Self-Organising Transparent Learning System

Xiaowei Gu

A thesis presented for the degree of

Doctor of Philosophy

School of Computing and Communications

Lancaster University, England

Supervisor: Prof. Plamen P. Angelov, Ph.D., D.Sc., FIEEE, FIET

September 2018

Abstract

Machine learning, as a subarea of artificial intelligence, is widely believed to reshape the human world in the coming decades. This thesis is focused on both the unsupervised and supervised self-organising transparent machine learning techniques. One particularly interesting aspect is the transparent self-organising deep learning systems.

Traditional data analysis approaches and most of the machine learning algorithms are built upon the basis of probability theory and statistics. The solid mathematical foundation of the probability theory and statistics guarantees the good properties of these learning algorithms when the amount of data tends to infinity and all the data comes from the same distribution. However, the prior assumptions of the random nature and same distribution imposed on the data generation model are often too strong and impractical in real applications. Moreover, traditional machine learning algorithms also require a number of free parameters to be predefined. However, without any prior knowledge of the problem, which is often the case in real situations, the performance of the algorithms can be largely influenced by the improper choice.

Deep learning-based approaches are currently the state-of-the-art techniques in the fields of machine learning and computer vision. However, they are also suffering from a number of deficiencies including the computational burden of training using huge amount of data, lack of transparency and interpretation, ad hoc decisions about the internal structure, no proven convergence for the adaptive versions that rely on reinforcement learning, limited parallelisation and offline training, etc. These shortcomings largely all hinder the wider applications of the deep learning in real situations.

The novel approaches presented in this thesis are developed within the Empirical Data Analytics framework, which is an alternative, but more advanced computational methodology to the traditional approaches based on the ensemble properties and mutual distribution of the empirical discrete observations.

The novel self-organising transparent machine learning algorithms presented in this work for clustering, regression, classification and anomaly detection are autonomous, self-organising, data-driven and free from user- and problem- specific parameters. They do not impose any data generation models on the data a priori, but are driven by the empirically observed data and are able to produce the objective results without prior knowledge of the problems. In addition, they are highly efficient and suitable for large-scale static/streaming data processing.

The newly proposed self-organising transparent deep learning systems are able to achieve human-level performance comparable to or even better than the deep convolutional neural networks on image classification problems with the merits of being fully transparent, self-evolving, highly efficient, parallelisable and human-interpretable. More importantly, the proposed deep learning systems have the ability of starting classification from the very first image of each class in the same way as humans do.

Numerical examples based on numerous challenging benchmark problems and comparisons conducted with the state-of-the-art approaches presented in this thesis demonstrated the validity and effectiveness of the proposed new machine learning algorithms and deep learning systems and show their potential for real applications.

Statement of Originality

I, Xiaowei Gu, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in this thesis.

Acknowledgements

Firstly, my utmost gratitude goes to my parents for their unconditional supports throughout the years. Without them, it will not be possible for me to pursue the Ph.D. degree in UK. I am deeply grateful to my supervisor, Professor Plamen Angelov, for all the very kind guidance and assistances he provided. The huge amount of time and efforts he spent on supervising me and the invaluable knowledge and experience he shared with me play the key role to my research advance. I am pleased to thank Professor Jose Principe and Dr. Dmitry Kangin for all the discussions and help. Many thanks go to the supervisor in my Master period in China, Professor Zhijin Zhao for the knowledge, sets of thinking and research skills she unreservedly passed to me. I also need to express my appreciation to Dr. Shen, Prof. Wang and Mr. Xu for their very kind help and supports.

Table of Contents

List of Figures

List of Tables

Abbreviations

AAD – Autonomous Anomaly Detection (Algorithm)

ADD – Autonomous Data-Driven (Clustering Algorithm)

ADP – Autonomous Data Partitioning (Clustering Algorithm)

ALMMO – Autonomous Learning Multi-Model (System)

ANFIS – Adaptive-Network-Based Fuzzy Inference System

ANN – Artificial Neural Network

APC – Affinity Propagation Clustering

ASR – Adaptive Sparse Representation

BOVW – Bag of Visual Words

CBDNET – Convolutional Deep Belief Network

CDF – Cumulative Distribution Function

CEDS – Clustering of Evolving Data Streams

CLFH – Learning Convolutional Feature Hierarchies

CNN – Convolutional Neural Network

CSAE – Convolutional Sparse Autoencoders

DBSCAN – Density-Based Spatial Clustering of Applications with Noise

DCNN – Deep Convolutional Neural Network

DDCAR – Data Density Based Clustering with Automated Radii

DECNNET – Deconvolutional Networks

DENFIS – Dynamic Evolving Neural-Fuzzy Inference System

DLNN – Deep Learning Neural Network

DPC – Density Peaks Clustering (Algorithm)

