
	SOF	0.9763±0.0000	0.39±0.03
	eClass0	0.7630±0.0001	0.73±0.08
ES	HP	0.5389±0.0047	1.95±0.02
	KNN-E	0.5050±0.0046	
	KNN-C	0.5389±0.0047	

	SVM	0.2220±0.0066	363.20±1.99
	DT	0.4588±0.0048	**0.69±0.03**
	FNN	0.4263±0.0611	0.82±0.28
	LSTM	0.3257±0.0101	13.23±7.35
	LVQ	0.2000±0.0000	285.97±0.61
	SOM	0.3743±0.0091	66.39±0.83
	ESAFIS	0.2597±0.0246	266.17±83.10
	ALMMo0	**0.5401±0.0053**	5.53±0.05
	SOF	0.4893±0.0073	0.77±0.03
	eClass0	0.2848±0.0067	3.20±0.17
LR	HP	0.9414±0.0011	5.31±0.26
	KNN-E	**0.9426±0.0016**	
	KNN-C	0.9415±0.0011	
	SVM	0.8540±0.0030	15.77±0.54
	DT	0.8255±0.0048	**0.06±0.02**
	FNN	0.4777±0.0354	1.55±0.14
	LSTM	0.4481±0.0418	53.03±10.03
	LVQ	0.0379±0.0011	437.47±0.56
	SOM	0.4645±0.0100	16.80±0.14
	ESAFIS	0.0394±0.0000	9.32±0.22
	ALMMo0	0.9197±0.0031	0.59±0.04
	SOF	0.9297±0.0021	0.24±0.03
	eClass0	0.4915±0.0040	0.87±0.07

In the following numerical example, the quality of the identified prototypes by the HP classifier is examined from the data partitioning point of view, and the four datasets, MF, PR, ES and LR are used for evaluation. During the experiments, a six-layer HP classifier ($L = 6$) is firstly trained with the whole datasets to extract prototypes from data with different levels of granularity. After that, Voronoi tessellations are created by using the identified prototypes at different layers to attract nearby data samples of the same classes forming data clouds. This results in six different data partitioning results (one per layer), each one corresponds to one level of granularity.

Then, three clustering quality indices, namely, Calinski-Harabasz [6], Davies-Bouldin [11] and Silhouette [36], are used for evaluating the quality of the data partitioning result of each layer, and the calculated index values are reported in Table 5. For better evaluation, the following widely used data partitioning/clustering algorithms are involved for comparison:

(1) Autonomous data partitioning (ADP) algorithm [18];

(2) Mean shift (MS) clustering algorithm [8];

(3) Subtractive (Sub) clustering algorithm [7];

(4) Affinity propagation (AP) clustering algorithm [14];

(5) DBSCAN clustering algorithm [13], and;

(6) Nonparametric mixture model (NMM) clustering algorithm [4].

In this example, the offline version of the ADP algorithm is used; the bandwidth of the MS algorithm is set to be 0.15; the initial cluster radius of the Sub algorithm is set to be 0.3 as recommended by [7]; for the AP algorithm, the maximum number of iterative refinements is 200, the cumulative number of iterations for monitoring the exemplar decision is set as 20, and the dampening factor is equal to 0.5; the minimum number of data samples within the radius is set to be 4 for the DBSCAN algorithm, and the cluster radius is set to be the value of the knee point of the sorted 4-dist graph as recommended by [13]; the NMM algorithm assumes that the data samples follow Gaussian distribution, its prior scaling parameter is set to be 1, and the maximum number of iterative refinements is 200. All the comparative algorithms use the same experimental protocol by identifying the data clouds/clusters from data samples of different classes, separately. For the Calinski-Harabasz and Silhouette indices, a higher value indicates a better clustering result. However, it is the opposite for the Davies-

Bouldin index that a lower value indicates a better clustering result. The best results are bolded in Table 5. It is also noticeable that occasionally, the clustering indices give abnormal values such as: 0.0000, 1.0000, not a number (nan) and infinite (inf). Such cases are in italic in Table 5 and should not be taken into consideration.

As we can see from Table 5, the HP classifier is able to identify prototypes from data with different levels of granularity and partitions the data space with high-quality data clouds surpassing the alternative approaches. This is an important feature showing the strong adaptive ability and flexibility of the proposed approach. By perceiving complex problems at different levels of specificity and self-organising a multi-layer structure through an "one pass" learning process, the HP classifier significantly simplifies complex problems and presents users the learned knowledge in a human-understandable prototype-based hierarchical form with different levels of details all in one. It allows users to choose the lower layers with lots of finer details for performing accurate classification or/and the easy-to-interpret upper layers for coarse classification. Users can determine the most suitable layer for a particular problem depending on preferences and/or specific requirements.

