Self-Organizing Divisive Hierarchical Voronoi Tessellation-Based Classifier

Xiaowei Gu[1] and Qiang Shen[2]

[1]School of Computing, University of Kent, Canterbury CT2 7NZ, UK
[2]Department of Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, UK
E-mails: X.Gu@kent.ac.uk; qqs@aber.ac.uk

Abstract: In this paper, a novel approach to the self-organization of hierarchical prototype-based classifiers from data is proposed. The approach recursively partitions the data at multiple levels of granularity into shape-free clusters of different sizes, resembling Voronoi tessellation, and naturally aggregates the resulting cluster medoids into a multi-layered prototype-based structure according to their descriptive abilities. Different from conventional classification models, it is nonparametric and entirely data-driven, and the learned model can offer a high-level of transparency and interpretability thanks to the underlying prototype-based nature. The system identification process underpinning the approach is driven by the aim of separating data samples of different classes into nonoverlapping multi-granular clusters. Its associated decision-making process follows the "nearest prototype" principle and hence, the rationales of the subsequent decisions made can be explicitly explained. Experimental studies based on popular benchmark classification problems, as well as on a practical application to remote sensing image classification, demonstrate the efficacy of the proposed approach.

Keywords: classification; divisive partitioning; prototype; self-organizing; hierarchical model.

1. Introduction

Classification is a hot research topic in machine learning and statistics. As a typical form of supervised learning, classification methods aim to construct predictive models from labelled training data capable of predicting the class labels of new observations. To date, classification methods have been developed and implemented for real-world applications in various areas such as remote sensing [1] and biomedical analysis [2], amongst many others.

In recent years, issues of understandability and explainability have gained increasing attention from both research communities and the general public [3]. This is largely due to the wide deployment of complicated machine learning models in life-critical applications, e.g., autonomous driving [4], structural health monitoring [5]. Currently, deep neural networks (DNNs) is one of the most popular classification methods offering the state-of-the-art performances in terms of accuracy on many complex practical problems concerning visual and audio information processing [6]. However, DNNs are highly sophisticated models with a huge number (millions) of hyperparameters with no clear physical meanings. Hence, they are often being criticized as a typical type of "black box" models lack of transparency [7]. Although DNNs can be simplified by pruning less important parameters [8], a large proportion of hyperparameters still need to be kept in order to maintain a high level of performance, thereby the complexity of the models remaining high. Many mainstream classification methods, such as random forests (RFs) [9], support vector machines (SVMs) [10], learning vector quantization (LVQ) [11] are also characterized as being opaque. Despite the great performances they have demonstrated, the lack of transparency and therefore, that of trustability of these predictive machine learning models is not a trivial issue and may cause severe consequences [7]. Having recognized this, researchers and industry practitioners are now frequently calling for explainable artificial intelligence (XAI) [3].

Decision trees (DTs) [12], k-nearest neighbours (KNN) [13] and evolving fuzzy systems (EFSs) [14] are generally regarded as interpretable machine learning models. In particular, DT aims to build a tree-like predictive model by recursively splitting data. However, the explainability of DT is usually limited if the dimensionality of the problem is high. Instead of learning a predictive model from data, KNN uses all the training samples directly to classify unlabelled data by following the "nearest neighbours" principle [15]. The operating mechanism of KNN is simple to understand and can be very effective to small-scale problems, but it also has several weaknesses when applied to large-scale problems, such as lower interpretability, and low computational- and memory-efficiency. EFSs are a powerful tool for data stream processing and have been widely applied to addressing a range of real-world problems [16]–[18]. Study of EFSs has increasingly become a major scientific endeavour over the past two decades, and a number of more advanced EFSs have been introduced in the past few years. These include self-organising fuzzy inference system (SOFIS) [16], recursive maximum correntropy-based evolving fuzzy system

[17], and jointly evolving and compressing fuzzy system [18], etc. Based on IF-THEN fuzzy production rules, EFSs are capable of self-adapting both model structure and meta-parameters online and doing so simultaneously from streaming data, to capture the dynamical changes of data patterns. Compared with their first-order counterparts [17], [18], zero-order EFSs [16], [19] are particularly designed for data stream classification. Being a prototype-based method, a zero-order EFS extracts a subset of highly representative prototypes from training data to facilitate classification. Such a prototype-based nature brings higher transparency and explainablity to the resulting EFSs. Nevertheless, it is often observed that the knowledge base of a zero-order EFS is unfavourably large. Whilst a large-sized knowledge base may be necessary for many applications to achieve high-level performance on complex problems, it can impair the interpretability of the predictive model as well as the explainability of the reasoning process that runs on such a complicated model, in addition to increased computational costs.

Although a universally perfect classifier for any given application is unobtainable, designing more effective classifiers with higher computational efficiency and model transparency is highly rewarding. More recently, a number of new classification methods have been proposed. For example, a sequence classifier (SC) is introduced in [20], which works by sorting and ranking data attributes to create a dictionary for classification. An eigenvalue-based classification method called EigenClass is presented in [21], which determines the class label of a testing sample based on its eigenvalues calculated with respect to the available training samples. A selective prototype-based learning (SPL) classifier is provided in [22] for streaming data classification, which learns from data streams, sample by sample, while simultaneously maintaining two sets of prototypes, namely, important instance set (ISet) and potential concept-drifting instance set (PSet). Interestingly, ISet contains the most important samples learned by considering error-driven representativeness, in an effort to capture the current concept for classification, and PSet stores the misclassified samples for detecting the abrupt concept drifts. A graph-based prototype selector ensemble model is proposed in [23], which exploits an undirected graph to store the prototypes selected at each iteration and also, the relationships between them to enable classification. In [24], a comprehensive study is conducted to investigate the influence of employing different base learners, and that of running different methods that combine such bases to form classifier ensembles, on the performances of the resulting ensembles. To boost the efficacy of classifier ensembles, in the literature, work has also been carried out to reduce the complexity while retaining model transparency through innovative application of attribute selection techniques [25].

In general, the use of prototypes helps preserve the structure and underlying patterns of the original data. Models constructed with a prototype-based method typically demonstrate a higher level of model transparency and explainability than alternative classifiers (e.g., DNN/ANN, RF, SVM). However, a common problem that such models suffer from is system obesity, often occurring when prototype-based methods are applied to performing large-scale complex classification tasks. A large-sized knowledge base (with too many prototypes) can impair the interpretability and explainability of the learned model greatly. A feasible solution to resolve this bottleneck problem is to arrange the identified prototypes in multiple layers according to their descriptive abilities [26]–[28], such that users can use the more descriptive prototypes representing the global patterns of data (that are placed at higher layers) to interpret and understand the general picture of the problem under consideration, whilst utilizing the less descriptive prototypes depicting local data structures (as placed at lower layers) to obtain auxiliary finer details. An example of this is the two-level prototype-based classifier as presented in [26], named SyncStream. It learns a two-level prototype-based structure from data with the first level containing raw prototypes for capturing the current concept and the second containing highly representative prototypes representing historical concepts. However, the main issue with SyncStream is that its model size and predictive performance are subject to a number of externally controlled parameters, including the maximum numbers of prototypes to be stored at the first and second levels, the decay rate, and the noise threshold. Without sufficient prior knowledge to predetermine these parameters appropriately, it can be very difficult for SyncStream to achieve satisfactory performance.

In contrast with systems like SyncStream, hierarchical prototype-based (HP) classifiers [27] and multi-granularity locally optimal prototype-based (MLOP) classifiers [28] offer more advanced multi-layer prototype-based approaches for classification. With a pre-determined model depth, a HP classifier self-organizes a number of pyramidical hierarchies from streaming data based on the prototypes identified at multiple levels of granularity. The constructed hierarchies are capable of continuously self-evolving by adding new prototypes to capture any new data patterns. A MLOP classifier takes one step ahead further, by employing the classical elbow method [29] to self-determine the model depth through adjusting the intra-cluster variance that is controlled by a regularization parameter. MLOP further involves an iterative optimization process to attain the local optimality of prototypes.

However, both HP and MLOP fail to capture the inter-class relationship of data because the prototypes are learned from data of different classes separately, ignoring potential class overlaps or interactions. Besides, they both still require externally controlled parameters (to be predefined by users), namely, the model depth for HP and the regularization parameter for MLOP. Nonetheless, these parameters play an important role during the learning process, in assisting in the optimization of the size and the predictive precision of the models learned from data.

In trying to attain the strengths of HP and MLOP while addressing their shortcomings, a novel approach is herein presented for self-organizing a Divisive Hierarchical Voronoi Tessellation-based (DHT) model to perform classification tasks. Particularly, mirroring the top-down data splitting mechanism used by hierarchical divisive clustering algorithm [30], DHT self-learns a hierarchical prototype-based structure, via recursively partitioning data and creating shape-free clusters resembling Voronoi tessellations at multiple levels of granularity from low to high, thereby separating data samples of different classes. The medoids of clusters obtained during the recursively partitioning process are selected as prototypes and are subsequently arranged in a single multi-layer pyramidical hierarchy. In so doing, the potential class overlaps are taken into consideration during the model identification process. Compared with its predecessors (SyncStream [26], HP [27] and MLOP [28]), DHT has the following unique advantages:

1) it is nonparametric, no externally controlled parameter is required to be predefined;

2) its system identification process is driven by data and thus, is highly objective; and

3) it enables a better understanding of multi-model distributions of data, by considering the inter-class overlaps.

An additional key feature of DHT is that its prototypes are themselves highly representative real samples in the data space rather than the commonly used cluster means, which usually do not physically exist and hence have no real meaning. As such, the DHT models can provide more institutive information about the problem and are guaranteed to be semantically interpretable in a given applied field. Experimental investigations conducted on a variety of benchmark problems, including a real-world application to remote sensing image classification, also demonstrate the efficacy of the proposed DHT approach.

The remainder of this paper is organized as follows. Technical details of the proposed approach are presented in Section 2. Its computational complexity is analysed in Section 3. Section 4 provides experimental case studies, and the paper is concluded in Section 5.

2. Proposed Approach

In this section, the general architecture and the learning and decision-making strategies of the proposed DHT approach are described in detail.

2.1. Key Notations

Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_K\}$ be a particular static dataset in the N dimensional real data space $\boldsymbol{\mathcal{R}}^N$ with the corresponding class labels $\boldsymbol{Y} = \{y_1, y_2, \dots, y_K\}$, where K is the cardinality of $\boldsymbol{X}$; $\boldsymbol{x}_k = \left[x_{k,1}, x_{k,2}, \dots, x_{k,N}\right]^T \in \boldsymbol{\mathcal{R}}^N$ denotes the kth data sample of $\boldsymbol{X}$; and y_k is the corresponding class label of $\boldsymbol{x}_k$. Also, without losing generality, assume that $\boldsymbol{X}$ is composed of data samples of C different classes, that is, $y_k \in \{1,2, \dots, C\}$ $(k = 1,2, \dots, K)$. For presentational simplicity, a list of key notations used is summarized in Table 1.

Table 1. List of key notations and their respective definitions

Notation	Definition
$\boldsymbol{X}$	Static dataset
$\boldsymbol{Y}$	Class labels of $\boldsymbol{X}$
$\boldsymbol{\mathcal{R}}^N$	Data space
K	Cardinality of $\boldsymbol{X}$
N	Dimensionality of $\boldsymbol{\mathcal{R}}^N$
$\boldsymbol{x}_k$	The kth data sample of $\mathbf{X}$
y_k	Class label of $\boldsymbol{x}_k$
C	Number of classes
g	Level of granularity
$\boldsymbol{D}$	Pairwise distance matrix
$\boldsymbol{\Sigma}$	Diagonal covariance matrix

σ_g	Radius of zone of influence around each prototype at the gth level of granularity
$\boldsymbol{U}_g$	Membership matrix of $\boldsymbol{X}$ obtained at the gth level of granularity
λ_g	Cumulative membership calculated from $\boldsymbol{U}_g$
$\boldsymbol{P}_g$	Set of prototypes identified at the gth level of granularity
$\mathbb{C}_g$	Set of clusters formed around $\boldsymbol{P}_g$
P_g	Cardinality of $\boldsymbol{P}_g$
$\boldsymbol{p}_{g,k}$	The kth prototype of $\boldsymbol{P}_g$
$\boldsymbol{C}_{g,k}$	Cluster formed around $\boldsymbol{p}_{g,k}$
$S_{g,k}$	Cardinality of $\boldsymbol{C}_{g,k}$
$\rho_{g,k}$	Purity of $\boldsymbol{C}_{g,k}$
$\boldsymbol{P}_g^L$	Set of leaf prototypes within $\boldsymbol{P}_g$
$\boldsymbol{P}_g^I$	Set of internal prototypes within $\boldsymbol{P}_g$
P_g^I	Cardinality of $\boldsymbol{P}_g^I$
$\boldsymbol{p}_{g,k}^I$	The kth prototype of $\boldsymbol{P}_g^I$
$\boldsymbol{C}_{g,k}^I$	Cluster formed around $\boldsymbol{p}_{g,k}^I$
$\boldsymbol{U}_{g+1,k}^I$	Local membership matrix of $\boldsymbol{C}_{g,k}^I$ obtained at the (g+1)th level of granularity
$S_{g,k}^I$	Cardinality of $\boldsymbol{C}_{g,k}^I$
$\lambda_{g+1,k}^I$	Local cumulative membership calculated from $\boldsymbol{U}_{g+1,k}^I$
$\boldsymbol{P}_{g+1,k}$	Set of prototypes identified from $\boldsymbol{C}_{g,k}^I$ at the (g+1)th level of granularity
$\boldsymbol{P}^L$	Set of leaf prototypes identified from $\boldsymbol{X}$
P^L	Cardinality of $\boldsymbol{P}^L$

2.2. General Architecture

The main aim of DHT is to self-learn a multi-layered prototype-based hierarchical structure from data with each node being a prototype, as depicted in Fig. 1. These prototypes are the medoids of clusters formed by neighbouring samples at different levels of granularity, ordered from low to high, resembling Voronoi tessellations [31].