DRB – Deep Rule-Based (System)

DT – Decision Tree

EC – Evolutionary Computation

ECM – Evolving Clustering Method

EDA – Empirical Data Analytics

EFUNN – Evolving Fuzzy Neural Networks

ELMC – Evolving Local Means Clustering (Algorithm)

ETS – Evolving Takagi-Sugeno (Fuzzy System)

FCM – Fuzzy C-Means (Algorithm)

FCMMS – Fuzzily Connected Multimodal Systems

FFNDL – Fast Feedforward Nonparametric Deep Learning

FRB – Fuzzy Rule-Based (System)

FWRLS – Fuzzily Weighted Recursive Least Squares

GGMC – Greedy Gradient Max-Cut (Algorithm)

HCDP – Hypercube-Based Data Partition (Algorithm)

HOG – Histogram of Oriented Gradients

ID3 – Iterative Dichotomiser 3

IID – Independent and Identically Distributed

KDE – Kernel Density Estimation

KNN – K-Nearest Neighbours

LAPSVM – Laplacian Support Vector Machine

LCLC – Local-Constraint Linear Coding

LGC – Local and Global Consistence

LSLR – Least Square Linear Regression

LSPM – Linear Spatial Pyramid Matching

MNSIT – Modified National Institute of Standards and Technology

MSC – Mean-Shift Clustering

NN – Neural Network

MUFL – Multipath Unsupervised Feature Learning

NMIBC – Nonparametric Mode Identification Based Clustering

NMMBC – Nonparametric Mixture Model Based Clustering

PDF – Probability Density Function

PMF – Probability Mass Function

QQSRM – QuantQuote Second Resolution Market

RBF – Radical Basis Function

RCNET – Random Convolutional Network

RDE – Recursive Density Estimation

RLSE – Recursive Linear Least-Square Estimator

RS – Random Swap (Algorithm)

RSN – Regularized Shearlet Network

SAFIS – Sequential Adaptive Fuzzy Inference System

SDAL$_{21}$M	Sparse Discriminant Analysis via Joint L$_{2,1}$-norm Minimization
SFC	Sparse Fingerprint Classification
SIFTSC	Scale-Invariant Feature Transform with Sparse Coding
SODA	Self-Organised Direction Aware
SOFL	Self-Organising Fuzzy Logic
SOM	Self-Organising Map
SPMK	Spatial Pyramid Matching Kernel
SSDRB	Semi-Supervised Deep Rule-Based
S&P	Standard and Poor
SUBC	Subtractive Clustering
SVM	Support Vector Machine
SWLSLR	Sliding Window Least Square Linear Regression
TEDA	Typicality-and Eccentricity-Based Data Analytics
TLFP	Two-Level Feature Representation
TSVM	Transductive Support Vector Machine

1. Research Overview

This chapter presents the research motivation and summary of the research contributions, publications and the research methodology. The chapter is organised as follows. Section 1.1 gives the research motivation. The research contributions are described in section 1.2. The methodology and publication summary are given in section 1.3 and section 1.4, respectively. This chapter is finished by the thesis outline.

1.1. Motivation

Nowadays, due to the more matured electronic manufacturing and information technologies as well as the widely distributed sensors networks, astronomical amount of streaming data is generated from every area of daily activities. As the world has already entered the Era of Big Data, data-intensive technologies are now being extensively used by the developed economies and numerous international organisations. Having realised the underlying economic benefits in these data, an increasing number of companies, corporations and research institutions are involved in developing more advanced data analytic and processing technologies.

Traditional data analytic methodologies [1]–[4] heavily rely on the classical probability theory and statistics. The appeals of the traditional data analytic methodologies come from their solid mathematical foundations and their ability that is always guaranteed when the amount of the data tends to infinity and all the data comes from the same distribution, as stated by the classical probability theory. Indeed, the traditional probability theory and statistics [1]–[4] assume the actual data to be realisations of imaginary random variables and further assume the prior distributions of these variables. However, these appeals also clearly demonstrate the problems/deficiencies of the traditional methodologies:

1) It is impossible to collect or process the infinity amount of observations;

2) The very strong prior assumptions are often impractical in the real cases;

3) The distribution of the data, or the generation model, is not clear in advance.