Table 5. Quality comparison between the formed clusters/data clouds by different algorithms

Dataset	Algorithm		# Clusters	Calinski-Harabasz	Davies-Bouldin	Silhouette
MF	HP	Layer 1	10.0	**1038.3387**	2.1662	0.1115
		Layer 2	11.0	1012.9386	2.6062	0.1048
		Layer 3	16.3	832.1617	2.5744	0.0448
		Layer 4	39.4	690.0422	1.7152	0.1311
		Layer 5	203.0	330.6662	1.1416	0.1678
		Layer 6	1182.0	129.1651	**0.5979**	**0.5118**
	ADP		408.0	282.1538	1.5072	0.1517
	MS		1994.0	*inf*	*0.0000*	*1.0000*
	Sub		1994.0	*inf*	*0.0000*	*1.0000*
	AP		52.0	976.3070	1.8232	0.2131
	DBSCAN		24.0	559.7079	3.8972	0.0742
	NMM		10.0	**1038.3387**	2.1662	0.111
PR	HP	Layer 1	10.0	1404.4659	2.3209	0.2187
		Layer 2	18.9	**1408.9932**	1.5576	0.2516
		Layer 3	97.7	778.4346	1.3151	0.2758
		Layer 4	820.4	262.97307	1.1739	0.2099
		Layer 5	5001.0	131.5285	0.6967	0.3932
		Layer 6	10297.2	231.9914	**0.2308**	**0.9260**
	ADP		1022.0	243.7096	1.3187	0.2524
	MS		2627.1	74.8120	0.7123	0.0535
	Sub		3957.0	65.0035	0.8365	0.2893
	AP		303.0	535.0103	1.5422	0.2292
	DBSCAN		393.0	670.5420	2.3932	0.1748
	NMM		79.7	695.2818	1.9006	0.1441
ES	HP	Layer 1	24.3	**49.7753**	6.6989	-0.3027
		Layer 2	7385.9	5.7619	0.9858	0.4832
		Layer 3	10925.7	11.97795	0.5728	0.9536
		Layer 4	11450.1	27.7188	0.2917	0.9976
		Layer 5	11497.1	43.4237	**0.0526**	**0.9999**
		Layer 6	11500.0	*nan*	*0.0000*	*1.0000*
	ADP		564.0	3.2650	7.0981	-0.3561
	MS		9790.2	18.0079	0.6878	0.7633
	Sub		9186.0	4.4166	1.1445	0.6347
	AP		1945.0	1.4213	3.1027	-0.3277
	DBSCAN		10.0	3.3213	36.4996	-0.2135
	NMM		28.9	6.3598	17.3070	-0.4429
LE	HP	Layer 1	26.0	382.5708	4.3511	-0.0129
		Layer 2	26.4	385.3833	4.2978	-0.0112
		Layer 3	87.0	**407.6502**	2.6115	0.0255

		Layer 4	858.8	197.5088	1.5564	0.2136
		Layer 5	7192.0	80.4005	0.8229	0.3717
		Layer 6	17111.8	225.1629	**0.2304**	**0.9149**
	ADP		1867.0	155.3645	1.3385	0.3105
	MS		4808.6	45.5177	0.7478	0.2036
	Sub		1343.0	69.7540	1.6665	-0.0647
	AP		831.0	212.9784	1.5094	*nan*
	DBSCAN		215.0	148.7575	2.3189	-0.1930
	NMM		150.0	225.2300	2.6303	-0.1008

Finally, numerical example is presented on the FC dataset to evaluate the performance of the proposed HP classifier on very large-scale numerical datasets. In this example, 50% of data samples of this dataset are randomly selected as the training set and the remaining ones are used for validation. The classification performance of a six-layer HP classifier ($L = 6$) in terms of accuracy, training and testing time consumptions (in seconds) is reported in Table 6. Here, the two nearest prototype searching modes for decision-making, namely, ***Mode A*** and ***Mode B***, are both involved. Alternative classification approaches as used in the numerical examples presented in Table 4 are involved for comparison, and their results are also tabulated in Table 6. However, LSTM and LVQ are not used in this example because their computational efficiency is significantly lower on large-scale datasets.

Table 6. Performance comparison on the FC dataset

Algorithm		Classification Accuracy	Training Time, s	Testing Time, s
HP	***Mode A***	0.9069±0.0006	544.95±36.85	7528.65±433.48
	Mode B	0.8711±0.0042		1459.61±18.14
KNN-E		**0.9331±0.0004**		6479.94±170.76
KNN-C		**0.9331±0.0004**		6511.59±217.21
SVM		0.7247±0.0008	4641.64±247.81	2605.98±161.16
DT		0.9180±0.0008	**37.82±6.59**	**0.36±0.04**
FNN		0.7186±0.0006	104.14±1.49	1.40±0.09
SOM		0.6335±0.0008	2596.12±120.83	788.05±114.33
ESAFIS		0.7359±0.0037	5479.86±1288.21	5.97±1.12
ALMMo0		0.8932±0.0005	1717.16±166.51	4138.93±210.86
SOF		0.9242±0.0005	6916.28±444.19	14181.74±1051.70
eClass0		0.3456±0.0005	73.95±2.04	26.20±3.12

As one can see from Table 6, during the training stage, the HP classifier is at least two times more computationally efficient than alternative prototype-based approaches with similar levels of precision, namely, ALMMo0 and SOF. Its computational efficiency is also four times higher than SOM and 10 times higher than SVM. One may notice that both KNN classifiers produce the most accurate classification results on this problem. This indicates that the classification accuracy of the HP classifier can be further improved by adding more layers to it. It is also noticeable that during the decision-making stage, the computational efficiency of the HP classifier is four times higher if ***Mode B*** is used. In contrast, using ***Mode A*** results in stronger classification performance in terms of accuracy.