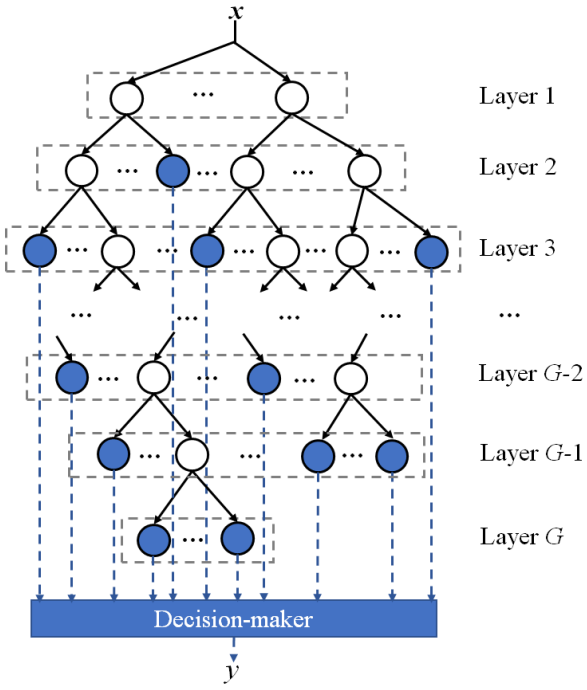


Fig. 1. General architecture of DHT

The hierarchical structure of DHT is purely determined by the ensemble properties and mutual distances of data samples observed in the data space. Each layer corresponds to a particular level of granularity. Prototypes at higher layers are identified from data at lower levels of granularity, and they represent the global patterns of data better and have a greater descriptive power. Prototypes at lower layers are obtained at higher levels of granularity, containing fine details about the data distribution and being able to better describe the local data patterns. These

prototypes can be further divided into two groups: *i)* internal prototypes, and *ii)* leaf prototypes. A prototype is recognized as a leaf only if the cluster formed around it consists of data samples of the same class. Otherwise, it is recognized as an internal prototype. An internal prototype can have multiple child prototypes directly linked to it, as the medoids of smaller clusters obtained by partitioning the cluster formed around it at a higher level of granularity aims to separate the data of different classes. These child prototypes can also have their own children if they are internal prototypes, being surrounded by data samples of mixed classes. Clusters formed around internal prototypes (white nodes in Fig. 1) are composed of data samples of multiple classes, whilst clusters formed around leaf prototypes (blue nodes in Fig. 1) are each composed of data samples of the same class.

During the system identification process, DHT firstly initializes its structure by identifying the apex prototypes from the given data at the lowest level of granularity. Then, it continuously self-develops its hierarchical structure by partitioning the data at higher levels of granularity and automatically increases its depth at the same time until all the prototypes at the bottom layer become leaves. Through this process, DHT achieves a multi-granular partition of data such that the data samples of different classes are separated by clusters formed around leaf prototypes. Of course, the identified leaf prototypes are connected directly to the decision-maker. For decision-making, the class labels of unlabelled data samples are determined on the basis of their distances to the leaf prototypes. In the following subsections, the learning and decision-making policies are further detailed.

2.3. Learning Policy

Step A. System Initialization - Apex Prototype Identification

The system identification process starts by identifying apex prototypes. These apex prototypes locate at the top layer of the hierarchical structure, and they are obtained from all the observed data samples at the first level of granularity, namely, $g = 1$. Therefore, apex prototypes are the most descriptive samples representing the global patterns of data distribution. In order to identify such samples, distances between data samples of $\boldsymbol{X}$ are firstly calculated and a $K \times K$ dimensional pairwise distance matrix is obtained as Eqn. (1):

$$\boldsymbol{D} = \left[d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)\right]_{j=1:K}^{i=1:K} \tag{1}$$

where $d^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \boldsymbol{x}_j)$; $\boldsymbol{\Sigma}$ is an $N \times N$ dimensional diagonal matrix with its main diagonal elements being the variances of $\boldsymbol{X}$. In this work, Mahalanobis distance metric is employed as the default distance measure to ensure that different attributes of the data contribute to the identification process equally. However, if preferred, one may adopt any of other commonly used distance metrics or pseudo metrics (e.g., Euclidean distance and cosine dissimilarity) as an alternative for DHT, depending on the nature of the problem.

Based on the distance matrix $\boldsymbol{D}$, the radius of zone of influence around each prototype, denoted as σ_g at the first level of granularity ($g = 1$) can be estimated using Eqn. (2), as the average distance between any two data samples:

$$\sigma_g^2 = \frac{1}{K(K-1)} \sum_{i=1}^{K} \sum_{j=1}^{K} d^2(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{2}$$

Then, every individual data sample, $\boldsymbol{x}_k$ is treated as a micro-cluster with the sample itself being the cluster medoid. From this, there are a total of K micro-clusters in the data space $\boldsymbol{\mathcal{R}}^N$. By letting the K micro-cluster medoids be assigned a membership with respect to each of the others (including itself), a $K \times K$ dimensional membership matrix is obtained as follows [32]:

$$\boldsymbol{U}_g = \left[\bar{\mu}(\boldsymbol{x}_i, \boldsymbol{x}_j, \sigma_g)\right]_{i=1:K}^{j=1:K} \tag{3}$$

where $\bar{\mu}(\boldsymbol{x}_i, \boldsymbol{x}_j, \sigma_g) = \frac{\mu(\boldsymbol{x}_i, \boldsymbol{x}_j, \sigma_g)}{\sum_{l=1}^{K} \mu(\boldsymbol{x}_l, \boldsymbol{x}_j, \sigma_g)}$ is the normalized membership of which the *i*th micro-cluster medoid, $\boldsymbol{x}_i$ is assigned to the *j*th micro-cluster medoid, with $\boldsymbol{x}_j$ at the *g*th level of granularity; and $\mu(\boldsymbol{x}_i, \boldsymbol{x}_j, \sigma_g)$ is calculated by Eqn. (4):

$$\mu(\boldsymbol{x}_i, \boldsymbol{x}_j, \sigma_g) = e^{-\frac{d^2(x_i, x_j)}{\sigma_g^2}} \tag{4}$$

Cumulative memberships of these micro-cluster medoids are calculated from $\boldsymbol{U}_g$ as Eqn. (5) [32]:

$$\lambda_g(\boldsymbol{x}_i) = \sum_{j=1}^{K} \bar{\mu}(\boldsymbol{x}_i, \boldsymbol{x}_j, \sigma_g) = \sum_{j=1}^{K} \frac{\mu(\boldsymbol{x}_i, \boldsymbol{x}_j, \sigma_g)}{\sum_{l=1}^{K} \mu(\boldsymbol{x}_l, \boldsymbol{x}_j, \sigma_g)} \tag{5}$$

where $\lambda_g(\boldsymbol{x}_i)$ is the cumulative membership of $\boldsymbol{x}_i$. Based on $\lambda_g(\boldsymbol{x}_i)$ ($i = 1,2,\dots,K$), apex prototypes at the first layer of the hierarchical structure are identified using **Condition 1** (Eqn. (6)) [32]:

$$\textbf{\textit{Condition 1}}: \quad if \left(\lambda_g(\boldsymbol{x}_i) = \max_{\substack{d^2(\boldsymbol{x}_i, \boldsymbol{x}) \le \sigma_g^2; \\ \boldsymbol{x} \in \boldsymbol{X};}} \left(\lambda_g(\boldsymbol{x}) \right) \right) \quad then \left(\boldsymbol{P}_g \leftarrow \boldsymbol{P}_g \cup \{\boldsymbol{x}_i\} \right) \tag{6}$$

Condition 1 selects data samples of a locally maximum cumulative membership value as apex prototypes with their collection denoted as $\boldsymbol{P}_g$.

Remark 1: In exceptional cases, there may exist two (or more) neighbouring data samples satisfying **Condition 1** at the same time, namely, $\lambda_g(\boldsymbol{x}_j) = \lambda_g(\boldsymbol{x}_i) = \max_{\substack{d^2(\boldsymbol{x}_i, \boldsymbol{x}) \le \sigma_g^2; \\ \boldsymbol{x} \in \boldsymbol{X};}} \left(\lambda_g(\boldsymbol{x}) \right)$ and $j \neq i$. In the event that this happens, those data samples satisfying **Condition 1** will be selected as apex prototypes unless they are positioned at an exactly identical location in the data space, which means that $\boldsymbol{x}_j = \boldsymbol{x}_i$ and $j \neq i$. The same principle also applies to **Condition 2**, which will be given in Eqn. (12) later.

Suppose that there are a total of P_g apex prototypes being identified, namely, $\boldsymbol{P}_g = \left\{ \boldsymbol{p}_{g,1}, \boldsymbol{p}_{g,2}, \dots, \boldsymbol{p}_{g,P_g} \right\}$. Then, clusters can be created using these apex prototypes to attract nearby data samples to form Voronoi tessellations [31]:

$$\boldsymbol{C}_{g,j^*} \leftarrow \boldsymbol{C}_{g,j^*} \cup \{\boldsymbol{x}_i\}; \quad j^* = \underset{j=1,2,\dots,P_g}{\operatorname{argmin}} \left(d^2(\boldsymbol{x}_i, \boldsymbol{p}_{g,j}) \right) \tag{7}$$

where $i = 1,2,\dots,K$; and $\boldsymbol{C}_{g,j}$ is the cluster constructed around $\boldsymbol{p}_{g,j}$. The collection of clusters at the first layer of this hierarchical structure is denoted as $\mathbb{C}_g$ ($\mathbb{C}_g = \left\{ \boldsymbol{C}_{g,1}, \boldsymbol{C}_{g,2}, \dots, \boldsymbol{C}_{g,P_g} \right\}$).

From the above, the purity of each cluster, $\boldsymbol{C}_{g,i}$ is calculated using Eqn. (8) ($i = 1,2,\dots,P_g$):

$$\rho_{g,i} = \frac{S_{g,i}^d}{S_{g,i}} \tag{8}$$

where $\rho_{g,i}$ is the purity of $\boldsymbol{C}_{g,i}$; $S_{g,i}$ is the cardinality (number of data samples) of $\boldsymbol{C}_{g,i}$; and $S_{g,i}^d$ is the number of data samples with the dominate class label in $\boldsymbol{C}_{g,i}$. If $\rho_{g,i} = 1$, it suggests that $\boldsymbol{C}_{g,i}$ is a pure cluster formed by data samples of the same class. In this case, the cluster medoid, $\boldsymbol{p}_{g,i}$ is recognized as a leaf prototype with the corresponding class label denoted as $y_{g,i}$. Otherwise, namely, $\rho_{g,i} < 1$, $\boldsymbol{p}_{g,i}$ is an internal prototype, and $\boldsymbol{C}_{g,i}$ is an impure cluster and needs to be partitioned at a higher level of granularity in order to separate data samples of different classes. Based on Eqn. (8), $\boldsymbol{P}_g$ can be divided into two non-overlapping sets: one is the set of leaf prototypes, $\boldsymbol{P}_g^L$, and the other is the set of internal prototypes, $\boldsymbol{P}_g^I$, which satisfy that $\boldsymbol{P}_g^L \cup \boldsymbol{P}_g^I = \boldsymbol{P}_g$ and $\boldsymbol{P}_g^L \cap \boldsymbol{P}_g^I = \emptyset$. Accordingly, the constructed clusters associated with $\boldsymbol{P}_g$ can be divided into two groups: the impure clusters formed around internal prototypes, $\boldsymbol{P}_g^I$ and the pure clusters formed around leaf prototypes, $\boldsymbol{P}_g^L$. Although internal prototypes are associated with data samples of different classes and will not participate in decision-making directly, they represent the peaks of multi-model distribution of data and help disclose key information about class overlaps and interactions.

If there are any clusters formed around the apex prototypes with purity values smaller than 1, it means that data samples of different classes cannot be satisfactorily separated at the current level of granularity and thus, the system identification process enters ***Step B*** for finer partitioning.

Step B. Pyramidical Hierarchy Growth - Child Prototype Identification

Without losing generality, suppose that there are P_g^I internal prototypes identified at the gth level of granularity, namely, $\boldsymbol{p}_g^I = \left\{\boldsymbol{p}_{g,1}^I, \boldsymbol{p}_{g,2}^I, \ldots, \boldsymbol{p}_{g,P_g^I}^I\right\} \subseteq \boldsymbol{P}_g$, the clusters formed around them, denoted as $\boldsymbol{C}_{g,1}^I, \boldsymbol{C}_{g,2}^I, \ldots, \boldsymbol{C}_{g,P_g^I}^I$ will then be partitioned at a higher level of granularity (namely, $g+1$) with the aim of separating data samples of different classes.

The radius of the zone of influence around each prototype at the (g+1)th level of granularity is first estimated using Eqn. (9):

$$\sigma_{g+1}^2 = \frac{1}{\sum_{i=1}^K \sum_{j=1}^K w_{g,i,j}} \sum_{i=1}^K \sum_{j=1}^K w_{g,i,j} d^2(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{9}$$

where $w_{g,i,j} = \begin{cases} 1, & if\ d^2(\boldsymbol{x}_i, \boldsymbol{x}_j) \le \sigma_g^2 \\ 0, & if\ d^2(\boldsymbol{x}_i, \boldsymbol{x}_j) > \sigma_g^2 \end{cases}$.

Next, for each impure cluster, $\boldsymbol{C}_{g,i}^I$ ($i = 1,2,\ldots,P_g^I$), a similar prototype identification process as used in ***Step A*** is applied to identify child prototypes from its members. Data sample associated with $\boldsymbol{C}_{g,i}^I$ are treated as micro-cluster medoids and a $S_{g,i}^I \times S_{g,i}^I$ dimensional local membership matrix is obtained by letting them assign memberships to each other, similar to Eqn. (3):

$$\boldsymbol{U}_{g+1,i}^I = \left[\bar{\mu}(\boldsymbol{z}_j, \boldsymbol{z}_l, \sigma_{g+1})\right]_{l=1:S_{g,i}^I}^{j=1:S_{g,i}^I} \tag{10}$$

where $\boldsymbol{z}_j, \boldsymbol{z}_l \in \boldsymbol{C}_{g,i}^I$; $\bar{\mu}(\boldsymbol{z}_j, \boldsymbol{z}_l, \sigma_{g+1}) = \frac{\mu(\boldsymbol{z}_j, \boldsymbol{z}_l, \sigma_{g+1})}{\sum_{k=1}^{S_{g,i}^I} \mu(\boldsymbol{z}_k, \boldsymbol{z}_l, \sigma_{g+1})}$; $\mu(\boldsymbol{z}_j, \boldsymbol{z}_l, \sigma_{g+1})$ is calculated by Eqn. (4).