These problems/deficiencies more often lead the traditional data analytic approaches to generate the subjective results, which undermine the effectiveness and correctness of the traditional data analytic approaches

Heavily relying on the probability theory and statistics, traditional machine learning technologies, i.e. clustering, classification, prediction, fault detection, etc., often need users to predefine various kinds of parameters and prior assumptions in order to guarantee an effective result [5]–[18]. These predefined parameters and assumptions usually require users to have a certain extent of prior knowledge and expertise. However, the prior knowledge is more often unavailable in real cases as the purpose of data analytics is to analyse and understand the unknown data, not to study the well-understood ones. It is also practically impossible to empirically predefine parameters for complex problems.

Moreover, most of the existing data processing technologies [5]–[18] were mainly built upon the basis of the traditional data analytic methodologies [1]–[4]. One cannot expect that these approaches can get rid of the deficiencies that the traditional probability theory and statistics suffer from. These data processing technologies often simplify the real data representation and assume the data following a specific distribution, i.e. the most widely used Gaussian. The actual data considered in the machine learning literature is usually discrete (or discretized), which in traditional probability theory and statistics are modelled as a realisation of the random variable, but one does not know a priori their distribution. If the prior data generation hypothesis is verified, good results can be expected; otherwise, this opens the door for many failures.

Besides, many well-known algorithms [5], [6], [13]–[16] as well as some recently published ones [10] are restricted to offline data processing. Many algorithms also lack the ability of following the ever-changing data pattern in streaming data. They require a full retraining when new data patterns emerge.

As the one of so-called latest developments in the fields of machine learning and artificial intelligence, deep learning [19] is a hot research area attracting the attention of machine learning researchers as well as the public. Relying on extracting high-level abstractions in data by using a multiple layer structure composed of linear and non-linear transformations, the published methods have presented very promising results in image processing [20]–[24]. Nonetheless, there are three major deficiencies in the current deep learning methods:

1) The features extracted and the steps to get them by the encoder-decoder methods have low-level of human interpretability (are opaque) [19]–[22];

2) The training process is off-line and requires a large amount of time as well as complex computational resources [22]–[24];

3) There are too many ad hoc decisions in terms of structures and parameters [20]–[24].

These deficiencies largely hinder the applications of the deep learning networks in real problems.

Aiming at overcoming the deficiencies deeply rooted in the traditional probability theory and statistics, Empirical Data Analytics (EDA) framework is a systematic methodology of nonparametric quantities recently introduced in [25]–[27] based on the ensemble properties and mutual distribution of the empirical discrete observations. It touches the very foundation of data analytics and serves as a strong alternative to the traditional statistics and probability theory, but is free from the paradoxes and problems that the traditional approaches are suffering from [26], [27].

The focus of this thesis is the novel machine learning algorithms and deep learning systems developed within the EDA framework. Compared with traditional ones, these new approaches presented in this thesis have the following distinctive features:

1) They are self-organising and self-evolving;

2) They are free from prior assumptions and user- and problem- specific parameters;

3) Their structure and operating mechanism are transparent and human interpretable.

These properties of the new approaches presented in this thesis make them appealing alternatives to both traditional and state-of-the-art methods.

1.2. Research Contribution

This research work focuses on the novel self-organising transparent learning systems. During the research, the following main contributions have been achieved:

1) Four novel unsupervised machine learning approaches have been developed for clustering and data partitioning, and they are evaluated on benchmark datasets;

2) Four novel supervised machine learning approaches have been developed for classification, regressions and anomaly detection problems, and they are evaluated on benchmark datasets and real-world high frequency trading problems;

3) Two new types of deep learning networks have been proposed for image classification problems, and they have been applied to various challenging benchmark datasets from different areas.

1.3. Methodology

This research work is focused on new machine learning algorithms and systems, which consists of the following parts:

1) Theoretical concepts research;

2) Algorithm implementation;

3) Application and validation.

Based on the theoretical concepts research, the mathematical and analytical description of the proposed approaches are formulated and investigated, which directly gives an evidence of the validity and effectiveness of the approaches as well as a basic understanding of their boundaries and limitations.

Then, the algorithm implementation is to show the practical feasibility of the theoretical concepts as well as to augment the theoretical analysis.

For the last part, the implemented theoretical concept is tested on benchmark problems for evaluating its applicability and validity, and it also gives an evidence of the effectiveness of the algorithms in real situations.

1.4. Publication Summary

The research work presented in this thesis was described in the following publications in the chronological order by the submission dates:

A. Journal Papers

P. Angelov, X. Gu, D. Kangin, Empirical data analytics, *International Journal of Intelligent Systems*, vol. 32(12), pp. 1261-1284, 2017.