4.3. Performance on Benchmark Image Sets

In this subsection, the following popular benchmark image sets are used for further evaluating the performance of the proposed HP classifier on image classification problems:

(1) MNIST dataset[7];

(2) UCMerced dataset[8];

[7] Available at: http://yann.lecun.com/exdb/mnist/

(3) WH-RS19 dataset[9];

(4) Caltech101 dataset[10], and;

(5) Caltech256 dataset[11];

MNIST is a large-scale image set for hand-written digits recognition (from "0" to "9"). MNIST dataset is composed of a training set and a validation set. The train set contains 60000 grey-level images and the validation set contains 10000 grey-level images. All the images have the uniform size of 28×28 pixels. The amounts of training and validation images of the 10 classes are more or less equal. Example images of this image set are given in Fig. 8(a) for illustration.

UCMerced image set is a well-known benchmark in the remote sensing domain [42]. This dataset consists of fine spatial remote sensing images of 21 challenging land-use categories (including 1) agricultural; 2) airplane; 3) baseball diamond; 4) beach; 5) buildings; 6) chaparral; 7) dense residential; 8) forest; 9) freeway; 10) golf course; 11) harbour;12) intersection; 13) medium residential;14) mobile home park; 15) overpass;16) parking lot; 17) river; 18) runway; 19) sparse residential; 20) storage tanks; and 21) tennis courts). Each category contains 100 images with the uniform size of 256×256 pixels. Example images of the UCMerced image set are presented in Fig. 8(b).

WH-RS19 is a popular benchmark image set collected from Google Earth (Google Inc.). It consists of 950 images with a size of 600×600 pixels. WH-RS19 has 19 different land-use categories, which include 1) airport; 2) beach; 3) bridge; 4) commercial; 5) desert; 6) farmland; 7) football field; 8) forest; 9) industrial; 10) meadow; 11) mountain; 12) park; 13) parking lot; 14) pond; 15) port; 16) railway; 17) residential; 18) river; and 19) viaduct, with 50 images in each. This image set contains aerial images with high variations in terms of illumination, scale resolution, etc., and, thus, is a challenging problem. Example images of the WH-RS19 dataset are presented in Fig. 8(c).

Caltech101 dataset has 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 and has 256 classes. The minimum number of images per class for Caltech256 dataset is 80, and in total, there are 29780 images. The two datasets contain both, 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 challenging problems. The images of both datasets are very uniform in presentation, aligned from left to right and usually not occluded. The example images of two datasets are given in Fig. 8(d) and 8(e).

In this paper, for the MNIST dataset, the GIST feature descriptor [32] is employed to extract 512×1 dimensional feature vectors from the original handwritten digit images. For the UCMerced, WH-RS19, Caltech101 and Caltech256 datasets, a high-level ensemble descriptor using the pre-trained AlexNet [26] and VGG-VD-16 [38] models is created for feature extraction, which results in a 9192×1 dimensional discriminative representation denoted by $\boldsymbol{x}$ from each image, $\mathbf{I}$:

$$\boldsymbol{x} = \mathrm{F}(\mathbf{I}) = \left[\frac{\mathrm{AN}(\mathbf{I})}{\|\mathrm{AN}(\mathbf{I})\|}, \frac{\mathrm{VN}(\mathbf{I})}{\|\mathrm{VN}(\mathbf{I})\|}\right]^T \tag{16}$$

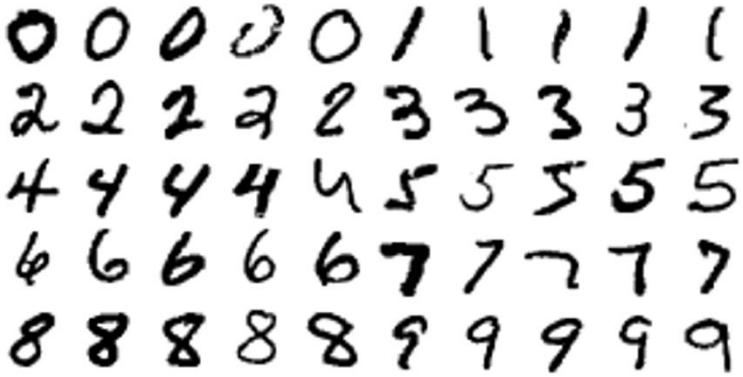
where $\mathrm{F}(\mathbf{I})$ represents 9192×1 dimensional representation extracted from $\mathbf{I}$ by the ensemble feature descriptor; $\mathrm{AN}(\mathbf{I})$ and $\mathrm{VN}(\mathbf{I})$ are the 1×4096 dimensional activations extracted from the first fully connected layers of the AlexNet [26] and VGG-VD-16 [38] models, respectively. In particular, the commonly used "centre, four corners and horizontal flipping" data augmentation technique is applied on UCMerced and WH-RS19 datasets, and the mean of feature vectors of the 10 sub-images created from each remote sensing image is used as the feature vector of the image [26]. Images of WH-RS19 dataset have been rescaled to the size of 400×400 pixels in advance to avoid loss of information during the data augmentation process. However, it has to be stressed that there is no image augmentation technique applied to MNIST, Caltech101 and Caltech256

[8] Available at: http://weegee.vision.ucmerced.edu/datasets/landuse.html
[9] Available at: http://captain.whu.edu.cn/repository.html
[10] Available at: http://www.vision.caltech.edu/Image_Datasets/Caltech101/
[11] Available at: http://www.vision.caltech.edu/Image_Datasets/Caltech256/

datasets in the numerical examples presented in this subsection. The details of the image sets are given in Table 7.