Local cumulative memberships are computed from $\boldsymbol{U}_{g+1,i}^I$ by Eqn. (11):

$$\lambda_{g+1,i}^I(\boldsymbol{z}_j) = \sum_{l=1}^{S_{g,i}^I} \mu(\boldsymbol{z}_j, \boldsymbol{z}_l, \sigma_{g+1}) = \sum_{l=1}^{S_{g,i}^I} \frac{\mu(\boldsymbol{z}_j, \boldsymbol{z}_l, \sigma_{g+1})}{\sum_{k=1}^{S_{g,i}^I} \mu(\boldsymbol{z}_k, \boldsymbol{z}_l, \sigma_{g+1})} \tag{11}$$

where $\lambda_{g+1,i}^I(\boldsymbol{z}_j, \sigma_{g+1})$ is the cumulative membership of $\boldsymbol{z}_j$ calculated locally within $\boldsymbol{C}_{g,i}^I$. **Condition 2** is used for identifying the child prototypes of $\boldsymbol{p}_{g,i}^I$ from members of $\boldsymbol{C}_{g,i}^I$:

$$\textbf{\textit{Condition 2}}: \quad if \left(\lambda_{g+1,i}^I(\boldsymbol{z}_j) = \max_{\substack{d^2(\boldsymbol{z}_j, \boldsymbol{x}) \le \sigma_{g+1}^2; \\ \boldsymbol{z} \in \boldsymbol{C}_{g,i}^I;}} \left(\lambda_{g+1,i}^I(\boldsymbol{z})\right) \right) \quad then\ \left(\boldsymbol{P}_{g+1,i} \leftarrow \boldsymbol{P}_{g+1,i} \cup \{\boldsymbol{z}_j\}\right) \tag{12}$$

Condition 2 selects data samples from $\boldsymbol{C}_{g,i}^I$ of the highest cumulative membership locally to join $\boldsymbol{P}_{g+1,i}$, which is the collection of child prototypes of $\boldsymbol{p}_{g,i}^I$. Given that there are $P_{g+1,i}$ prototypes identified from $\boldsymbol{C}_{g,i}^I$ at the (g+1)th level of granularity, namely, $\boldsymbol{P}_{g+1,i} = \left\{\boldsymbol{p}_{g+1,i,1}, \boldsymbol{p}_{g+1,i,2}, \ldots, \boldsymbol{p}_{g+1,i,P_{g+1,i}}\right\}$, $\boldsymbol{C}_{g,i}^I$ is partitioned into $P_{g+1,i}$ smaller clusters by forming Voronoi tessellations locally:

$$\boldsymbol{C}_{g+1,i,k^*}^I \leftarrow \boldsymbol{C}_{g+1,i,k^*}^I \cup \{\boldsymbol{z}_j\}; \quad k^* = \underset{k=1,2,\ldots,P_{g+1,i}}{\operatorname{argmin}} \left(d^2(\boldsymbol{z}_j, \boldsymbol{p}_{g+1,i,k})\right) \tag{13}$$

where $\boldsymbol{z}_j \in \boldsymbol{C}_{g,i}^I$. The collection of smaller clusters obtained from $\boldsymbol{C}_{g,i}^I$ is denoted as $\mathbb{C}_{g+1,i}$, namely, $\mathbb{C}_{g+1,i} = \left\{\boldsymbol{C}_{g+1,1}, \boldsymbol{C}_{g+1,2}, \ldots, \boldsymbol{C}_{g+1,P_{g+1,i}}\right\}$.

After all the impure clusters have been partitioned at the (g+1)th level of granularity, a new layer is added to the hierarchical structure of DHT based on these newly identified child prototypes. The collection of prototypes at the (g+1)th layer is denoted as $\boldsymbol{P}_{g+1}$, namely, $\boldsymbol{P}_{g+1} = \boldsymbol{P}_{g+1,1} \cup \boldsymbol{P}_{g+1,2} \cup \ldots \cup \boldsymbol{P}_{g+1,P_g^I}$. Correspondingly, the

collection of the clusters formed around them is denoted as $\mathbb{C}_{g+1}$, namely, $\mathbb{C}_{g+1} = \mathbb{C}_{g+1,1} \cup \mathbb{C}_{g+1,2} \cup \ldots \cup \mathbb{C}_{g+1,P_g^I}$. The cardinality of $\boldsymbol{P}_{g+1}$ is denoted as P_{g+1}, where $P_{g+1} = \sum_{i=1}^{P_g^I} P_{g+1,i}$.

Then, purity values of the newly obtained clusters are calculated using Eqn. (8). If all the clusters at the (g+1)th layer have the purity value 1, it means that the data samples of different classes have been sufficiently separated at the current level of granularity, and that the system identification process terminates. Otherwise, ***Step B*** is repeated to partition the newly obtained impure clusters at a higher level of granularity (while setting $g \leftarrow g + 1$).

Remark 2: The system identification process of DHT will in general, self-terminate automatically thanks to the recursive partitioning mechanism utilized. At each partitioning cycle (say, the gth cycle), only those impure clusters at the previous cycle (namely, the $(g-1)$th) are partitioned, and the resulting prototypes will form smaller clusters with a smaller radius of the zone of influence (as specified by Eqn. (9)), associated with less data samples. The impure clusters derived from the current cycle will then be partitioned at a higher level of granularity in the next cycle. Hence, this recursive partitioning process will generally separate data samples of different classes and terminate itself at the end when the purity values of all the newly obtained clusters reach 1. Such a mechanism is also computationally efficient because there are always less data samples to be partitioned cycle by cycle. In theory, however, this system identification process may enter into an infinite loop in the unlikely event where there are data samples of different classes accidentally positioned at an exactly identical location in the data space, namely, $\boldsymbol{x}_j = \boldsymbol{x}_i$ whilst $y_j \neq y_i$. Yet, the likelihood for this to take place is extremely low. Besides, it can be avoided by carrying out a simple procedure of data cleaning to explicitly block such partitions in advance.

Remark 3: The main aim of DHT is to identify a group of leaf prototypes at multiple levels of granularity with different descriptive abilities for classification, by recursively partitioning the data into non-overlapping multi-granular clusters one cycle after another. Once a leaf prototype is obtained, the associated pure cluster will no longer participate in the partitioning process further. Hence, the obtained classification model is unlikely to be overfitting because leaf prototypes will only distribute densely in the areas where class overlaps are observed, unless the data itself rejects the cluster hypothesis, which is extremely rare in practice. On the other hand, users may further consider terminating the system identification process earlier, by pre-setting the maximum model depth (number of layers) over which DHT may achieve. However, this would require prior knowledge of the problem under consideration in order to achieve the best classification performance, reducing the pure data-driven nature of the proposed approach.

Remark 4: The interpretability of the DHT models comes from three different aspects. First, all identified prototypes are data samples physically existing in the data space and are guaranteed to attain their inherent meanings. Second, the resulting prototypes are arranged in multiple layers according to their descriptive abilities. A smaller amount of more descriptive prototypes that represent global patterns of data are located at higher layers, whilst a greater amount of less descriptive prototypes that represent local patterns of data are located at lower layers. Users can utilize prototypes at higher layers to obtain a general picture of the problem and also, they can retrieve finer details of the problem based on prototypes at lower layers. Third, the relationships between prototypes identified at different levels of granularity are preserved by the models in the form of meaningful links, which are readily visualizable in a human-understandable manner. Users may use these links to interpret the internal system identification processes of the DHT models top-down.

2.4. Summary and Illustration of System identification Algorithm

The main system identification procedure is summarized in the form of pseudo code as given in ***Algorithm 1***.

Algorithm 1. Learning policy of DHT

i. calculate $\boldsymbol{D}$ from $\boldsymbol{X}$ as Eqn. (1);
ii. $g \leftarrow 1$;
iii. calculate σ_g by Eqn. (2);
iv. obtain $\boldsymbol{U}_g$ by Eqns. (3) and (4);
v. calculate $\lambda_g(\boldsymbol{x}_i)$ $(i = 1,2,\ldots,K)$ by Eqn. (5);
vi. identify $\boldsymbol{P}_g$ from $\boldsymbol{X}$ by **Condition 1** as Eqn. (6);
vii. obtain $\mathbb{C}_g$ from $\boldsymbol{P}_g$ and $\boldsymbol{X}$ by Eqn. (7);
viii. calculate $\rho_{g,i}$ of $\boldsymbol{C}_{g,i}$ $(i = 1,2,\ldots,P_g)$ by Eqn. (8);

ix. separate $\boldsymbol{P}_g$ to $\boldsymbol{P}_g^L$ and $\boldsymbol{P}_g^I$;
xi. **while** $\left(\boldsymbol{P}_g^I \neq \emptyset\right)$:
1. calculate σ_{g+1} by Eqn. (9);
2. $\boldsymbol{P}_{g+1} \leftarrow \emptyset$;
3. $\mathbb{C}_{g+1} \leftarrow \emptyset$;
4. **for** $i = 1$ **to** P_g^I **do:**
   * obtain $\boldsymbol{U}_{g+1,i}^I$ from $\boldsymbol{C}_{g,i}^I$ by Eqn. (10);
   * calculate $\lambda_{g+1,i}^I(\boldsymbol{z}_j)$ for $\forall \boldsymbol{z}_j \in \boldsymbol{C}_{g,i}^I$ by Eqn. (11);
   * identify $\boldsymbol{P}_{g+1,i}$ from $\boldsymbol{C}_{g,i}^I$ by **Condition 2** as Eqn. (12);
   * obtain $\mathbb{C}_{g+1,i}$ from $\boldsymbol{P}_{g+1,i}$ and $\boldsymbol{C}_{g,i}^I$ by Eqn. (13);
   * $\boldsymbol{P}_{g+1} \leftarrow \boldsymbol{P}_{g+1} \cup \boldsymbol{P}_{g+1,i}$;
   * $\mathbb{C}_{g+1} \leftarrow \mathbb{C}_{g+1} \cup \mathbb{C}_{g+1,i}$;
5. **end for**
6. calculate $\rho_{g+1,j}$ of $\mathbb{C}_{g+1}$ $(j = 1,2, \ldots, P_g)$ by Eqn. (8);
7. separate $\boldsymbol{P}_{g+1}$ into $\boldsymbol{P}_{g+1}^L$ and $\boldsymbol{P}_{g+1}^I$;
8. $g \leftarrow g + 1$;
xii. **end while**

To better illustrate the system identification process of DHT, a two-dimensional synthetic dataset is used as a visual example. This dataset considered is composed of 300 data samples of three different classes (100 samples per class), as visualized in Fig. 2, where dots "·" in three different colours represent data samples of three classes. The cumulative memberships of the 300 data samples calculated by Eqn. (11) at the first level of granularity ($g = 1$) are visualized in Fig. 3 (a), and the apex prototype identified from the dataset with the local maximum cumulative membership is given by Fig. 3(b), represented by the red asterisk "*". Note that as there is just one apex prototype identified, only one cluster containing all the 300 data samples is created. Hence, the entire dataset needs to be partitioned at a higher level of granularity, namely, $g = 2$. At this stage, the calculated cumulative memberships of the 300 data samples are visualized in Fig. 4(a). Two data samples with the local maximum cumulative membership are identified as prototypes, represented by blue asterisks "*", as shown in Fig. 4(b). Accordingly, two clusters are formed via using them as the cluster medoids to attract nearby data samples forming Voronoi tessellations. It can be seen from Fig. 4(b) that one of the clusters is formed by data samples of two classes, this cluster requires to be partitioned at the next level of granularity, $g = 3$. The obtained cumulative membership calculated locally within this cluster is presented in Fig. 5(a) and the identified prototypes are given in Fig. 5(b) as black asterisks "*". It follows from Fig. 5(b) that the cluster is partitioned into two smaller ones, each of which contains data samples of the same class. As data samples of three classes have been sufficiently separated, the system identification process terminates with the final three-layer prototype-based hierarchical structure obtained as visualized in Fig. 6.

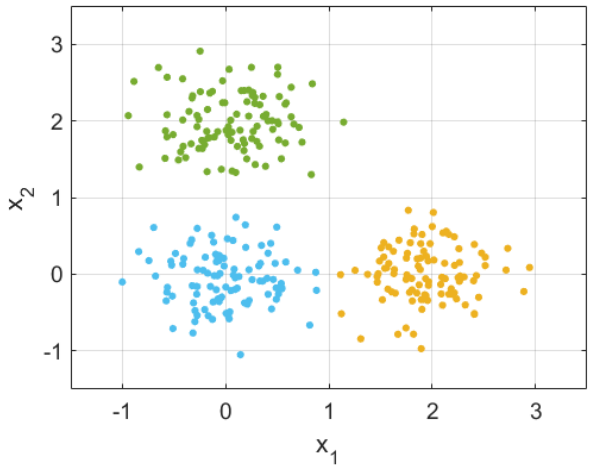
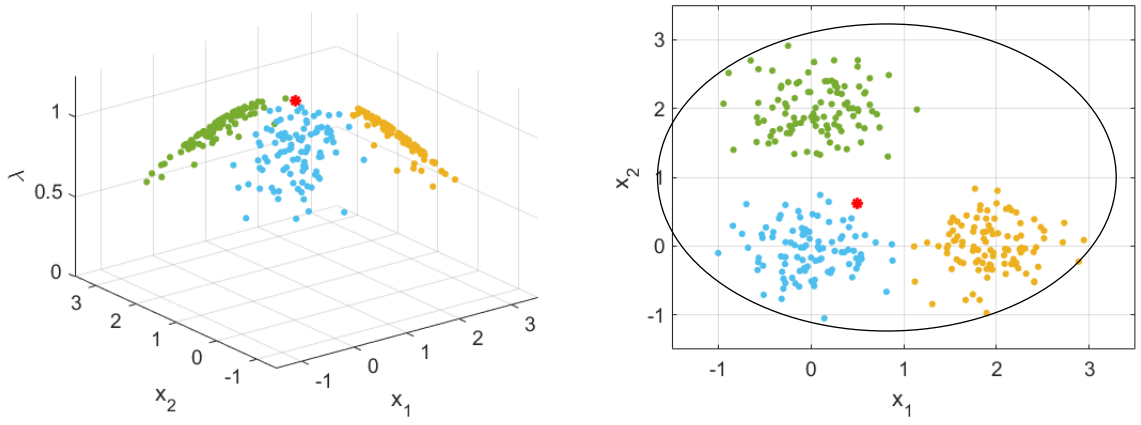
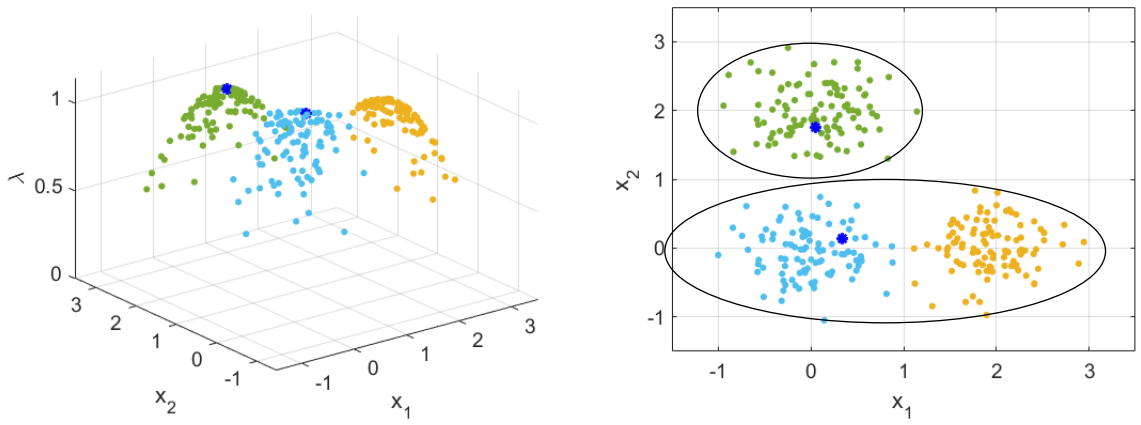