P. Angelov, X. Gu, J. Principe, A generalized methodology for data analysis, *IEEE Transactions on Cybernetics*, vol. 48(10), pp. 2981 - 2993, 2018.

P. Angelov, X. Gu, J. Principe, Autonomous learning multi-model systems from data streams, *IEEE Transactions on Fuzzy Systems*, vol. 26(4), pp. 2213-2224, 2018.

X. Gu, P. Angelov, D. Kangin, J. Principe, A new type of distance metric and its use for clustering, *Evolving Systems*, vol. 8 (3), pp.167-177, 2017.

X. Gu, P. Angelov, D. Kangin, J. Principe, Self-organised direction aware data partitioning algorithm, *Information Sciences*, vol. 423, pp. 80-95, 2017.

P. Angelov, X. Gu, Empirical Fuzzy Sets, *International Journal of Intelligent Systems*, vol.33(2), pp. 362-395, 2018.

X. Gu, P. Angelov, C. Zhang, P. Atkinson, A massively parallel deep rule-based ensemble classifier for remote sensing scenes, *IEEE Geoscience and Remote Sensing Letters*, vol.15(3), pp. 345-349, 2018.

X. Gu, P. Angelov, J. Principe, A method for Autonomous data partitioning, *Information Sciences*, vol. 460–461, pp. 65-82, 2018.

P. Angelov, X. Gu, Deep rule-based classifier with human-level performance and characteristics, *Information Sciences*, vol. 463-464, pp. 196-213, 2018.

X. Gu, P. Angelov, Semi-supervised deep rule-based approach for image classification, *Applied Soft Computing*, vol. 68, pp. 53-68, 2018.

X. Gu, P. Angelov, Self-organising fuzzy logic classifier, *Information Sciences*, vol. 447, pp. 36-51, 2018

B. Conference Papers

X. Gu, P. Angelov, A. Ali, W. Gruver, G. Gaydadjiev, Online evolving fuzzy rule-based prediction model for high frequency trading financial data stream, in *IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS),* Natal, Brazil, 2016, pp.169 - 175.

P. Angelov, X. Gu, G. Gutierrez, J. Iglesias, A. Sanchis, Autonomous data density based clustering method, in *International Joint Conference on Neural Networks (IJCNN)* , Vancouver Canada, 2016, pp.2405-2413.

P. Angelov, X. Gu, D. Kangin, J. Principe, Empirical data analysis: a new tool for data analytics, in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Budapest, Hungary 2016, pp. 000052 - 000059.

X Gu, P. Angelov, Autonomous data-driven clustering for live data stream, in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Budapest, Hungary, 2016, pp. 001128 - 001135.

X. Gu, P. Angelov, G. Gutierrez, J. Iglesias, A. Sanchis, Parallel computing TEDA for high frequency streaming data clustering, in *INNS Conference on Big Data*, Thessaloniki, Greece, 2016, pp.238-253.

P. Angelov, X. Gu, Local modes-based free-shape data partitioning, in *IEEE Symposium Series on Computational Intelligence (SSCI),* Athens, Greece, 2016 pp.1-8.

P. Angelov, X. Gu, J. Principe, Fast feedforward non-parametric deep learning network with automatic feature extraction, in *International Joint Conference on Neural Networks (IJCNN),* Anchorage, Alaska, USA, 2017, pp. 534-541.

Angelov, X. Gu, Autonomous learning multi-model classifier of 0-order (ALMMo-0), in *IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS),* Ljubljana, Slovenia, 2017, pp. 1-7.

X. Gu, P. Angelov, Autonomous anomaly detection, in *IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS),* Ljubljana, Slovenia, 2017, pp. 1-8.

P. Angelov, X. Gu, MICE: Multi-layer multi-model images classifier ensemble, in *IEEE International Conference on Cybernetics (CYBCONF*), Exeter, UK, 2017, pp. 1-8.

P. Angelov, X. Gu, A Cascade of deep learning fuzzy rule-based image classifier and SVM, in *IEEE International Conference on Systems, Man, and Cybernetics (SMC2017),* Banff, Canada, 2017, pp. 746-751.

1.5. Thesis Outline

The remainder of the thesis is organised as follows.

Chapter 2 - Research Background and Theoretical Basis: contains three parts, the data analysis methodologies survey, computational intelligence methodologies survey and machine learning techniques survey. The review serves as the research background and the theoretical basis of the research works presented in the thesis.