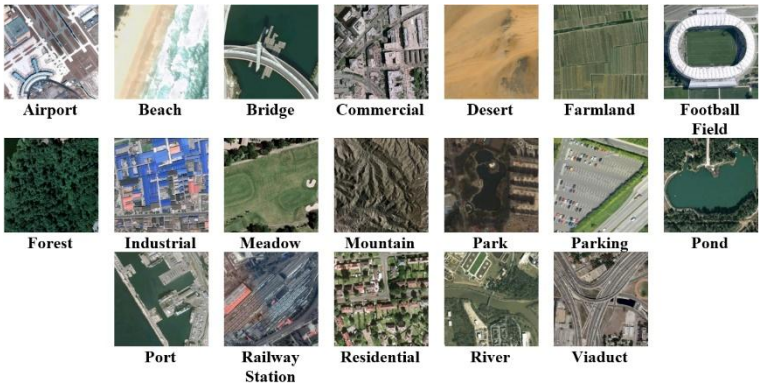
Table 7. Details of the benchmark numerical datasets for evaluation

Image Set		# Images	# Classes	# Pixels	# Features
MNIST	Training	60000	10	28×28	512×1
	validation	10000			
UCMerced		2100	21	256×256	9192×1
WH-RS19		950	19	600×600	
Caltech101		8677	101	Roughly 200×300	
Caltech256		29780	256		

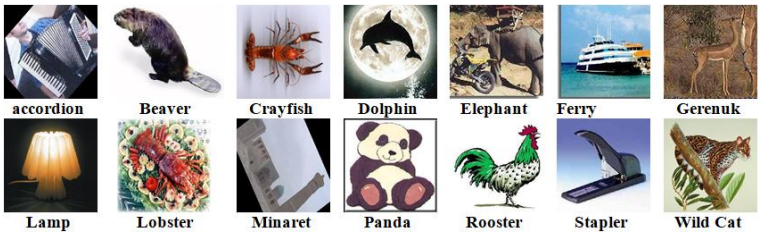


(a) MNIST

(b) UCMerced



(c) WH-RS19



(d) Caltech101

(e) Caltech256

Fig. 8. Example images of the benchmark image sets for performance evaluation

Firstly, the performance of the proposed approach is tested on the MNIST dataset. In this experiment, a four-layer HP classifier ($L = 4$) is trained on the training sets of different size (5000, 10000, 20000, 30000, 40000, 50000 and 60000 images), and then, each individual layer is used for testing on the validation set. Statistical performance in terms of average number of prototypes (per class) at each layer and classification accuracy is reported in Table 8.

From Table 8 one can see that the bottom layer of the HP classifier is able to classify the unlabelled handwritten digits with the highest accuracy. Meanwhile, one may also notice that the bottom layer has a huge number of prototypes. This is due to the very high variability demonstrated by these handwritten digit images in the MNIST, where even images of the same class can vary a lot.

Table 8. The statistical performance of the HP classifier

#Training Images	Layer	# Prototypes	Classification Accuracy
5000	1	1.00±0.00	0.8973±0.0030
	2	1.79±0.18	0.8807±0.0069
	3	89.93±2.65	0.9643±0.0013
	4	466.94±2.13	0.9731±0.0015
10000	1	1.00±0.00	0.8985±0.0024
	2	1.98±0.16	0.8818±0.0090
	3	129.95±3.81	0.9698±0.0013
	4	920.49±3.30	0.9784±0.0011
20000	1	1.00±0.00	0.8987±0.0016
	2	2.28±0.12	0.8852±0.0115
	3	186.33±4.53	0.9738±0.0015
	4	1806.01±4.60	0.9828±0.0001
30000	1	1.00±0.00	0.8988±0.0011
	2	2.44±0.15	0.8899±0.0067
	3	228.84±4.97	0.9754±0.0011
	4	2673.84±5.96	0.9845±0.0001

40000	1	1.00±0.00	0.8988±0.0009
	2	2.54±0.15	0.8935±0.0049
	3	263.31±4.70	0.9763±0.0010
	4	3527.52±5.71	0.9855±0.0001
50000	1	1.00±0.00	0.8988±0.0007
	2	2.62±0.15	0.8949±0.0045
	3	292.96±6.01	0.9774±0.0009
	4	4369.10±5.41	0.9860±0.0001
60000	1	1.00±0.00	0.8989±0.0002
	2	2.71±0.16	0.8958±0.0047
	3	318.35±6.30	0.9777±0.0010
	4	5202.36±5.34	0.9864±0.0001

Furthermore, the performance of the HP classifier is compared with the following comparative algorithms in terms of classification accuracy and training time consumption (in seconds):

(1) KNN-C with $k = 1$ [10];

(2) SVM with linear kernel [9];

(3) DT classifier [39];

(4) Deep rule-based (DRB) classifier [19];

(5) SOF classifier using cosine dissimilarity with the level of granularity $G = 12$ [17];

(6) eClass1 classifier [1], and;

(7) TEDAClass classifier [23].