Fig. 2. Visualization of two-dimensional synthetic dataset composed of 300 samples of three classes (100 samples per class)



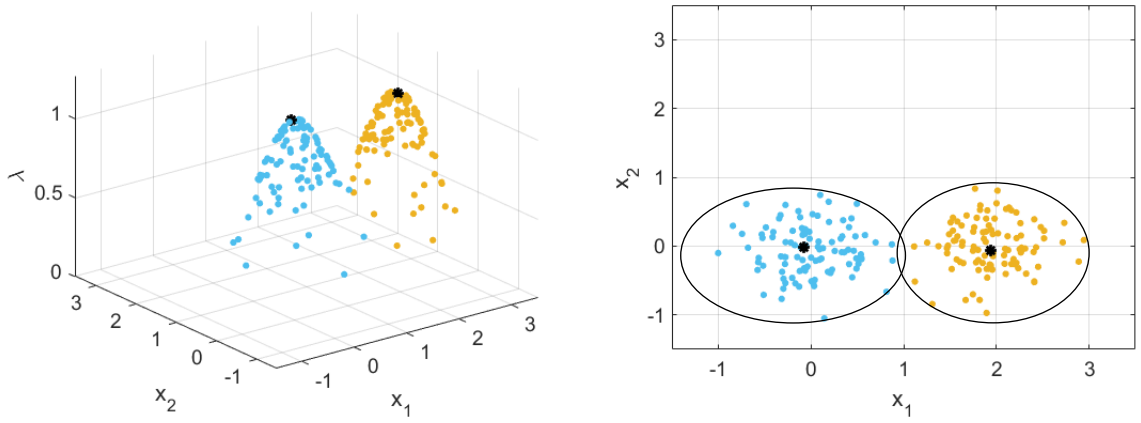
(a) Cumulative membership (b) Prototype

Fig. 3. Cumulative membership and identified prototype at the first level of granularity ($g = 1$)



(a) Cumulative membership (b) Prototypes

Fig. 4. Cumulative membership and identified prototypes at the second level of granularity ($g = 2$)



(a) Cumulative membership (b) Prototypes

Fig. 5. Cumulative membership and identified prototypes at the third level of granularity ($g = 3$)

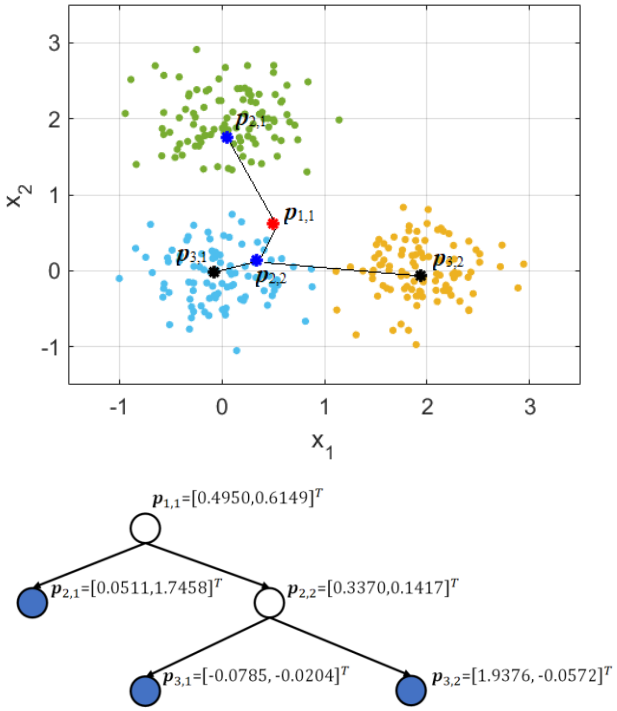


Fig. 6. Prototype-based hierarchical structure identified from data

2.5. Decision-Making Policy

During the decision-making stage, the decision-maker component of DHT determines the class labels of unlabelled data samples based on the distances between these unlabelled samples and the identified leaf prototypes. For a particular data sample, $\boldsymbol{x}_k$, its class label is decided by Eqn. (14):

$$\hat{y}_k \leftarrow y_{n^*}; \quad n^* = \underset{n=1,2,\ldots,P^L}{\operatorname{argmin}} \left(d^2(\boldsymbol{x}_k, \boldsymbol{p}_n)\right) \tag{14}$$

where $\boldsymbol{p}_n \in \boldsymbol{P}^L$; $\boldsymbol{P}^L$ is the collection of identified leaf prototypes; P^L is the cardinality of $\boldsymbol{P}^L$.

Remark 5: Since the class labels of unlabelled data samples are determined by the nearest leaf prototypes, if desired, users can track and trace the internal decision-making processes of the DHT models in a straightforward manner. Understanding about the rationales behind the decisions made can be obtained by exploiting the links constructed between prototypes at different layers (e.g., to appreciate how well an unlabelled data sample fits the related local and global patterns).

3. Computational Complexity Analysis

To reflect the efficiency of the proposed approach, a formal analysis of the computational complexity of the two main processes of DHT is provided herein.

3.1. Learning Process

Step A of the learning process of DHT starts by calculating a $K \times K$ dimensional pairwise distance matrix, $\boldsymbol{D}$ from $\boldsymbol{X}$. The computational complexity of calculating $\boldsymbol{D}$ by Eqn. (1) is $O(NK^2)$; that of deriving σ_g from $\boldsymbol{D}$ by Eqn. (2) is $O(K^2)$; and that of calculating the membership matrix, $\boldsymbol{U}_g$ and cumulative membership, λ from $\boldsymbol{D}$ and σ_g (Eqns. (3)-(5)) is $O(K^3)$. The computational complexity of identifying $\boldsymbol{P}_g$ from $\boldsymbol{X}$ by **Condition 1** is $O(K^2)$; that of forming Voronoi tessellations by Eqn. (7) is $O(NKP_g)$; and that of calculating the purity for each cluster is $O(KP_g)$. Therefore, the overall computational complexity of ***Step A*** is $O((N+K)K^2)$.

In ***Step B***, the obtained impure clusters are recursively partitioned at higher levels of granularity in order to separate data samples of different classes. The computational complexity of calculating σ_{g+1} by Eqn. (9) is $O(K^2)$; and that of calculating the local membership matrix, $\boldsymbol{U}^I_{g+1,i}$ and local cumulative membership, $\lambda^I_{g+1,i}$ for each

impure cluster, $\boldsymbol{C}_{g,i}^{I}$ by Eqns. (10)-(11) is $O\left(\left(S_{g,i}^{I}\right)^{3}\right)$. The computational complexity of identifying $\boldsymbol{p}_{g,i}^{I}$ from $\boldsymbol{C}_{g,i}^{I}$ using **Condition 2** is $O\left(\left(S_{g,i}^{I}\right)^{2}\right)$; and that of forming Voronoi tessellations within $\boldsymbol{C}_{g,i}^{I}$ by Eqn. (7) is $O\left(NS_{g,i}^{I}P_{g+1,i}\right)$. As there are a total of P_{g}^{I} impure clusters at the gth level of granularity, the overall computational complexity of ***Step B*** is $O\left(K^{2}+\sum_{i=1}^{P_{g}^{I}}\left(S_{g,i}^{I}\right)^{3}\right)$.

Despite that ***Step B*** may be repeated for a few times before the complete separation of data can be achieved, the computational complexity of each iteration is kept decreasing due to the continued reduction of the sizes of impure clusters obtained at higher levels of granularity. Hence, it can be concluded from the above that the overall computational complexity of DHT is $O\left((N+K)K^{2}\right)$.

3.2. Decision-Making Process

During the decision-making process, the computational complexity of calculating the distances between a particular testing sample and the learned leaf prototypes is $O(NP^{L})$. For K testing samples, the overall computational complexity to determine their class labels is $O(NKP^{L})$.

4. Experimental Investigation

In this section, experimental studies are conducted to evaluate the effectiveness and validity of the proposed DHT algorithm.

4.1. Configuration

In this work, a total of 21 commonly used benchmark numerical datasets (including 10 binary and 11 multi-class ones) and two real-world remote sensing image sets are utilized in the experimental investigations. Key information regarding the 23 datasets is summarized in Tables 2-4, and the web links to these datasets are given in Table 5.

Table 2. Key information of binary benchmark classification problems

Dataset		Abbreviation	#(Samples)	#(Attributes)	#(Minority)	#(Majority)
Epileptic seizure recognition		ES	11500	175+1 label	2300	9200
German credit		GC	1000	24+1 label	300	700
Mammography		MA	11183	6+1 label	260	10923
Magic gamma telescope		MG	19020	10+1 label	6688	12332
Occupancy detection	Training	OD	8143	5+1 label	1729	6414
	Testing		12417		3021	9396
Phishing websites		PW	11055	30+1 label	4898	6157
Shill bidding		SB	6321	9+1 label	675	5646
Seismic		SE	2584	18+1 label	170	2414
Spambase		SP	4601	57+1 label	1813	2788
Wilt	Training	WI	4339	5+1 label	74	4265
	Testing		500		187	313

Table 3. Key information of multi-class benchmark classification problems

Dataset		Abbreviation	#(Samples)	#(Attributes)	#(Classes)
Iris		IR	150	4+1 label	3
Cardiotocography		CA	2126	21+1 label	3
Gesture phase segmentation		GP	9901	18+1 label	5
Image segmentation	Training	IS	420	19+1 label	7
	Testing		2100		
Letter recognition		LR	20000	16+1 label	26
Multiple features		MF	2000	649+1 label	10
Page-blocks		PB	5473	10+1 label	5
Pen-based recognition of handwritten digits		PR	10996	16+1 label	10
Semeion handwritten digit		SH	1593	256+1 label	10
Steel plates faults		SPF	1941	27+1 label	7
Wall-following robot navigation		WF	5456	24+1 label	4

Table 4. Key information of benchmark remote sensing image sets for land-use classification

Dataset	#(Images)	#(Categories)	#(Images per category)	Image size
OPTIMAL-31	1860	31	60	256×256
RSI-CB256	24747	35	198~1331	256×256

Table 5. Web links to benchmark datasets involved in experimental investigation

Dataset	Web link
ES	https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition
GC	https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)
MA	http://odds.cs.stonybrook.edu/mammography-dataset/
MG	https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope
OD	https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+
PW	https://archive.ics.uci.edu/ml/datasets/phishing+websites
SB	https://archive.ics.uci.edu/ml/datasets/Shill+Bidding+Dataset
SE	https://archive.ics.uci.edu/ml/datasets/seismic-bumps
SP	https://archive.ics.uci.edu/ml/datasets/Spambase
WI	http://archive.ics.uci.edu/ml/datasets/wilt
IR	https://archive.ics.uci.edu/ml/datasets/iris
CA	https://archive.ics.uci.edu/ml/datasets/cardiotocography
GP	https://archive.ics.uci.edu/ml/datasets/gesture+phase+segmentation
IS	https://archive.ics.uci.edu/ml/datasets/image+segmentation
LR	https://archive.ics.uci.edu/ml/datasets/Letter+Recognition
MF	https://archive.ics.uci.edu/ml/datasets/Multiple+Features
PB	https://archive.ics.uci.edu/ml/datasets/Page+Blocks+Classification
PR	https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits
SH	https://archive.ics.uci.edu/ml/datasets/semeion+handwritten+digit
SPF	https://archive.ics.uci.edu/ml/datasets/Steel+Plates+Faults
WF	https://archive.ics.uci.edu/ml/datasets/Wall-Following+Robot+Navigation+Data
OPTIMAL-31	https://drive.google.com/file/d/1Fk9a0DW8UyyQsR8dP2Qdakmr69NVBhq9
RSI-CB256	https://github.com/lehaifeng/RSI-CB

Binary Classification Problems. In running the experiments, for the OD and WI datasets, the original training-testing splits are used. For the other eight binary benchmark datasets, namely, ES, GC, MA, MG, PW, SB, SE and SP, 50% of the data samples are randomly selected for building the training sets and the rest for testing [28].

The performance of DHT is evaluated on the 10 binary classification datasets in terms of classification accuracy (Acc) and execution time (t_{exe}). As some of the binary classification datasets are highly imbalanced, the standard classification accuracy may not be the best performance index. Hence, the following two additional measures are also employed: balanced accuracy score ($BAcc$) [33] and F1 score ($F1$). Expressions of Acc, $BAcc$ and $F1$ are given by Eqns. (15a), (15b) and (15c).

$$Acc = \frac{TP+TN}{TP+FP+TN+FN} \tag{15a}$$

$$BAcc = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{15b}$$

$$F1 = \frac{TP}{TP+\frac{1}{2}(FP+FN)} \tag{15c}$$

where TP, TN, FP and FN denote true positive, true negative, false positive and false negative, respectively.

Multi-Class Classification Problems. Thanks to its smaller scale and simpler structure, the IR dataset is taken to provide visual illustration. The remaining ten multi-class benchmark datasets are used for performance evaluation. Similar to the experiment protocols used for binary classification problems, the original training-testing splits of the IS and PR dataset are retained. For the other eight datasets, including CA, GP, LR, MF, PB, SH, SPF and WF, 50% of data samples are randomly selected for training with the remaining for testing [28]. Acc and t_{exe} are used as the two criteria to evaluate the working of DHT on the 10 multi-class classification problems.

Remote Sensing Image Classification Problems. To examine the capability of DHT to handle high-dimensional, complex problems, two popular remote sensing image sets for land-use classification problems are employed. In carrying out the experiments, three popular DNNs including ResNet50 [34], DenseNet121 [35] and InceptionV3 [36] are exploited for feature extraction after fine-tuning on the NWPU45 dataset, following the same process as described in [37]. Each of the three DNNs extracts a 1024×1 dimensional feature vector from every remote sensing image. The three resulting feature vectors are combined into a more descriptive high-level representation by arithmetic mean. Hence, for each image within the datasets, a 1024×1 dimensional high-level representation is extracted. As with the common practice in the literature, the training-testing split ratio of OPTIMAL-31 is set to 8:2; and for the RSI-CB256 dataset, two different split ratios are considered, namely, 5:5 and 8:2. The classification performances of DHT on the two remote sensing image classification problems are measured with respect to Acc, as commonly done in the literature.