Chapter 3 - Self-Organising Unsupervised Machine Learning Algorithms: proposes four different unsupervised machine learning algorithms for clustering and data partitioning, 1) autonomous data-driven clustering algorithm [28]–[30]; 2) hypercube-based data partitioning algorithm; 3) autonomous data partitioning algorithm [31] and 4) self-organising direction-aware data partitioning algorithm [32], [33]. These approaches are developed within the EDA computational framework, and thus, are nonparametric, self-organising and entirely data-driven.

Chapter 4 - Self-Organising Supervised Machine Learning Algorithms: proposes a first-order autonomous learning multi-model system for regression and classification [34], a zero-order autonomous learning multi-model classifier [35], a self-organising fuzzy logic classifier [36] and an autonomous anomaly detection algorithm [37]. These approaches are also developed within the EDA framework, therefore, they are free from problem- and user-specific parameters and prior assumptions.

Chapter 5 - Transparent Deep Learning Systems: proposes a fast feedforward nonparametric deep learning network [38] and deep rule-based systems [39] for image classification. The semi-supervised, active learning mechanism of the deep rule-based system is presented [40]. Some successful examples of deep rule-based ensemble classifiers are also given [41]–[43]. Compared with other deep learning approaches, the deep learning systems developed within the EDA framework are transparent, nonparametric, feedforward, human interpretable and free from ad hoc decisions.

Chapter 6 - Implementation and Validation of the Developed Algorithms: presents numerical examples based on benchmark problems for validating the algorithms presented in this thesis. A number of state-of-the-art approaches are involved for comparison for a better evaluation [28]–[43].

Chapter 7 - Conclusion and Future Work: summarises this thesis and gives the directions for further work.

2. Research Background and Theoretical Basis

In this chapter, a review of data analysis methodologies, computational intelligence methodologies and machine learning techniques is presented serving as the research background and the theoretical basis of this thesis.

2.1. Data Analysis Methodologies Survey

Data analysis can be described as a process of describing, illustrating and evaluating data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making. Besides the engineering, natural sciences and economics, nowadays, other scientific areas, i.e. biomedical, social science, etc., are also becoming data-centred.

In this section, the traditional data analytics approach (probability theory and statistics) and the more recently introduced data-centred ones are reviewed.

2.1.1. Probability Theory and Statistics

A key concept in the field of pattern recognition is “uncertainty” [2], [3]. Uncertainty exists in our daily lives as well as in every discipline in science, engineering, and technology. Many actions have consequences that are unpredictable in advance just like tossing a coin or throwing a dice, both of which are simple daily examples. Some of the more complex examples can be, for example, stock prices changes, foreign currency exchange rates. Probability theory is about such actions and their consequences. It starts with the idea of an experiment, being a course of action whose consequence is not predetermined and this experiment is reformulated as a mathematical object called a probability space [44]. Given any experiment involving chance, there is a corresponding probability space, and the study of such spaces is called probability theory [44].

Probability theory provides a consistent framework for the quantification and manipulation of uncertainties and forms one of the central foundations for pattern recognition and data analysis [2], [3]. Probability theory serves as the mathematical foundation for statistics [3] and is essential to many human activities that involve quantitative analysis of data. The core of the statistical approaches is the definition of a random variable, i.e. a functional measure from the space of events to the real line, which defines the probability theory [1]–[4]. Methods of probability theory also apply to study the average behaviour of a mechanical system, where the state of the system is uncertain, as in the field of statistical mechanics [45].

2.1.1.1. Discrete Probability Distribution

Initially, the probability theory considers only the discrete random variables, where the concept of "discrete" means that the random variables take only finite or countably finite values in the data space. A probability mass function (PMF) is a function that describes the probability that a discrete random variable is exactly equal to some value. The PMF is the primary means of defining a discrete probability distribution, and PMFs exist for random variables including the multivariate ones in the discrete domains. The formal definition of a PMF is as [44]:

For a random variable x with the value range $\{x\} = \{x_1, x_2, x_3, \ldots\}$ (finite or countable infinite), the function,

$$P_x(x_k) = \mathrm{P}(x = x_k) \text{ for } k = 1,2,3, \ldots, \tag{2.1}$$

is called the PMF of x, where the subscript x indicates that this is the PMF of the random variable, x. As one can see from equation (2.1), PMF is a function that describes the probabilities of the possible values for a random variable and the PMF is defined within a certain range. In general, there is:

$$P_x(x) = \begin{cases} \mathrm{P}(x) & x \in \{x\} \\ 0 & otherwise \end{cases}, \tag{2.2}$$

and PMFs have the following properties [44]:

$$0 \le P_x(x) \le 1; \tag{2.3}$$

$$\textstyle\sum_{x\in\{x\}} P_x(x) = \sum_{x\in\{x\}} \mathrm{P}(x) = 1; \tag{2.4}$$

$$\text{For } \{x\}_o \subseteq \{x\},\ P_x(x \in \{x\}_o) = \textstyle\sum_{x\in\{x\}_o} \mathrm{P}(x). \tag{2.5}$$