Note that cosine dissimilarity is more effective than Euclidean distance on high-dimensional problems [17] and, thus, both KNN and SOF classifiers use cosine dissimilarity in the numerical experiments presented in this subsection. The comparison on classification accuracy is tabulated in Table 9, and the comparison on training time consumption is depicted in Fig. 9, where the same four-layer HP classifier is used for the experiments.

Table 9. Performance comparison on MNIST dataset in terms of accuracy

# Training Images	Classification Accuracy							
	HP	KNN-C	SVM	DT	DRB	SOF	eClass1	TEDAClass
5000	0.9731	0.9679	**0.9733**	0.8197	0.9712	0.9679	0.9685	0.9716
10000	0.9784	0.9747	**0.9786**	0.8459	0.9769	0.9747	0.9719	0.9738
20000	**0.9828**	0.9795	0.9819	0.8692	0.9815	0.9795	0.9732	0.9752
30000	**0.9845**	0.9816	0.9835	0.8803	0.9836	0.9816	0.9746	0.9768
40000	**0.9855**	0.9827	0.9846	0.8890	0.9848	0.9827	0.9745	0.9766
50000	**0.9860**	0.9833	0.9851	0.8940	0.9856	0.9833	0.9746	0.9765
60000	**0.9864**	0.9835	0.9857	0.9010	0.9864	0.9835	0.9746	0.9763

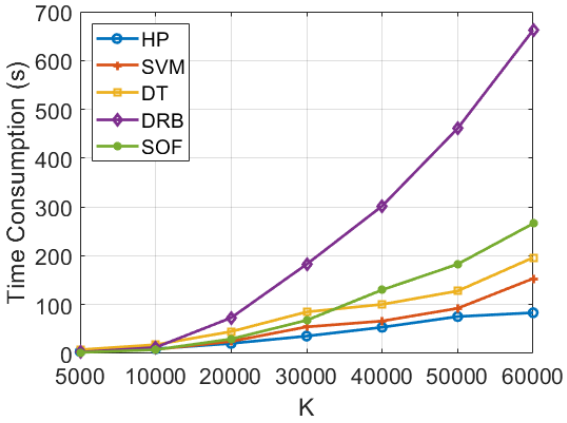


Fig. 9. Comparison on training time consumption between different algorithms on MNIST image set

From Table 9 one can see that the proposed HP classifier demonstrates very high classification performance surpassing or, at least, on par with comparative algorithms. Moreover, Fig. 9 further demonstrates that the computational efficiency of the proposed HP classifier is much higher than the alternatives, especially, on large-scale, high-dimensional problems.

In the following numerical examples, the performance of the HP classifier is tested on UCMerced and WH-RS19 image sets. Following the common practice [42], the ratios between the training and validation images of the UCMerced dataset are set to be 50% and 80%; the ratios of the WH-RS19 dataset are set as 40% and 60%, respectively. Since the scales of the two remote sensing image sets are smaller than the MNIST image set, a three-layer HP classifier ($L = 3$) is trained. The accuracy rates of the classification results by the HP classifier on the two image sets are tabulated in Tables 10 and 11, respectively. The KNN-C, SVM and DRB classifiers as used in the previous example are involved for comparison, and the classification performance of the competitors is reported in Tables 10 and 11 as well. Selected state-of-the-art results reported by recent publications are also given for informed comparison. The average number of the prototypes (per class) at each layer of the HP classifier is reported in Fig. 10.

Table 10. Performance comparison on UCMerced image set

Algorithm	Classification Accuracy	
	50% Training Images	80% Training Images
HP	0.9329±0.0092	0.9594±0.0151
KNN-C	0.9090±0.0101	0.9359±0.0141
SVM	**0.9425±0.0080**	**0.9601±0.0112**
DRB	0.9313±0.0087	0.9580±0.0146
RCNet [49]		0.9453
CaffeNet [42]	0.9398±0.0067	0.9502±0.0081
GoogLeNet [42]	0.9270±0.0060	0.9431±0.0089
VGG-VD-16 [42]	0.9414±0.0069	0.9521±0.0120
BoVW(SIFT) [42]	0.7190±0.0079	0.7412±0.0330
VLAD(SIFT) [42]	0.7323±0.0102	0.7819±0.0166
MS-CLBP+FV [22]	0.8876±0.0079	0.9300±0.0120
salM^{3}LBP-CLM [4]	0.9421±0.0075	0.9575±0.0080
salM^{3}LBP [4]	0.8997±0.0085	0.9314±0.0100
salCLM (eSIFT) [4]	0.9293±0.0092	0.9452±0.0079

Table 11. Performance comparison on WH-RS19 image set

Algorithm	Classification Accuracy	
	40% Training Images	60% Training Images
HP	0.9320±0.0069	0.9470±0.0099
KNN-C	0.9264±0.0083	0.9379±0.0097
SVM	0.9458±0.0105	0.9577±0.0100
DRB	0.9335±0.0098	0.9470±0.0085
CaffeNet [42]	0.9511±0.0120	0.9624±0.0056
GoogLeNet [42]	0.9312±0.0082	0.9471±0.0133
VGG-VD-16 [42]	**0.9544±0.0060**	0.9605±0.0091
BoVW (SIFT) [42]	0.7526±0.0139	0.8013±0.0201
VLAD (SIFT) [42]	0.7637±0.0201	0.8082±0.0215
salM^{3}LBP-CLM [4]	0.9535±0.0076	**0.9638±0.0082**
salM^{3}LBP [4]	0.8974±0.0184	0.9258±0.0089
salCLM (eSIFT) [4]	0.9381±0.0091	0.9592±0.0095
MS-CLBP+FV [22]		0.9453±0.0102