State-of-the-Art Methods for Benchmark Comparison. For better evaluation, the following 14 mainstream classification approaches are involved for benchmark comparison on the 20 numerical datasets, under the same experimental protocols used by DHT.

1) Multi-granularity locally optimal prototype-based (MLOP) classifier [28];

2) Hierarchical prototype-based (HP) classifier [27];

3) SVM classifier with linear kernel (SVM-L) [10];

4) SVM classifier with Gaussian kernel (SVM-G) [10];

5) DT classifier [12];

6) KNN classifier [13];

7) Multilayer perceptron (MLP);

8) SC [20];

9) Sequence-dictionary-based KNN (SDKNN) classifier [20];

10) Zero-order autonomous learning multiple-model (ALMMo0) classifier [19];

11) Self-organising fuzzy inference system (SOFIS) [16];

12) Extreme learning machine (ELM) [38];

13) EigenClass [21], and;

14) Generalized learning vector quantization (GLVQ) [39].

In running these experiments, system parameters are set with respect to the commonly adopted default values [28]. Particularly, the layer number of HP is set as $H = 6$ [27]; the regularization parameter of MLOP is set as $\rho = 0.05$ [28]; SC uses the recommended setting given by [20]; the box constraint, C for SVM is set as $C = 1$; both KNN and SDKNN use $k = 5$; the maximum depth of DT is set as $K - 1$; MLP has three hidden layers with 20 neurons per layer; the level of granularity of SOFIS is set as $G = 12$ [16]; the maximum number of neurons for ELM is set as 200; EigenClass considers the first five eigenvalues for classification; and GLVQ has 25 reference vectors per class with the gain factor set to $\alpha = 0.005$.

In addition, three evolutionary algorithms are also employed in experimental investigation, including:

1) Genetic algorithm (GA) [40];

2) Particle swarm optimization algorithm (PSO) [41], and;

3) Genetic learning particle swarm optimization algorithm (GLPSO) [42].

In these experiments, GA, PSO, GLPSO are utilized to optimize the learned prototypes by SOFIS from data, under a similar experimental protocol as used in [43]. Following the common practice in the literature, the crossover probability p_c, mutation probability p_m, distribution indexes for crossover and mutation operators, η_c and η_c for

GA are set as: $p_c = 0.9$; $p_m = \frac{1}{N}$; $\eta_c = 20$ and $\eta_m = 20$, respectively. The externally controlled parameters for PSO are set as: $\omega = 0.7298$; $c_1 = c_2 = 1.49618$, and that for GLPSO are set as: $\omega = 0.7298$; $c = c_1 = c_2 = 1.49618$; $p_m = 0.01$ [42], [43]. The population size is set to 100, the maximum number of iterations is 200 for each of the three evolutionary algorithms, and the fitness of the solutions is evaluated on the basis of the classification error rate $(1 - Acc)$ on the training data. The optimized SOFISs by GA, PSO and GLPSO are denoted as GA-SOFIS, PSO-SOFIS and GLPSO-SOFIS. Here, the level of granularity is set as $G = 9$ for GA-SOFIS, PSO-SOFIS and GLPSO-SOFIS to avoid overfitting.

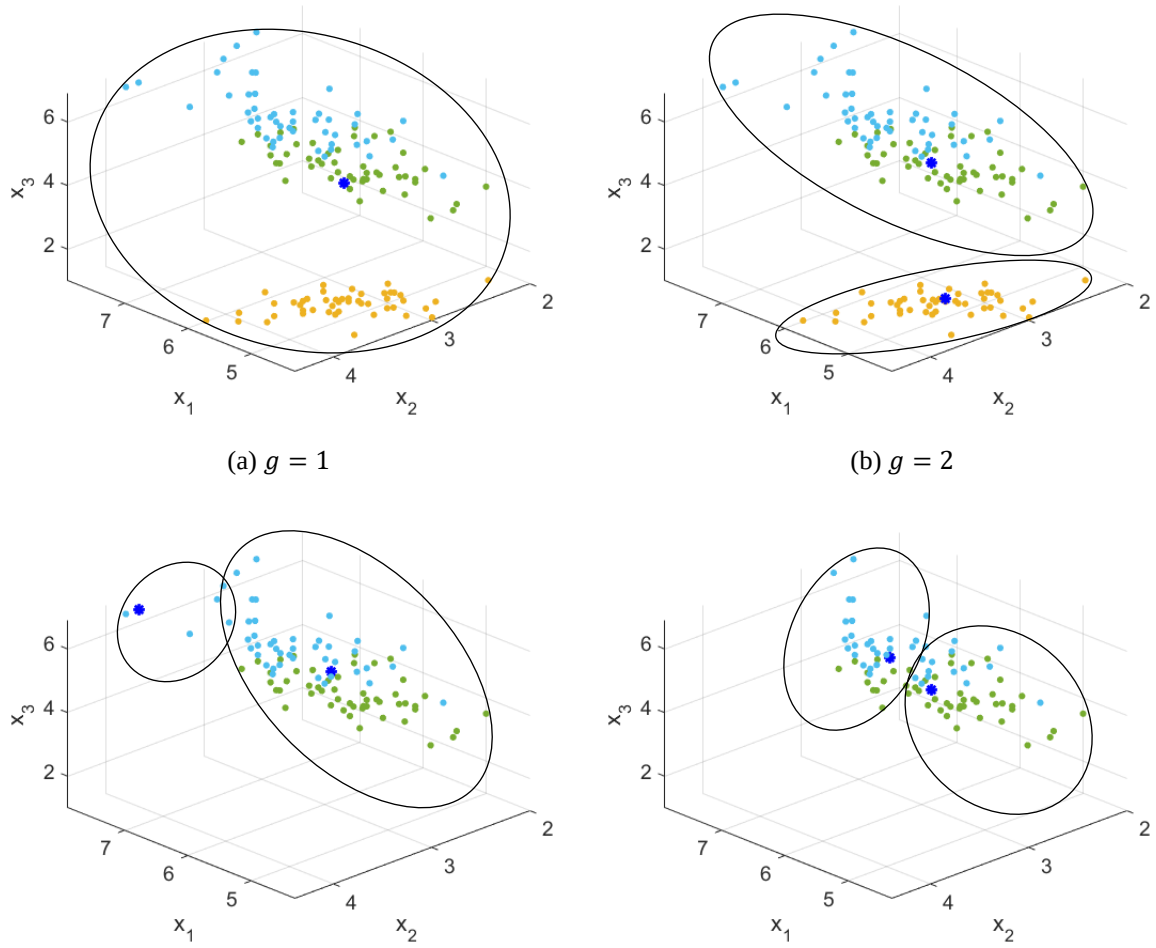
The proposed approach is implemented on the MATLAB2020b platform, and the performance evaluation is conducted on a laptop with dual core i7 processer 2.60GHz×2 and 16.0GB RAM. Unless otherwise stated, the reported results are obtained after 10 Monte Carlo experiments to allow a certain degree of randomness and hence, a fair comparison. The MATLAB code of DHT is publicly available at: https://github.com/Gu-X/Self-Organizing-Divisive-Hierarchical-Voronoi-Tessellation-Based-Classifier.

4.2. Visual Illustration

The IR dataset is employed as the first example to illustrate the proposed concept. In this case study, all data samples available are used for training. During the experiment, DHT repeatedly partitions the dataset at seven different levels of granularity until a clear separation of data samples of the three classes is achieved. The recursive partitioning results are visualized in Fig. 7, where dots "·" in three different colours represent data samples of three classes and the blue asterisks "*" represent the identified prototypes. Whilst DHT self-constructs a seven-layer prototype-based hierarchical structure from data, for visual clarity, only the partitioning results obtained at the first four levels of granularity are given. The constructed prototype-based hierarchy is presented in Fig. 8, and the corresponding prototypes are listed in Table 6.

Table 6. Identified prototypes from IR dataset

Prototype	Prototype	Prototype
$\boldsymbol{p}_{1,1} = [5.7,3.0,4.2,1.2]^T$	$\boldsymbol{p}_{5,14} = [7.4,2.8,6.1,1.9]^T$	$\boldsymbol{p}_{6,19} = [6.3,2.5,5.0,1.9]^T$
$\boldsymbol{p}_{2,1} = [5.0,3.4,1.6,0.4]^T$	$\boldsymbol{p}_{5,15} = [7.7,2.6,6.9,2.3]^T$	$\boldsymbol{p}_{6,20} = [6.2,3.4,5.4,2.3]^T$
$\boldsymbol{p}_{2,2} = [6.0,2.9,4.5,1.5]^T$	$\boldsymbol{p}_{6,1} = [4.9,2.5,4.5,1.7]^T$	$\boldsymbol{p}_{6,21} = [6.3,3.3,4.7,1.6]^T$
$\boldsymbol{p}_{3,1} = [6.2,2.8,4.8,1.8]^T$	$\boldsymbol{p}_{6,2} = [5.2,2.7,3.9,1.4]^T$	$\boldsymbol{p}_{6,22} = [6.7,3.1,4.7,1.5]^T$
$\boldsymbol{p}_{3,2} = [7.7,3.8,6.7,2.2]^T$	$\boldsymbol{p}_{6,3} = [5.5,2.5,4.0,1.3]^T$	$\boldsymbol{p}_{6,23} = [7.0,3.2,4.7,1.4]^T$
$\boldsymbol{p}_{4,1} = [6.0,2.9,4.5,1.5]^T$	$\boldsymbol{p}_{6,4} = [5.5,2.6,4.4,1.2]^T$	$\boldsymbol{p}_{6,24} = [6.3,3.3,6.0,2.5]^T$
$\boldsymbol{p}_{4,2} = [6.5,3.0,5.2,2.0]^T$	$\boldsymbol{p}_{6,5} = [5.7,2.5,5.0,2.0]^T$	$\boldsymbol{p}_{6,25} = [6.4,2.8,5.6,2.1]^T$
$\boldsymbol{p}_{5,1} = [4.9,2.5,4.5,1.7]^T$	$\boldsymbol{p}_{6,6} = [5.8,2.7,5.1,1.9]^T$	$\boldsymbol{p}_{6,26} = [6.4,3.2,5.3,2.3]^T$
$\boldsymbol{p}_{5,2} = [5.0,2.0,3.5,1.0]^T$	$\boldsymbol{p}_{6,7} = [6.0,2.7,5.1,1.6]^T$	$\boldsymbol{p}_{6,27} = [6.7,3.1,5.6,2.4]^T$
$\boldsymbol{p}_{5,3} = [5.0,2.3,3.3,1.0]^T$	$\boldsymbol{p}_{6,8} = [6.1,2.6,5.6,1.4]^T$	$\boldsymbol{p}_{6,28} = [6.8,3.0,5.5,2.1]^T$
$\boldsymbol{p}_{5,4} = [6.0,2.2,4.0,1.0]^T$	$\boldsymbol{p}_{6,9} = [6.2,2.8,4.8,1.8]^T$	$\boldsymbol{p}_{7,1} = [6.0,2.2,5.0,1.5]^T$
$\boldsymbol{p}_{5,5} = [6.0,2.7,5.1,1.6]^T$	$\boldsymbol{p}_{6,10} = [6.3,2.8,5.1,1.5]^T$	$\boldsymbol{p}_{7,2} = [6.2,2.2,4.5,1.5]^T$
$\boldsymbol{p}_{5,6} = [6.0,2.9,4.5,1.5]^T$	$\boldsymbol{p}_{6,11} = [5.7,2.9,4.2,1.3]^T$	$\boldsymbol{p}_{7,3} = [6.3,2.3,4.4,1.3]^T$
$\boldsymbol{p}_{5,7} = [6.2,2.2,4.5,1.5]^T$	$\boldsymbol{p}_{6,12} = [5.9,3.2,4.8,1.8]^T$	$\boldsymbol{p}_{7,4} = [6.5,3.0,5.5,1.8]^T$
$\boldsymbol{p}_{5,8} = [6.3,2.5,4.9,1.5]^T$	$\boldsymbol{p}_{6,13} = [6.0,2.9,4.5,1.5]^T$	$\boldsymbol{p}_{7,5} = [6.5,3.0,5.8,2.2]^T$
$\boldsymbol{p}_{5,9} = [5.8,2.8,5.1,2.4]^T$	$\boldsymbol{p}_{6,14} = [6.0,3.0,4.8,1.8]^T$	$\boldsymbol{p}_{7,6} = [6.7,3.0,5.0,1.7]^T$
$\boldsymbol{p}_{5,10} = [6.3,3.3,4.7,1.6]^T$	$\boldsymbol{p}_{6,15} = [6.0,3.4,4.5,1.6]^T$	$\boldsymbol{p}_{7,7} = [6.7,3.0,5.2,2.3]^T$
$\boldsymbol{p}_{5,11} = [6.7,2.5,5.8,1.8]^T$	$\boldsymbol{p}_{6,16} = [6.4,3.2,4.5,1.5]^T$	$\boldsymbol{p}_{7,8} = [6.8,3.0,5.5,2.1]^T$
$\boldsymbol{p}_{5,12} = [6.8,2.8,4.8,1.4]^T$	$\boldsymbol{p}_{6,17} = [6.2,2.2,4.5,1.5]^T$	$\boldsymbol{p}_{7,9} = [7.1,3.0,5.9,2.1]^T$
$\boldsymbol{p}_{5,13} = [6.8,3.0,5.5,2.1]^T$	$\boldsymbol{p}_{6,18} = [6.3,2.5,4.9,1.5]^T$	