The cumulative distribution function (CDF) of the random variable x, evaluated at x_o, is defined as:

$$F_x(x_o) = \textstyle\sum_{y\in\{x\}\wedge y\le x_o} \mathrm{P}(y). \tag{2.6}$$

From equation (2.6) one can see that for the discrete random variable x, the corresponding CDF increases only at the points where it "jumps" to a higher value, and is constant between these jumps. The points where jumps occur are precisely the values that the random variable x may take. Therefore, the CDF of a discrete random variable is a discontinuous function.

2.1.1.2. Continuous Probability Distribution

Modern probability theory also considers the continuous random variables. A probability density function (PDF) of a continuous random variable, is a function, whose value at any given point in its value range can be interpreted as providing a relative likelihood that the value of the random variable would be equal to that sample, meanwhile, the absolute likelihood for a continuous random variable to take on any particular value is 0 [44].

In fact, the PDF is used to specify the probability of the random variable falling within a particular range of values, as opposed to taking on any single value. This probability is given by the integral of this variable's PDF over that range.

For a continuous random variable, the probability for it to fall in to the value range of $[x_1, x_2]$ is calculated as [44]:

$$P_x(x_1 < x < x_2) = \int_{x=x_1}^{x_2} f_x(x)dx, \tag{2.7}$$

where $f_x(x)$ stands for the PDF of x. And the CDF of x calculated at x_o is defined as [44]:

$$F_x(x_o) = \int_{x=-\infty}^{x_o} f_x(x)dx, \tag{2.8}$$

from which one can see that, the CDF of a continuous random variable is a continuous function.

PDFs have the following similar properties as the PMFs have:

$$0 \leq f_x(x), \tag{2.9}$$

$$\int_{x=-\infty}^{+\infty} f_x(x)dx = 1. \tag{2.10}$$

One of the commonly used PDFs is the Gaussian function.

2.1.1.3. Problems in Probability Theory and Statistics

Kolmogorov defined the general problem of probability theory as follows [46]:

"*Given a CDF, describe outcomes of random experiments for a given theoretical model.*"

Vapnik and Izmailov defined the general problem of statistics as follows [47]:

"*Given independent and identically distributed (IID) observations of outcomes of the same random experiments, estimate the statistical model that defines these observations.*"

Traditional probability theory and statistics have strong and often impractical requirements and assumptions. They also assume the random nature for the variables [26]. Indeed, the appeal of the traditional statistical approach is its solid mathematical foundation and the ability to provide guaranteed performance when data is plenty and created from the same distribution that is hypothesized in the probability law [27]. However, in the field of machine learning, the actual data considered is usually discrete (or discretised), which in probability theory and statistics are modelled as the realisations of the random variables. Moreover, one does not know a priori their distribution. Good results can only be expected on condition that the prior data generation hypothesis is verified. Otherwise, this opens the door for failures, namely, meaningless results [27].

Even in the case that the hypothesised measure meets the realisations, one has to address the difference of working with realisations and random variables, which brings the issue of choosing estimators of the statistical quantities necessary for data analysis [27]. Moreover, different estimators may provide different results. The reason is very likely that the functional properties of the estimators do not preserve all the properties embodied in the statistical quantities. Therefore, they behave differently in the finite (and even in the infinite) sample cases [27].

One can conclude that, the major problem of the traditional data analytic approaches is lying in the strong prior assumptions, which often fail in the reality. As a result, there is a growing demand for alternative new concepts for data analysis that are centred at the actual data collected from the real world rather than at theoretical prior assumptions that need to be confronted for verification with the experimental data as is the case within the traditional statistical approaches.

2.1.2. Typicality and Eccentricity-based Data Analytics

With this need identified (as stated in the end of the previous section), the so-called Typicality- and Eccentricity-based Data Analytics (TEDA) approach was introduced in [48]–[50] as a new concept to address these problems. The core idea of the TEDA approach [48] is to use the data typicality and eccentricity scores calculated from the data for analysing its ensemble properties. As it is concluded in [51], TEDA is a data analytics approach to a "per point" online data analysis without making unrealistic assumptions.

TEDA framework includes the following three operators:

1) Cumulative proximity;

2) Eccentricity;

3) Typicality.