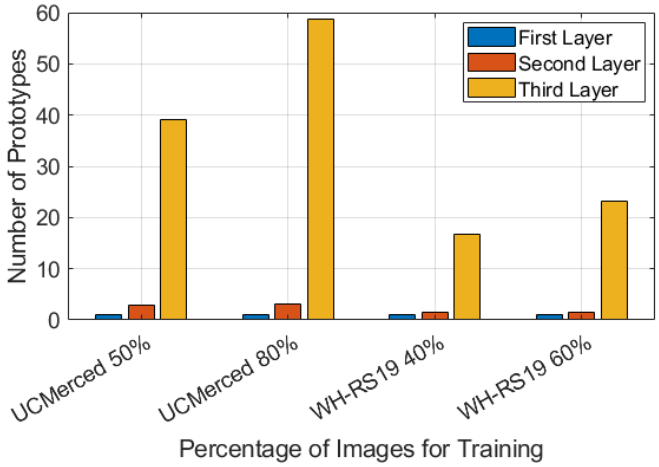


Fig. 10. The average number of prototypes at each layer of the HP classifier trained on remote sensing image sets

Finally, Caltech101 and Caltech256 datasets are used for numerical examples. Following the common practice [44], for Caltech101 image set, 15 and 30 images are randomly selected out from each category for training respectively, and the rest of the dataset is used for validation. For Caltech256 image set, 15, 30, 45 and 60 images are randomly selected from each category for training, respectively, and the rest is used for validation. A two-layer HP classifier ($L = 2$) is used for the experiments because of the smaller scale and less variability of training images in each category. The statistical performance of the HP classifier in terms of classification accuracy is reported in Tables 12 and 13, respectively. The average number of prototypes of each layer (per class) is given in Fig. 11. Similarly, the KNN-C, SVM and DRB classifiers as used in the previous example are involved for comparison, and the selected state-of-the-art results reported by recent publications are also given for a better evaluation.

Table 12. Performance comparison on Caltech101 image set

Algorithm	Classification Accuracy	
	15 Training Images	30 Training Images
HP	**0.8863±0.0060**	**0.9224±0.0040**

KNN-C	0.8670±0.0063	0.8999±0.0051
SVM	0.8729±0.0104	0.9027±0.0087
DRB	0.8491±0.0062	0.8864±0.0047
ICAC [45]	0.7148±0.0056	0.7663±0.0079
CASE-LLC-SVM [29]	0.6400±0.0040	0.7140±0.0120
LLC [41]	0.6543	0.7344
ScSPM [44]	0.6700±0.0045	0.7320±0.0054
DEFEATnet [15]	0.7128±0.0061	0.7760±0.0096

Table 13. Performance comparison on Caltech256 image set

Algorithm	Classification Accuracy			
	15 Training Images	30 Training Images	45 Training Images	60 Training Images
HP	**0.6353±0.0045**	**0.6908±0.0029**	**0.7172±0.0034**	**0.7347±0.0030**
KNN-C	0.6249±0.0033	0.6723±0.0026	0.6986±0.0032	0.7210±0.0027
SVM	Out of System Memory			
DRB	0.6239±0.0033	0.6711±0.0026	0.6976±0.0034	0.7187±0.0029
DEFEATnet [15]	0.3507±0.0038	0.4206±0.0025	0.4598±0.0026	0.4852±0.0032
ScSPM [44]	0.2773±0.0051	0.3402±0.0035	0.3746±0.0055	0.4014±0.0091
LSC-LG [47]	0.4314±0.0063	0.5062±0.0053	0.5327±0.0056	0.5576±0.0048
LLC [41]	0.2776±0.0032	0.3207±0.0024	0.3509±0.0044	0.4014±0.0091
FV [37]	0.3850±0.0020	0.4740±0.0010	0.5210±0.0040	0.5480±0.0040
OCB-FV [50]	0.4403±0.0046	0.5315±0.0044	0.5784±0.0040	0.5903±0.0045
SWSS-DeCAF [46]	0.6152±0.0039	0.6768±0.0065	0.6977±0.0053	0.7283±0.0044
SWSS-FV [46]	0.4246±0.0038	0.4985±0.0042	0.5466±0.0047	0.5652±0.0041
SC^2-CNN [48]	0.4758±0.0062	0.5542±0.0056	0.5912±0.0051	0.6174±0.0050