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

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

Fig. 7. Partitioning results obtained at four different levels of granularity

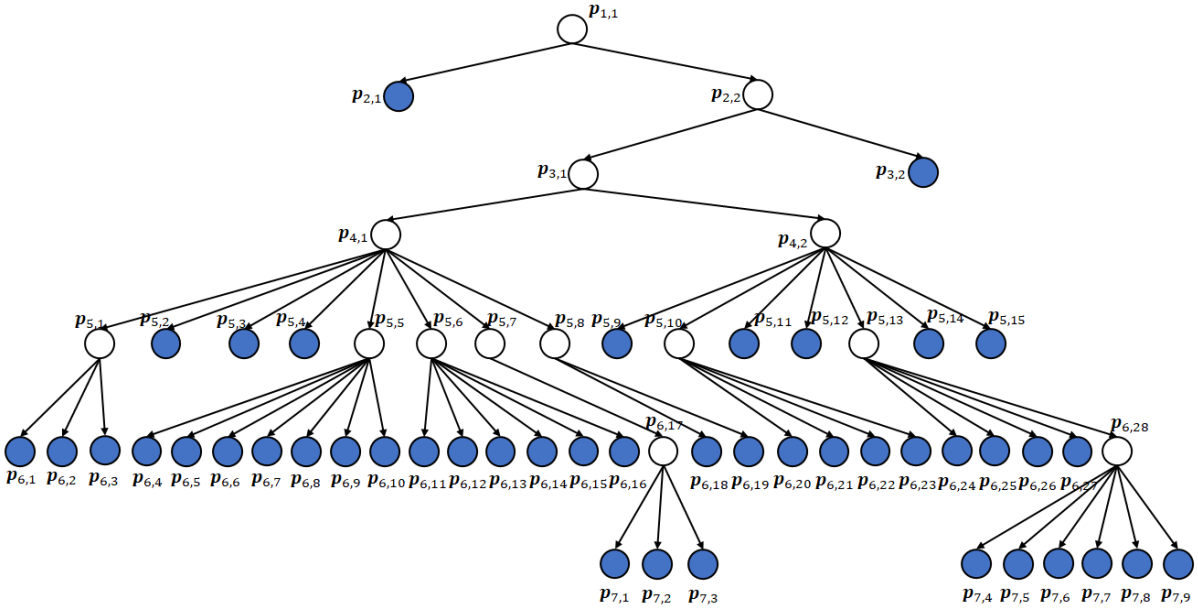


Fig. 8. Constructed seven-layer prototype-based hierarchical structure from data

The numerical example shown by Figs. 7-8 and Table 6 demonstrates the operating process of DHT. In particular, DHT obtains an initial partition of the IR dataset at the first level of granularity ($g = 1$), where only one large

cluster is created (see Fig. 7(a)). Next, DHT splits the large cluster at the second level of granularity ($g = 2$) and obtains two smaller ones (see Fig. 7(b)). Then, the proposed algorithm selects an impure one from these two newly obtained clusters and continues to partition it at a higher level of granularity (see Fig. 7(c)). The same process is repeated until data samples of different classes have been well separated. In this way, DHT self-organizes a prototype-based hierarchical structure from data as given in Fig. 8.

4.3. Performance Examination

Binary Classification Problems. Firstly, the performance of the proposed DHT classifier is evaluated on 10 binary benchmark classification problems. The results, in terms of Acc, $BAcc$ and $F1$, obtained by DHT and the 17 comparative algorithms on each benchmark problem are reported in Tables 7-9, respectively. In addition, the average performance measures ($Acc, BAcc,\ F1$ and t_{exe}) of each classification approach across the 10 problems are given in Table 10, and the ranks per measure are presented within the same table. The results obtained by the proposed approach are shown in bold for visual clarity.

Table 7. Classification accuracy (Acc) of different classification approaches on binary benchmark classification problems

Algorithm	Dataset									
	ES	GC	MA	MG	OD	PW	SB	SE	SP	WI
DHT	**0.9388**	**0.6636**	**0.9715**	**0.7839**	**0.9170**	**0.9217**	**0.9792**	**0.8672**	**0.8635**	**0.7840**
MLOP	0.9190	0.6414	0.8306	0.7646	0.9441	0.9359	0.9958	0.8301	0.6678	0.7760
HP	0.8695	0.6392	0.6667	0.6851	0.7932	0.9194	0.9835	0.7796	0.7073	0.8076
SVM-L	0.2157	0.7560	0.9810	0.7122	0.8436	0.9277	0.9793	0.6694	0.8889	0.7150
SVM-G	0.7979	0.6944	0.9825	0.6576	0.7607	0.8587	0.9979	0.9342	0.6902	0.6280
DT	0.9344	0.7000	0.9803	0.8181	0.9314	0.9472	0.9971	0.9005	0.9042	0.8140
KNN	0.9150	0.6912	0.9848	0.8040	0.9580	0.9303	0.9966	0.9301	0.7822	0.7260
MLP	0.9394	0.7154	0.9840	0.8550	0.9311	0.9391	0.9890	0.9338	0.8617	0.6376
SC	0.9202	0.7038	0.5203	0.7788	0.7351	0.9094	0.9785	0.9300	0.8865	0.3980
SDKNN	0.9449	0.6616	0.6259	0.7764	0.5189	0.9470	0.9808	0.8913	0.8853	0.6720
ALMMo0	0.8939	0.6662	0.7040	0.7248	0.9438	0.9426	0.9906	0.8882	0.7826	0.7614
SOFIS	0.9218	0.6488	0.8330	0.7695	0.9588	0.9361	0.9964	0.9049	0.7637	0.7560
GA-SOFIS	0.9170	0.6570	0.9819	0.7667	0.9547	0.9361	0.9958	0.9322	0.7577	0.7668
PSO-SOFIS	0.9154	0.6514	0.9832	0.7655	0.9588	0.9380	0.9951	0.9309	0.7491	0.7578
GLPSO-SOFIS	0.9145	0.6434	0.9828	0.7651	0.9592	0.9375	0.9957	0.9318	0.7495	0.7552
ELM	0.2737	0.7790	0.9681	0.5500	0.9894	0.9104	0.9944	0.8400	0.8772	0.8594
EigenClass	0.8010	0.7060	0.8936	0.8092	0.9615	0.9461	0.9704	0.9331	0.8583	0.7080
GLVQ	0.7707	0.7346	0.9655	0.8083	0.9314	0.9063	0.9777	0.8317	0.8420	0.6260

Table 8. Balanced classification accuracy ($BAcc$) of different classification approaches on binary benchmark classification problems

Algorithm	Dataset									
	ES	GC	MA	MG	OD	PW	SB	SE	SP	WI
DHT	**0.8647**	**0.5997**	**0.8108**	**0.7575**	**0.8759**	**0.9208**	**0.9624**	**0.5560**	**0.8567**	**0.7349**
MLOP	0.8061	0.5616	0.6750	0.7199	0.9186	0.9357	0.9854	0.5416	0.6188	0.7048
HP	0.7901	0.5815	0.6191	0.6669	0.7678	0.9171	0.9750	0.5802	0.7023	0.7761
SVM-L	0.3759	0.6772	0.6375	0.6706	0.8082	0.9255	0.9837	0.5245	0.8764	0.6427
SVM-G	0.5000	0.5000	0.6770	0.5089	0.5084	0.8443	0.9935	0.5000	0.6093	0.5027
DT	0.8956	0.6383	0.7637	0.7995	0.8981	0.9460	0.9898	0.5437	0.8998	0.7653
KNN	0.7916	0.5753	0.7496	0.7541	0.9547	0.9291	0.9878	0.5066	0.7703	0.6348
MLP	0.8737	0.6094	0.7486	0.8220	0.9142	0.9375	0.9778	0.5031	0.8506	0.5164
SC	0.8681	0.5890	0.6107	0.7324	0.6164	0.9446	0.9620	0.5268	0.8868	0.5776
SDKNN	0.8045	0.5997	0.6901	0.6963	0.6802	0.9083	0.9646	0.5185	0.8787	0.5192
ALMMo0	0.8202	0.5983	0.6114	0.6868	0.9161	0.9415	0.9801	0.5202	0.7891	0.6905
SOFIS	0.8117	0.5717	0.7015	0.7333	0.9600	0.9359	0.9884	0.5308	0.7526	0.6792
GA-SOFIS	0.8000	0.5788	0.7213	0.7260	0.9490	0.9359	0.9864	0.5020	0.7388	0.6920
PSO-SOFIS	0.7948	0.5769	0.7471	0.7260	0.9570	0.9376	0.9836	0.5050	0.7296	0.6800
GLPSO-SOFIS	0.7928	0.5664	0.7370	0.7250	0.9570	0.9371	0.9864	0.5080	0.7301	0.6760
ELM	0.5327	0.7374	0.7560	0.5284	0.9911	0.9059	0.9844	0.5710	0.8672	0.8218
EigenClass	0.5078	0.6100	0.6222	0.7420	0.9499	0.9455	0.8771	0.5087	0.8270	0.6107
GLVQ	0.5951	0.7011	0.7421	0.7607	0.9344	0.9027	0.9486	0.6376	0.8254	0.5000

Table 9. F1 scores ($F1$) of different classification approaches on binary benchmark classification problems

Algorithm	Dataset									
	ES	GC	MA	MG	OD	PW	SB	SE	SP	WI
DHT	**0.8304**	**0.4396**	**0.5260**	**0.8359**	**0.8234**	**0.9293**	**0.9062**	**0.9278**	**0.8270**	**0.6516**
MLOP	0.7546	0.3765	0.3162	0.8275	0.8832	0.9420	0.9802	0.9051	0.4724	0.5852
HP	0.6703	0.4204	0.0772	0.7505	0.6279	0.9282	0.9262	0.8718	0.6473	0.7166
SVM-L	0.2495	0.5396	0.4058	0.7837	0.7644	0.9356	0.9106	0.7365	0.8471	0.4827
SVM-G	0.0000	0.0000	0.4972	0.7918	0.0332	0.8846	0.9904	0.9660	0.3591	0.0106
DT	0.8365	0.4933	0.5707	0.8604	0.8552	0.9527	0.9864	0.9471	0.8791	0.6971
KNN	0.7353	0.3516	0.6173	0.8593	0.9165	0.9374	0.9840	0.9637	0.7218	0.4268
MLP	0.7990	0.3904	0.6029	0.8932	0.8626	0.9455	0.9491	0.9658	0.8203	0.0586
SC	0.8444	0.4188	0.0722	0.8364	0.4492	0.9529	0.9124	0.9421	0.8607	0.3167
SDKNN	0.7554	0.4038	0.0913	0.8509	0.5130	0.9183	0.9038	0.9636	0.8546	0.5541
ALMMo0	0.7263	0.4348	0.1094	0.7937	0.8818	0.9484	0.9564	0.9403	0.7494	0.5608
SOFIS	0.7640	0.3915	0.3397	0.8282	0.9191	0.9422	0.9830	0.9497	0.7010	0.5344
GA-SOFIS	0.7460	0.4006	0.5383	0.8280	0.9090	0.9422	0.9803	0.9650	0.6791	0.5580
PSO-SOFIS	0.7387	0.4015	0.5901	0.8270	0.9180	0.9440	0.9770	0.9640	0.6675	0.5330
GLPSO-SOFIS	0.7355	0.3847	0.5747	0.8260	0.9190	0.9435	0.9796	0.9650	0.6683	0.5260
ELM	0.3810	0.6346	0.5832	0.4844	0.9786	0.9215	0.9737	0.8652	0.8398	0.7815
EigenClass	0.0308	0.4237	0.3455	0.8681	0.9214	0.9513	0.8452	0.9654	0.7899	0.3652
GLVQ	0.3460	0.5845	0.4488	0.8619	0.8696	0.9173	0.8972	0.9040	0.7884	0.0000

Table 10. Overall performances and ranks of different classification approaches on binary benchmark classification problems

Algorithm	Acc		$BAcc$		$F1$		t_{exe}	
	Average	Rank	Average	Rank	Average	Rank	Average	Rank
DHT	**0.8690**	**4**	**0.7939**	**2**	**0.7697**	**2**	**3.1203**	**9**
MLOP	0.8305	11	0.7468	12	0.7043	11	5.6767	11
HP	0.7851	16	0.7376	13	0.6636	14	1.8165	8
SVM-L	0.7689	18	0.7122	17	0.6655	13	30.1451	14
SVM-G	0.8002	14	0.6144	18	0.4533	18	1.0589	6
DT	0.8927	1	0.8140	1	0.8078	1	0.0577	3
KNN	0.8718	3	0.7654	6	0.7514	6	0.0197	1
MLP	0.8786	2	0.7753	3	0.7287	9	0.4322	4
SC	0.7761	17	0.7314	14	0.6606	16	5.1244	10
SDKNN	0.7904	15	0.7260	15	0.6809	12	11.6159	12
ALMMo0	0.8298	12	0.7554	10	0.7101	10	1.7216	7
SOFIS	0.8489	9	0.7665	5	0.7353	8	0.7829	5
GA-SOFIS	0.8666	5	0.7630	8	0.7546	4	664.0393	16
PSO-SOFIS	0.8645	6	0.7638	7	0.7561	3	608.3320	15
GLPSO-SOFIS	0.8635	7	0.7616	9	0.7522	5	1188.7767	18
ELM	0.8042	13	0.7696	4	0.7444	7	0.0462	2
EigenClass	0.8587	8	0.7201	16	0.6506	17	865.4351	17
GLVQ	0.8394	10	0.7548	11	0.6618	15	22.7965	13

It can be seen from Table 10 that the average Acc, $BAcc$ and $F1$ of DHT obtained on the 10 binary benchmark classification problems are 0.8690, 0.7939 and 0.7697, respectively. The average Acc of DHT is ranked at the fourth place over the 18 classification approaches involved in the experiments, whilst its average $BAcc$ and $F1$ are both ranked at the second place. The results show that DHT is able to achieve very high performance on binary classification problems including the imbalanced ones. Note that such excellent performance is achieved by DHT with a practically acceptable computational efficiency. Whilst DT offers the best performance overall, it operates at attribute level and builds the classification model via splitting data based on these more important attributes. This mechanism is highly effective on simpler problems, enabling DT to outperform DHT as well as other classifiers on binary classification tasks. In contrast, DHT builds a hierarchical classification model via identifying prototypes at multiple levels of granularity and recursively partitioning the data [31]. Hence, the resulting model can provide more intuitive information about the underlying patterns and multi-model distribution of the given data. Nonetheless, as to be shown next, for more challenging application problems concerning multi-class classification, DHT is able to perform the best, beating DT in accuracy.

Note that evolutionary algorithms help SOFIS achieve greater classification performance in terms of Acc and $F1$. PSO-SOFIS ranks the third place over the 18 classification methods on $F1$, and GA-SOFIS ranks the fifth on Acc. However, such improvement comes at the price of much higher computational resource consumption, which is due to the iterative optimization processes required by the evolutionary algorithms.

Multi-class Classification Problems. Next, the performance of DHT is evaluated on 10 multi-class benchmark classification problems, and compared with the 17 comparative classification algorithms as listed in Section 4.1. The performances of the involved classification approaches on the 10 multi-class classification problems, in terms of Acc, are reported in Table 11. A high-level summary of the results (Acc and t_{exe}) obtained by the 18 classification approaches is given in Table 12, including the respective rankings as per each of the two performance criteria.