However, TEDA only considers discrete and unimodal operators with the condition that the operators sum up to 1, not integrate to 1. Development of this concept into a systematic framework under the name of Empirical Data Analytics (EDA) framework was done in [25]–[27], and this PhD work was instrumental to this development. In the remainder of this subsection, the three TEDA operators are summarised. The details of EDA framework will be presented in the next subsection.

First of all, a real metric space $\mathbf{R}^M$ and a particular data set/stream $\{\boldsymbol{x}\}_K = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_K\}$ ($\boldsymbol{x}_i = \left[x_{i,1}, x_{i,2}, \ldots, x_{i,M}\right]^T \in \mathbf{R}^M$) are considered, where $K > 2$; the subscripts denote data samples (for a set) or the time instances when they arrive (for a stream). In the remainder of this thesis, all the mathematical derivations are conducted in the K^{th} time instance by default except when specially declared. The most obvious choice of $\mathbf{R}^M$ is the Euclidean space, but TEDA definitions can be extended to Hilbert space as well.

2.1.2.1. Cumulative Proximity

Cumulative proximity, q, was firstly introduced in [48]–[50], which can be seen as a square form of farness. It plays an important role in the TEDA framework and is derived empirically from the observations without making any prior assumptions on the generation model of the data. The cumulative proximity at $\boldsymbol{x}_i$, denoted by $q_K(\boldsymbol{x}_i)$, is expressed as ($i = 1,2,3, \ldots, K$):

$$q_K(\boldsymbol{x}_i) = \sum_{j=1}^{K} d^2\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right), \tag{2.11}$$

where $d\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ denotes the distance/dissimilarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, which can be of any type.

2.1.2.2. Eccentricity

Eccentricity, ξ, is defined as the normalised cumulative proximity [48], [49]. It is an important measure of the ensemble property qualifying data samples away from the mode, and it is useful to disclose distribution tails and anomalies/outliers. The eccentricity at $\boldsymbol{x}_i$, denoted by $\xi_K(\boldsymbol{x}_i)$, is expressed as ($i = 1,2,3, \ldots, K$):

$$\xi_K(\boldsymbol{x}_i) = \frac{2q_K(\boldsymbol{x}_i)}{\sum_{j=1}^{K} q_K(\boldsymbol{x}_j)} = \frac{2\sum_{j=1}^{K} d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sum_{j=1}^{K}\sum_{k=1}^{K} d^2(\boldsymbol{x}_j, \boldsymbol{x}_k)}, \tag{2.12}$$

where the coefficient 2 is used because the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is counted twice in the denominator. From equation (2.12) one can see that $0 \leq \xi_K(\boldsymbol{x}_i) \leq 1$, and there is $\sum_{j=1}^{K} \xi_K(\boldsymbol{x}_j) = 2$.

Eccentricity can also be normalised as [49]:

$$\zeta_K(\boldsymbol{x}_i) = \frac{\xi_K(\boldsymbol{x}_i)}{2}. \tag{2.13}$$

Therefore, for the normalised eccentricity, there is $\sum_{j=1}^{K} \zeta_K(\boldsymbol{x}_j) = 1$

2.1.2.3. Typicality

Typicality, τ, is defined as a complement of eccentricity ($i = 1,2,3, \ldots, K$):

$$\tau_K(\boldsymbol{x}_i) = 1 - \xi_K(\boldsymbol{x}_i). \tag{2.14}$$

One can tell from the above that the typicality also can be summed up to a constant:

$$\sum_{j=1}^{K} \tau_K(\boldsymbol{x}_i) = K - 2, \tag{2.15}$$

and it can be normalised as:

$$t_K(\boldsymbol{x}_i) = \frac{\tau_K(\boldsymbol{x}_i)}{K-2}. \tag{2.16}$$

Similar to the normalised eccentricity, the sum of the normalised typicality is sum up to 1, $\sum_{j=1}^{K} t_K(\boldsymbol{x}_i) = 1$.

The three TEDA operators can be updated recursively online on a sample-by-sample basis, and the recursive calculation expressions are of paramount in streaming data processing, the details of which can be found in [48]–[51].

2.1.3. Empirical Data Analytics

The latest development in the field of data analysis, Empirical Data Analytics (EDA) computational methodology takes the TEDA framework one level further.

As a systematic methodology of nonparametric quantities introduced in [25]–[27] based on the ensemble properties and mutual distribution of the empirical discrete observations, the EDA framework is a strong alternative to the traditional statistics and probability theory, but is free from the paradoxes and problems that the traditional approaches are suffering from [26], [27]. This is because that all the non-parametric EDA quantities are derived from the empirically observed data without making any prior assumptions or using predefined parameters. Thus, it can be viewed as a powerful extension of the traditional probability theory and statistical learning.