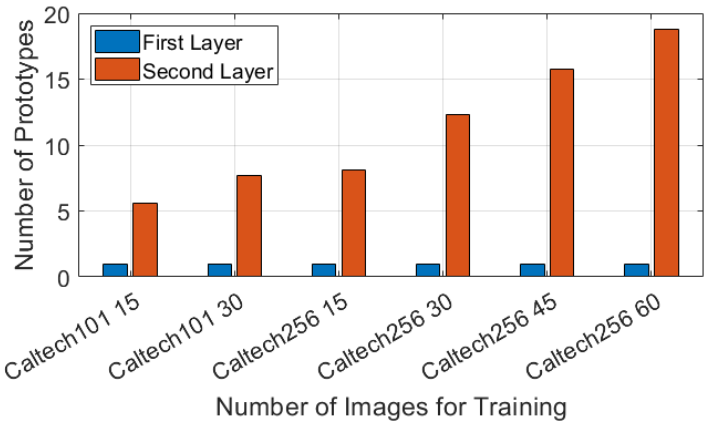


Fig. 11. Number of prototypes at each layer of the HP classifier trained on Caltech101 and Caltech256 image sets

4.4. Discussions

As we can see from the numerical examples presented in this section, the proposed approach outperforms the comparative approaches on various benchmark classification problems. Its computational efficiency is also extremely high, especially for high-dimensional, large-scale, complex problems. Thanks to its unique hierarchal system structure and prototype-based nature, the HP classifier is able to effectively handle complex problems by approaching them at different levels of granularity and can further present the learning results in the form of prototype-based hierarchies. In addition, the HP classifier also puts users "in the driving seat" by allowing users to determine the layer number for the system structure and choose the method for decision-making. Experienced users can further change the settings of the HP classifier to meet various preferences or problem requirements, for example, by adjusting the values of radii r_l $(l = 1,2, \dots, L)$ of influence areas or specifying a layer number for each hierarchy. The strong flexibility significantly strengthens the adaptive ability of the proposed approach in real-world applications. Therefore, one can conclude that the proposed HP classifier is a strong alternative to

the state-of-the-art approaches for classification, and its unique advantages make it highly attractive for large-scale, high-dimensional and complex problems.

Nonetheless, it has to be admitted that there are still few issues for future improvements. Firstly, there is no scientific way at this moment to determine the most suitable layer number for the HP classifier to perform the best on a particular problem. Users have to either rely on their experience and expertise or make multiple attempts to obtain the best setting. Secondly, the system structure and meta-parameters of the HP classifier are self-evolving and self-updating all the time with new data samples. However, if one wants to add an extra layer to the system structure, the whole HP classifier has to be re-trained fully. Thirdly, the radii of influence areas of prototypes at different layers of the HP classifier are hard-coded in advance at this moment.

5. Conclusion

This paper presents a novel hierarchical prototype-based (HP) approach for classification. The proposed approach has a highly transparent hierarchical system structure composed of meaningful prototypes. These prototypes are identified through an autonomous, non-iterative learning process, and they naturally represent the local peaks of multimodal distributions derived from data at different levels of granularity. The HP classifier can start working from the very first training example per class, and continuously learn from new observations in a computationally lean, "one pass" manner and dynamically self-evolve to follow the rapidly changing data patterns. Thanks to its prototype-based nature, the learning and decision-making processes of the HP classifier are fully interpretable, traceable, explainable to/by humans. The proposed approach is also capable of visualising the learned knowledge, namely, the identified prototypes with different levels of details and the meaningful links between them, in a human-understandable hierarchical form naturally. Users can use either the more sophisticated lower layers of the HP classifier for finer classification or/and the more generalized upper layers to quickly understand complex problems and perform coarse classification. Numerical examples on various benchmark problems demonstrated that the HP classifier can perform highly accurate classification surpassing or, at least, on par with the state-of-the-art approaches. Moreover, the very high computational efficiency of the HP classifier on large-scale problems is also justified.

There are several considerations for future work. Firstly, the optimality of the HP classifier needs to be investigated. This is of paramount importance to the proposed approach because it determines the validity and effectiveness of the learning results. There are a few potential modifications that can enhance the objectiveness and effectiveness of the HP classifier and strengthen its adaptive ability and applicability for real-world problems. For example, the current HP classifier requires a full re-training if extra layers are required to be added into the system structure. An approach to dynamically add new layers based on requirements during the online learning process will be valuable. It will be a strong novelty if some operating mechanisms are introduced to the HP classifier that enables the approach to autonomously determine the optimal layer number for each hierarchy based on the ensemble properties of data. Alternatively, developing some criteria that can help users to determine the most desirable layer number for a given problem without *prior* knowledge will also be useful. Another important direction will be introducing a data-driven approach for determining parameter setting (radii) to replace the hard-coding method used by the current version.

References

[1] 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.

[2] 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.

[3] M. Antonelli, D. Bernardo, H. Hagras, and F. Marcelloni, "Multiobjective evolutionary optimization of type-2 fuzzy rule-based systems for financial data classification," IEEE Trans. Fuzzy Syst., vol. 25, no. 2, pp. 249–264, 2017.

[4] X. Bian, C. Chen, L. Tian, and Q. Du, "Fusing local and global features for high-resolution scene classification," IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 10, no. 6, pp. 2889–2901, 2017.

[5] D. M. Blei and M. I. Jordan, "Variational inference for Dirichlet process mixtures," Bayesian Anal., vol. 1, no. 1 A, pp. 121–144, 2006.

[6] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," Commun. Stat. Methods, vol. 3, no. 1, pp. 1–27, 1974.