Table 11. Classification accuracy (Acc) of different classification approaches on multi-class benchmark classification problems

Algorithm	Dataset									
	CA	GP	IS	LR	MF	PB	PR	SH	SPF	WF
DHT	**0.8810**	**0.8777**	**0.8876**	**0.9221**	**0.9634**	**0.9450**	**0.9651**	**0.8700**	**0.7052**	**0.8454**
MLOP	0.8773	0.6980	0.7819	0.9281	0.9148	0.9340	0.9757	0.8593	0.3727	0.8404
HP	0.8127	0.6597	0.7929	0.9153	0.9250	0.8845	0.9671	0.8794	0.2234	0.8360
SVM-L	0.8748	0.5681	0.9405	0.8552	0.9696	0.9255	0.9551	0.9154	0.1376	0.7315
SVM-G	0.7805	0.7675	0.2369	0.3799	0.1026	0.9363	0.1039	0.0895	0.3507	0.7344
DT	0.9117	0.8192	0.9048	0.8236	0.9214	0.9644	0.9122	0.6925	0.6950	0.9912
KNN	0.8807	0.8080	0.8438	0.9325	0.9290	0.9510	0.9760	0.8770	0.4382	0.8297
MLP	0.8892	0.6099	0.8878	0.4660	0.8420	0.9455	0.9172	0.5706	0.6790	0.8273
SC	0.8767	0.7619	0.8381	0.8339	0.9748	0.9616	0.9531	0.8172	0.6890	0.8963
SDKNN	0.8890	0.8410	0.8824	0.8561	0.9760	0.9423	0.9511	0.8935	0.6823	0.9284
ALMMo0	0.8432	0.7104	0.7671	0.9190	0.9329	0.9492	0.9753	0.8918	0.3438	0.8352
SOFIS	0.8792	0.7819	0.7976	0.9289	0.9200	0.9414	0.9763	0.8973	0.4095	0.8406
GA-SOFIS	0.8798	0.7859	0.8040	0.9290	0.9189	0.9455	0.9753	0.8971	0.4748	0.8414
PSO-SOFIS	0.8783	0.7796	0.7966	0.9266	0.9172	0.9457	0.9737	0.8894	0.4842	0.8419
GLPSO-SOFIS	0.8793	0.7799	0.8009	0.9269	0.9159	0.9459	0.9734	0.8902	0.4791	0.8418
ELM	0.8605	0.5283	0.1430	0.5286	0.9761	0.8983	0.9813	0.4269	0.0827	0.7821
EigenClass	0.8830	0.8436	0.8700	0.9266	0.8969	0.9692	0.9148	0.7056	0.6693	0.9274
GLVQ	0.8369	0.5417	0.8433	0.7607	0.9333	0.9159	0.8453	0.8296	0.5569	0.6782

Table 12. Overall performances and ranks of different classification approaches on multi-class benchmark classification problems

Algorithm	Acc		t_{exe}	
	Average	Rank	Average	Rank
DHT	**0.8863**	**1**	**2.7853**	**10**
MLOP	0.8182	11	0.3258	6
HP	0.7896	13	1.1938	8
SVM-L	0.7873	14	10.4753	13
SVM-G	0.4482	18	1.3368	9
DT	0.8636	3	0.0349	3
KNN	0.8466	6	0.0154	1
MLP	0.7634	16	0.4146	7
SC	0.8603	5	3.7449	11
SDKNN	0.8842	2	7.2275	12
ALMMo0	0.8168	12	0.1775	5
SOFIS	0.8373	10	0.1116	4
GA-SOFIS	0.8452	7	420.1389	16
PSO-SOFIS	0.8433	8.5	369.8214	15
GLPSO-SOFIS	0.8433	8.5	722.0387	17
ELM	0.6208	17	0.0311	2
EigenClass	0.8606	4	878.1962	18
GLVQ	0.7742	15	26.2725	14

It can be observed from Table 12 that the average Acc of DHT obtained over the 10 multi-class benchmark classification problems is 0.8863, ranked at the top place across all 18 approaches compared. This shows the very strong predictive performance of DHT for multi-class problems.

To examine the statistical significance of the better performance achieved by DHT, over the other 17 classification approaches and on the 10 multi-class benchmark problems, pairwise Wilcoxon signed rank tests [44] are conducted. The outcomes of the pairwise tests in terms of p-value are tabulated in Table 13, where the cascaded classification results by each approach across the 10 experiments are used. It can be observed that 87.65% of the p-values returned by the pairwise Wilcoxon tests are below the level of significance specified by $\alpha = 0.05$. This suggests that the performance of DHT is significantly better than the other 17 classifiers.

Table 13. p-values returned by pairwise Wilcoxon signed rank tests

DHT versus	Dataset									
	CA	GP	IS	LR	MF	PB	PR	SH	SPF	WF
MLOP	0.0000	0.0000	0.0000	0.0807	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
HP	0.0000	0.0000	0.8944	0.5376	0.1452	0.0000	0.0000	0.0000	0.0000	0.0000
SVM-L	0.0000	0.0000	0.0000	0.0219	0.0028	0.0000	0.0226	0.0000	0.0000	0.0000
SVM-G	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
DT	0.0135	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0009	0.4874	0.0000
KNN	0.0000	0.0000	0.0000	0.0000	0.7913	0.0000	0.0026	0.0000	0.0000	0.0510
MLP	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1423	0.0000	0.0000
SC	0.0000	0.0000	0.0078	0.0000	0.0000	0.0000	0.0003	0.8339	0.0000	0.2011
SDKNN	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0050
ALMMo0	0.0000	0.0000	0.0000	0.1763	0.2042	0.0000	0.0000	0.0000	0.0000	0.0028
SOFIS	0.0000	0.0000	0.0000	0.0820	0.1245	0.0000	0.0000	0.0000	0.0000	0.0000
GA-SOFIS	0.0000	0.0000	0.0000	0.1310	0.0635	0.0000	0.0000	0.0000	0.0000	0.0000
PSO-SOFIS	0.0000	0.0000	0.0000	0.0808	0.0015	0.0000	0.0000	0.0000	0.0000	0.0000
GLPSO-SOFIS	0.0000	0.0000	0.0000	0.4120	0.0883	0.0000	0.0000	0.0000	0.0000	0.0000
ELM	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
EigenClass	0.0000	0.0000	0.0000	0.2280	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
GLVQ	0.0000	0.0000	0.0001	0.0000	0.8242	0.0001	0.0000	0.0000	0.0000	0.0000

Remote Sensing Image Classification. Finally, experiments on popular real-world remote sensing image classification problems are conducted to further evaluate the performance of DHT. The results (in terms of Acc) on the these two problems are reported in Tables 14 and 15, respectively. For comparison, the results obtained by the relevant state-of-the-art approaches in the literature are given in these two tables also. It can be observed that DHT is able to achieve very high classification accuracy on the testing sets over both datasets, surpassing, or at least on par with the best performing models. This once again, demonstrates the strong performance of DHT.

Table 14. Performance comparison on OPTIMAL-31

Algorithm	Acc
DHT	**0.9989±0.0014**
GBNet [45]	0.9328±0.0027
MSNet [46]	0.9392±0.0041
ARCNet-VGG16 [47]	0.9270±0.0035
ARCNet-ResNet34 [47]	0.9128±0.0045
ARCNet-AlexNet [47]	0.8575±0.0035
Fine-tune VGGNet16 [47]	0.8745±0.0045
Fine-tune GoogLeNet [47]	0.8257±0.0012
Fine-tune AlexNet [47]	0.8122±0.0019
MAA-CNN [1]	0.9570±0.0054
EfficientNetB3-Basic [48]	0.9476±0.0026
EfficientNetB3-Attn-2 [48]	0.9586±0.0022

Table 15. Performance comparison on RSI-CB256

Algorithm	*Acc*	
	50 % labelled	80 % labelled
DHT	**0.9863±0.0011**	**0.9898±0.0015**
SIFT [49]	0.3796±0.0027	0.4012±0.0034
LBP [49]	0.6910±0.0020	0.7198±0.0036
CH [49]	0.8408±0.0026	0.8408±0.0026
Gist [49]	0.6174±0.0035	0.6359±0.0045
Enhanced Fusion of DCNNs [50]	-	0.9950

Four example images taken from the OPTIMAL-31 dataset are presented in Fig. 9, to provide a visual comparison between DHT and MAA-CNN [1]. Figs. 9(a) and (b) belong to the category of "commercial area". Figs. 9(c) and (d) belong to the two categories of "church" and "industrial area", respectively. As reported in [1], the four images are classified to the category of "commercial area" by MAA-CNN, two of which are misclassifications. Whilst DHT correctly classified all of them.

(a) Image of category "commercial area" (b) Image of category "commercial area"

(c) Image of category "church" (d) Image of category "industrial area"

Fig. 9. Visual comparison between DHT and MAA-CNN [1] on OPTIMAL-31 ((a) and (b) are correctly classified by both approaches; (c) and (d) are correctly classified by DHT but misclassified by MAA-CNN)

4.4. Further Evaluations

Further to the above systematic evaluations, additional experimental investigations are conducted on the basis of the 10 multi-class benchmark classification problems, under the same experimental protocols used previously.

Firstly, the impact of utilizing different distance measures on the performance of DHT is investigated. In this example, the most commonly used Euclidean distance metric is adopted as the alternative, and the performances of the resulting DHT model in terms of *Acc* are tabulated in Table 16. In addition, the same experiments are repeated using cosine dissimilarity as the distance measure, and the obtained results are also reported in Table 16. The outcomes obtained by DHT with the default Mahalanobis distance metric are given in the same table for easy

comparison. The average Acc of DHT while using each of the three distance measures over the 10 datasets is shown in Fig. 10.

Table 16. Performance of DHT with different distance measures on multi-class benchmark classification problems

Distance measure	Dataset									
	CA	GP	IS	LR	MF	PB	PR	SH	SPF	WF
Mahalanobis	0.8810	0.8771	0.8876	0.9221	0.9634	0.9450	0.9651	0.8700	0.7052	0.8454
Euclidean	0.8571	0.8727	0.8405	0.9284	0.9232	0.9418	0.9697	0.8766	0.3903	0.8493
Cosine	0.8481	0.8656	0.7957	0.9225	0.9300	0.9427	0.9640	0.8742	0.4151	0.8455

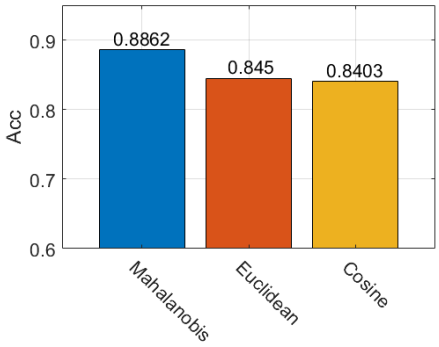


Fig. 10. Overall performances of DHT with different distance measures

It can be seen from Table 16 that the employment of different distance measures may affect the classification accuracy of the proposed DHT model to a certain degree. Using the default Mahalanobis distance metric, DHT produces better results on the largest number of datasets, including CA, GP, IS, MF, PB, and SPF. With the Euclidean distance metric, DHT performs the best on the LR, PR, SH and WF datasets, whilst being outperformed by the model utilizing cosine dissimilarity on the MF, PB and SPF datasets. In practice, however, without prior knowledge of the problem under consideration, it is generally impossible to prejudge which distance metric would enable DHT to achieve the best performance, but the use of Mahalanobis distance as the default is empirically supported as it provides the highest average Acc rate on the 10 datasets, as reflected by Fig. 10.

Experiments are also conducted to examine the behaviour of DHT with a predefined maximum model depth. During these experiments, the maximum depth of the DHT model varies from 5 to 20. The performances (Acc) of DHT with a different maximum model depth on the 10 multi-class benchmark datasets are reported in Table 17. The results by DHT with the default experimental setting are also presented in this table to facilitate comparison. Note that the system identification process of DHT self-terminates automatically before the maximum model depth is reached if the data samples of different classes have been appropriately separated at the current depth. In such cases, the models perform in exactly the same way as the one with the default setting, despite that a maximum model depth has been given.

Table 17. Performance of DHT with different maximum model depths on multi-class benchmark classification problems

G_{max}	Dataset									
	CA	GP	IS	LR	MF	PB	PR	SH	SPF	WF
5	0.8198	0.3082	0.6829	0.0691	0.8981	0.8975	0.6012	0.7829	0.5355	0.5529
10	0.8798	0.7670	0.8876	0.9077	0.9634	0.9487	0.9660	0.8700	0.7072	0.8379
15	0.8811	0.8757	0.8876	0.9221	0.9634	0.9449	0.9651	0.8700	0.7055	0.8456
20	0.8810	0.8771	0.8876	0.9221	0.9634	0.9450	0.9651	0.8700	0.7052	0.8454
Default	0.8810	0.8777	0.8876	0.9221	0.9634	0.9450	0.9651	0.8700	0.7052	0.8454

The results given in Table 17 confirm that without artificially restricting the maximum model depth, the system identification process of DHT is able to self-terminate in less than 20 recursive partitioning cycles in most cases. In general, the number of recursive partitioning cycles required by DHT is purely depending on the nature of data.

For example, the system identification process self-terminates in less than 10 cycles for the IS, MF and SH datasets, whilst it still carries on after 20 cycles for the GP dataset. Although users can choose to terminate the system identification process earlier via controlling the maximum model depth externally, the performance of DHT may be adversely impacted if the model depth is set to a too small number (e.g., less than 10).

In the final experimental study, the robustness of the proposed DHT is examined by adding Gaussian noise to the experimental data. During the experiments, 0dB additive white Gaussian noise is randomly added onto 10% of the data samples. The classification results (Acc) of DHT are reported in Table 18. The following nine classification algorithms are run for comparison (under the same experimental protocols), including: MLOP, HP, SVM-G, DT, KNN, SC, SDKNN, ALMMo0 and SOFIS. The obtained results by these comparative algorithms are tabulated in Table 18 as well. Furthermore, the same experiments are repeated by randomly selecting 20% of data samples to be added with the 0dB additive white Gaussian noise. The obtained results by the 10 classification algorithms are tabulated in the same table. The average Acc rates of the 10 algorithms over the 10 datasets across the experiments are shown in Fig. 11.