[7] S. L. Chiu, “Fuzzy model identification based on cluster estimation,” J. Intell. Fuzzy Syst., vol. 2, no. 3, pp. 267–278, 1994.
[8] 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.
[9] N. Cristianini and J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods. Cambridge: Cambridge University Press, 2000.
[10] P. Cunningham and S. J. Delany, “K-nearest neighbour classifiers,” Mult. Classif. Syst., vol. 34, pp. 1–17, 2007.
[11] D. L. Davies and D. W. Bouldin, “A cluster separation measure,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 2, pp. 224–227, 1979.
[12] O. Erkaymaz, M. Ozer, and M. Perc, “Performance of small-world feedforward neural networks for the diagnosis of diabetes,” Appl. Math. Comput., vol. 311, pp. 22–28, 2017.
[13] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in International Conference on Knowledge Discovery and Data Mining, 1996, vol. 96, pp. 226–231.
[14] B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” Science (80-.)., vol. 315, no. 5814, pp. 972–976, 2007.
[15] 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.
[16] 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.
[17] X. Gu and P. P. Angelov, “Self-organising fuzzy logic classifier,” Inf. Sci. (Ny)., vol. 447, pp. 36–51, 2018.
[18] X. Gu, P. P. Angelov, and J. C. Principe, “A method for autonomous data partitioning,” Inf. Sci. (Ny)., vol. 460–461, pp. 65–82, 2018.
[19] X. Gu, P. P. Angelov, C. Zhang, and P. M. Atkinson, “A massively parallel deep rule-based ensemble classifier for remote sensing scenes,” IEEE Geosci. Remote Sens. Lett., vol. 15, no. 3, pp. 345–349, 2018.
[20] H. Hagras, “Toward human-understandable, explainable AI,” Computer (Long. Beach. Calif)., vol. 51, no. 9, pp. 28–36, 2018.
[21] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” Neural Networks, vol. 2, no. 5, pp. 359–366, 1989.
[22] L. Huang, C. Chen, W. Li, and Q. Du, “Remote sensing image scene classification using multi-scale completed local binary patterns and fisher vectors,” Remote Sens., vol. 8, no. 6, pp. 1–17, 2016.
[23] D. Kangin, P. Angelov, and J. A. Iglesias, “Autonomously evolving classifier TEDAClass,” Inf. Sci. (Ny)., vol. 366, pp. 1–11, 2016.
[24] 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.
[25] T. Kohonen, Self-organizing maps. Berlin: Springer, 1997.
[26] 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.
[27] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nat. Methods, vol. 13, no. 1, pp. 35–35, 2015.
[28] E. Lughofer, “Evolving fuzzy systems—fundamentals, reliability, interpretability, useability, applications,” in Handbook on Computational Intelligence, P. Angelov, Ed. New York: World Scientific, 2016, pp. 67–135.
[29] W. Luo, J. Li, J. Yang, W. Xu, and J. Zhang, “Convolutional sparse autoencoders for image classification,” IEEE Trans. Neural Networks Learn. Syst., vol. 29, no. 7, pp. 3289–3294, 2018.
[30] M. N. Murty and V. S. Devi, Introduction to pattern recognition and machine learning. World Scientific., 2015.
[31] A. Nguyen, J. Yosinski, and J. Clune, “Deep neural networks are easily fooled: high confidence predictions for unrecognizable images,” in IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 427–436.
[32] 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.
[33] A. B. Penatti, K. Nogueira, and J. A. Santos, “Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?” in IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 44–51.

[34] 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.
[35] 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.
[36] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” J. Comput. Appl. Math., vol. 20, pp. 53–65, 1987.
[37] J. Sanchez et al., “Image classification with the Fisher vector: theory and practice,” Int. J. Comput. Vis., vol. 105, no. 3, pp. 222–245, 2013.
[38] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in International Conference on Learning Representations, 2015, pp. 1–14.
[39] S. S. Sivatha Sindhu, S. Geetha, and A. Kannan, “Decision tree based light weight intrusion detection using a wrapper approach,” Expert Syst. Appl., vol. 39, pp. 129–141, 2012.
[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. 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.
[42] 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.
[43] S. Yang, W. Wang, C. Liu, and W. Deng, “Scene understanding in deep learning-based end-to-end controllers for autonomous vehicles,” IEEE Trans. Syst. Man, Cybern. Syst., vol. 49, no. 1, pp. 53–63, 2019.
[44] 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.
[45] 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.
[46] 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.
[47] C. Zhang, J. Cheng, C. Li, and Q. Tian, “Image-Specific Classification with Local and Global Discriminations,” IEEE Trans. Neural Networks Learn. Syst., vol. 29, no. 9, pp. 4479–4486, 2018.
[48] 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.
[49] L. Zhang, L. Zhang, and V. Kumar, “Deep learning for remote sensing data,” IEEE Geosci. Remote Sens. Mag., vol. 4, no. 2, pp. 22–40, 2016.
[50] C. Zhang, G. Zhu, C. Liang, Y. Zhang, Q. Huang, and Q. Tian, “Image class prediction by joint object, context, and background modeling,” IEEE Trans. Circuits Syst. Video Technol., vol. 28, no. 2, pp. 428–438, 2018.

*Declaration of Interest Statement

Declaration of Interests

This manuscript is the authors' original work. The authors declare no competing interests.

Xiaowei Gu and Weiping Ding