Table 18. Performances of different classification approaches on multi-class benchmark classification problems with different ratios of noisy samples

Algorithm	Dataset with 10% of noisy samples									
	CA	GP	IS	LR	MF	PB	PR	SH	SPF	WF
DHT	**0.8264**	**0.8046**	**0.8378**	**0.8358**	**0.8946**	**0.9302**	**0.8877**	**0.8612**	**0.4001**	**0.8032**
MLOP	0.8222	0.6175	0.7501	0.8255	0.8752	0.9096	0.8993	0.8483	0.3722	0.7846
HP	0.7764	0.6181	0.7414	0.8256	0.8972	0.8634	0.8899	0.8722	0.2270	0.7778
SVM-G	0.7804	0.7131	0.2205	0.3261	0.1023	0.9229	0.1040	0.0892	0.3507	0.6884
DT	0.8772	0.7502	0.8282	0.7387	0.8402	0.9447	0.8374	0.6464	0.6275	0.9277
KNN	0.8503	0.7444	0.8044	0.8446	0.9098	0.9381	0.9126	0.8684	0.4283	0.7954
SC	0.8501	0.7623	0.8142	0.7814	0.9054	0.9338	0.8930	0.8800	0.3559	0.8522
SDKNN	0.8525	0.6845	0.7544	0.7645	0.9175	0.9242	0.9004	0.8600	0.1928	0.8143
ALMMo0	0.8221	0.6443	0.7089	0.8273	0.9027	0.9335	0.9028	0.8850	0.3469	0.7884
SOFIS	0.8392	0.6849	0.7563	0.8222	0.8794	0.9240	0.9011	0.8844	0.4024	0.7860
Algorithm	Dataset with 20% of noisy samples									
	CA	GP	IS	LR	MF	PB	PR	SH	SPF	WF
DHT	**0.8042**	**0.7310**	**0.7777**	**0.7483**	**0.8586**	**0.9119**	**0.8167**	**0.8468**	**0.3818**	**0.7541**
MLOP	0.7859	0.5466	0.7074	0.7269	0.8219	0.9017	0.8261	0.8355	0.3659	0.7322
HP	0.7373	0.5617	0.6859	0.7365	0.8741	0.8391	0.8158	0.8582	0.2325	0.7277
SVM-G	0.7805	0.6629	0.2105	0.2765	0.1016	0.9168	0.1040	0.0891	0.3507	0.6502
DT	0.8439	0.6831	0.7770	0.6562	0.7694	0.9302	0.7658	0.6048	0.5898	0.8655
KNN	0.8268	0.6825	0.7628	0.7548	0.8898	0.9305	0.8460	0.8608	0.4225	0.7561
SC	0.8303	0.6980	0.7513	0.6977	0.8615	0.9176	0.8240	0.8703	0.3347	0.8027
SDKNN	0.8409	0.6325	0.6866	0.6830	0.8788	0.9178	0.8366	0.8481	0.1741	0.7739
ALMMo0	0.7976	0.5898	0.6659	0.7361	0.8776	0.9146	0.8300	0.8691	0.3303	0.7433
SOFIS	0.8033	0.6167	0.7136	0.7222	0.8456	0.9121	0.8287	0.8658	0.4137	0.7322

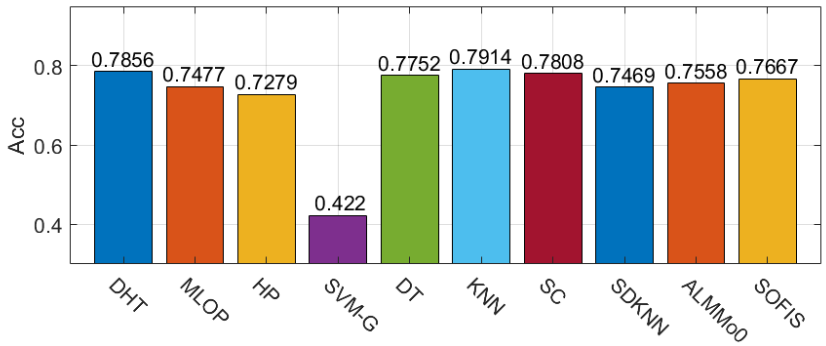


Fig. 11. Overall performances of different classification approaches on multi-class benchmark classification problems with 0dB additive white Gaussian noise

Collectively, from the results of Table 18 and Fig. 11, it can be seen that the overall *Acc* of DHT is the second highest among the 10 classification methods (only outperformed by KNN). The proposed approach is therefore verified to offer a generally strong resistance to Gaussian noise.

4.5. Discussions

In short, all experimental studies carried out so far collectively demonstrate the significant potential of DHT as a powerful nonparametric method for classification. It offers the highest overall predictive precision on multi-class classification problems, and the second best on binary ones. In addition, its performance on remote sensing datasets is also top ranked, showing the capability of DHT on solving high-dimensional, complex problems. Experimental studies also demonstrate that DHT is robust to Gaussian noise.

The computational efficiency of DHT is however, basically at the same level as techniques such as HP, MLOP, SVM and SC (as shown in Section 4.3). Fortunately, this limitation does not form a major concern in practice since it is devised as an offline classification approach. Nevertheless, for offline learning, DHT requires a sufficient amount of training samples to be available. This may significantly limit its applicability to very large-scale, high-dimensional problems, mostly due to hardware limitation.

In addition, the prototypes within the hierarchical structure produced by DHT are identified from data through a recursive data partitioning process. Such prototypes may not be optimal because no iterative optimization is carried out during the system identification process. It has been shown above that evolutionary algorithms can effectively help to improve the classification accuracy of prototype-based classifiers, and it is reasonable to presume that the classification performance of DHT can be further improved once the optimality of the prototypes is attained. However, the prototype optimization processes by evolutionary algorithms are time consuming and can cost much more computational resources. Hence, a more efficient optimization scheme would be needed for DHT to improve its classification performance without significantly increasing the computation burden. Otherwise, a trade-off between classification performance and training cost has to be considered.

Finally, the default implementation of DHT is to work with Mahalanobis distance, but different types of distance measure, e.g., Euclidean distance, cosine dissimilarity, are also supported by this approach. Numerical examples have shown that the predictive precision of DHT on a particular problem may vary significantly with different types of distance measure being used. Currently, it is difficult for DHT to self-determine which metric to be utilized as the best option, but the empirical results achieved so far have indicated that it is appropriate to use the Mahalanobis distance metric as the default without the need of human intervention in the setup of the DHT model.

5. Conclusions

This paper has presented a nonparametric approach to self-constructing a prototype-based classification model, named DHT, which does not require any externally controlled parameters to be predefined a priori and is entirely data driven. By recursively partitioning the empirically observed data at multiple levels of granularity in a divisive manner, DHT achieves a multi-granular partition of data and autonomously self-constructs a multi-layered hierarchical structure from the identified prototypes for classification. Experimental case studies on a wide range of benchmark binary- and multi-class problems, including those involving real-world remote sensing image data, show the high-level predictive performance of DHT, justifying the effectiveness and validity of the proposed novel approach.

There are several considerations for future work. Firstly, it can be highly rewarding to design an online learning extension for the proposed approach, such that it can handle streaming data, possibly on a chunk-by-chunk basis. Secondly, the optimality of prototypes within the hierarchical structure of DHT needs to be further investigated as they play an instrumental role in the structure of the resulting classification model. It would be helpful to introduce a novel optimization mechanism with greater computational efficiency to attain local optimality of the identified prototypes. Thirdly, an encoding mechanism can be introduced to add semantics to data. This can effectively help DHT to handle data with categorical attributes. Finally, it would be interesting to develop a meta-control scheme that would enable DHT to self-determine an appropriate distance metric for use in a generic practical problem-solving setting.

References

[1] F. Li, R. Feng, W. Han, and L. Wang, “An augmentation attention mechanism for high-spatial-resolution remote sensing image scene classification,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp.

3862–3878, 2020.

[2] M. Zhu *et al.*, “Class weights random forest algorithm for processing class imbalanced medical data,” *IEEE Access*, vol. 6, pp. 4641–4652, 2018.

[3] H. Hagras, “Toward human-understandable, explainable AI,” *Computer (Long. Beach. Calif).*, vol. 51, no. 9, pp. 28–36, 2018.

[4] 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.

[5] Y. Pan, L. Zhang, X. Wu, and M. J. Skibniewski, “Multi-classifier information fusion in risk analysis,” *Inf. Fusion*, vol. 60, no. December 2019, pp. 121–136, 2020.

[6] Y. Li, H. Zhang, X. Xue, Y. Jiang, and Q. Shen, “Deep learning for remote sensing image classification: a survey,” *WIREs Data Min. Knowl. Discov.*, vol. 8, no. 6, p. e1264, 2018.

[7] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *Nat. Mach. Intell.*, vol. 1, no. 5, pp. 206–215, 2019.

[8] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” in *International Conference on Learning Representations*, 2017, pp. 1–17.

[9] L. Breiman, “Random forests,” *Mach. Learn. Proc.*, vol. 45, no. 1, pp. 5–32, 2001.

[10] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge: Cambridge University Press, 2000.

[11] T. Kohonen, “The self-organizing map,” *Neurocomputing*, vol. 21, no. 1–3, pp. 1–6, 1998.

[12] J. R. Quinlan, “Induction of decision trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[13] P. Cunningham and S. J. Delany, “K-nearest neighbour classifiers,” *Mult. Classif. Syst.*, vol. 34, pp. 1–17, 2007.

[14] 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.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Burlin: Springer, 2009.

[16] X. Gu and P. P. Angelov, “Self-organising fuzzy logic classifier,” *Inf. Sci. (Ny).*, vol. 447, pp. 36–51, 2018.

[17] H. Rong, Z. Yang, and P. K. Wong, “Robust and noise-insensitive recursive maximum correntropy-based evolving fuzzy system,” *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 9, pp. 2277–2284, 2019.

[18] H. Huang, H.-J. Rong, Z.-X. Yang, and C.-M. Vong, “Jointly evolving and compressing fuzzy system for feature reduction and classification,” *Inf. Sci. (Ny).*, vol. 579, pp. 218–230, 2021.

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

[20] R. N. Patro, S. Subudhi, P. K. Biswal, and F. Dell’Acqua, “Dictionary-based classifiers for exploiting feature sequence information and their application to hyperspectral remotely sensed data,” *Int. J. Remote Sens.*, vol. 40, no. 13, pp. 4996–5024, 2019.

[21] U. Erkan, “A precise and stable machine learning algorithm: eigenvalue classification (EigenClass),” *Neural Comput. Appl.*, vol. 33, no. 10, pp. 5381–5392, 2021.

[22] D. Chen, Q. Yang, J. Liu, and Z. Zeng, “Selective prototype-based learning on concept-drifting data streams,” *Inf. Sci. (Ny).*, vol. 516, pp. 20–32, 2020.

[23] G. Cerruela-García, A. de Haro-García, J. P. P. Toledano, and N. García-Pedrajas, “Improving the combination of results in the ensembles of prototype selectors,” *Neural Networks*, vol. 118, pp. 175–191, 2019.

[24] M. P. Sesmero, J. A. Iglesias, E. Magán, A. Ledezma, and A. Sanchis, "Impact of the learners diversity and combination method on the generation of heterogeneous classifier ensembles," *Appl. Soft Comput.*, vol. 111, p. 107689, 2021.

[25] R. Diao, F. Chao, T. Peng, N. Snooke, and Q. Shen, "Feature selection inspired classifier ensemble reduction," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1259–1268, 2014.

[26] J. Shao, F. Huang, Q. Yang, and G. Luo, "Robust prototype-based learning on data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 978–991, 2018.

[27] X. Gu and W. Ding, "A hierarchical prototype-based approach for classification," *Inf. Sci. (Ny).*, vol. 505, pp. 325–351, 2019.

[28] X. Gu and M. Li, "A multi-granularity locally optimal prototype-based approach for classification," *Inf. Sci. (Ny).*, vol. 569, pp. 157–183, 2021.

[29] T. M. Kodinariya and P. R. Makwana, "Review on determining number of cluster in K-means clustering," *Int. J. Adv. Res. Comput. Sci. Manag. Stud.*, vol. 1, no. 6, pp. 2321–7782, 2013.

[30] C. Ding and X. He, "Cluster merging and splitting in hierarchical clustering algorithms," in *IEEE International Conference on Data Mining*, 2002, pp. 139–146.

[31] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams*, 2nd ed. Chichester, England: John Wiley & Sons., 1999.

[32] X. Gu, Q. Ni, and G. Tang, "A novel data-driven approach to autonomous fuzzy clustering," *IEEE Trans. Fuzzy Syst.*, DOI: 10.1109/TFUZZ.2021.3074299, 2021.

[33] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *Proceedings - International Conference on Pattern Recognition*, 2010, pp. 3121–3124.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[35] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[37] X. Gu and P. Angelov, "A multi-stream deep rule-based ensemble system for aerial image scene classification," in *Handbook on Computer Learning and Intelligence*, 2021.

[38] G. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 42, no. 2, pp. 513–529, 2012.

[39] A. Sato and K. Yamada, "Generalized learning vector quantization," in *Advances in neural information processing systems*, 1996, pp. 423–429.

[40] M. Mitchell, *An introduction to genetic algorithms*. Cambridge, MA: MIT Press, 1996.

[41] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[42] Y. J. Gong *et al.*, "Genetic learning particle swarm optimization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, 2016.

[43] X. Gu, Q. Shen, and P. Angelov, "Particle swarm optimized autonomous learning fuzzy system," *IEEE Trans. Cybern.*, vol. 51, no. 11, pp. 5352–5363, 2021.

[44] F. Wilcoxon, "Individual comparisons of grouped data by ranking methods," *J. Econ. Entomol.*, vol. 39, no. 6, pp. 269–270, 1946.

[45] H. Sun, S. Li, X. Zheng, and X. Lu, "Remote sensing scene classification by gated bidirectional network," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 1, pp. 82–96, 2020.

[46] N. Liu, T. Celik, and H. Li, "MSNet: a multiple supervision network for remote sensing scene classification," *IEEE Geosci. Remote Sens. Lett.*, DOI: 10.1109/LGRS.2020.3043020, 2020.

[47] Q. Wang, S. Liu, and J. Chanussot, "Scene classification with recurrent attention of VHR remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 1155–1167, 2019.

[48] H. Alhichri, A. S. Alswayed, Y. Bazi, N. Ammour, and N. A. Alajlan, "Classification of remote sensing images using EfficientNet-B3 CNN model with attention," *IEEE Access*, vol. 9, pp. 14078–14094, 2021.

[49] H. Li *et al.*, "RSI-CB: a large-scale remote sensing image classification benchmark using crowdsourced data," *Sensors*, vol. 20, no. 6, pp. 28–32, 2020.

[50] G. J. Scott, K. C. Hagan, R. A. Marcum, J. A. Hurt, D. T. Anderson, and C. H. Davis, "Enhanced fusion of deep neural networks for classification of benchmark high-resolution image data sets," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 9, pp. 1451–1455, 2018.