EDA framework touches the very foundation of data analytics, and thus, there are a wide range of applications including, but not limited to, data analysis, clustering, data partitioning, classification, prediction, anomaly detection, fuzzy rule-based (FRB) system, deep rule-based (DRB) system, etc.

EDA also serves as the main theoretical basis of the self-organising transparent machine learning techniques presented in this thesis. In this subsection, the nonparametric discrete quantities within EDA framework and their corresponding recursive expressions are summarised. The relationship between EDA quantities and the well-known Chebyshev inequality [52] is presented as well.

The nonparametric discrete EDA quantities include:

1) Cumulative proximity;

2) Eccentricity and standardised eccentricity;

3) Unimodal and multimodal density;

4) Unimodal and multimodal typicality.

EDA framework shares the same expressions for cumulative proximity and eccentricity with TEDA, but redefines the typicality in two different versions (unimodal and multimodal), and further introduces standardised eccentricity, unimodal and multimodal density. However, it has to be stressed that the EDA framework is not limited to the concepts presented in this thesis, but to a much wider range in both discrete and continuous domains [25]–[27].

Firstly, in addition to the TEDA framework presented in section 2.1.2, within the data set/stream $\{\boldsymbol{x}\}_K$, it is further taken into consideration that some data samples may repeat more than once, namely $\exists \boldsymbol{x}_i = \boldsymbol{x}_j, i \neq j$. The set of the sorted unique data samples, denoted by $\{\boldsymbol{u}\}_N = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N\}$, and the corresponding number of occurrence, denoted by $\{f\}_N = \{f_1, f_2, \ldots, f_N\}$ ($\sum_{j=1}^{N} f_j = K$), can be obtained automatically from the data, where N denotes the number of unique data samples. With $\{\boldsymbol{u}\}_N$ and $\{f\}_N$, the primary data set/stream can be constructed.

2.1.3.1. Standardised Eccentricity

As the value of eccentricity decreases very fast with the increase of the amount of data, K (see equation (2.12)), the standardised eccentricity, ε, is introduced as $(i = 1,2,3, \dots, K)$ [48], [49]:

$$\varepsilon_K(\boldsymbol{x}_i) = K\xi_K(\boldsymbol{x}_i) = \frac{2q_K(\boldsymbol{x}_i)}{\frac{1}{K}\sum_{j=1}^{K} q_K(\boldsymbol{x}_j)} = \frac{2\sum_{j=1}^{K} d^2(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\frac{1}{K}\sum_{j=1}^{K}\sum_{k=1}^{K} d^2(\boldsymbol{x}_j,\boldsymbol{x}_k)}. \tag{2.17}$$

There is $\sum_{j=1}^{K} \varepsilon_K(\boldsymbol{x}_j) = 2K$.

2.1.3.2. Unimodal density

Unimodal density, D, was firstly introduced in [48] and is redefined as the inverse of standardised eccentricity in [25], [26]. It plays as the indictor of the main mode within EDA framework. The unimodal density at $\boldsymbol{x}_i$, denoted by $D_K(\boldsymbol{x}_i)$, is given as $(i = 1,2,3, \dots, K)$:

$$D_K(\boldsymbol{x}_i) = \varepsilon_K^{-1}(\boldsymbol{x}_i) = \frac{\sum_{j=1}^{K} q_K(\boldsymbol{x}_j)}{2Kq_K(\boldsymbol{x}_i)} = \frac{\sum_{j=1}^{K}\sum_{k=1}^{K} d^2(\boldsymbol{x}_j,\boldsymbol{x}_k)}{2K\sum_{j=1}^{K} d^2(\boldsymbol{x}_i,\boldsymbol{x}_j)}, \tag{2.18}$$

where $0 \le D_K(\boldsymbol{x}_i) \le 1$. Unimodal density, in both the discrete and continuous forms, is very fundamental and resembles the membership functions of fuzzy sets, which represents the degree of truth in fuzzy logic and can take any value from the interval [0,1] [53]. More details of fuzzy sets and systems are given in section 2.2.1. The link between the unimodal density and membership function is explained in detail in [54].

2.1.3.3. Multimodal density

Multimodal density, D^G [25]–[27] is valid at the unique data samples only. The multimodal density at the unique data sample $\boldsymbol{u}_i$ $(i = 1,2,3, \dots, N)$, denoted by $D_K^G(\boldsymbol{u}_i)$, is defined as the combination of the unimodal density weighted by the corresponding frequency of occurrence of this unique data sample f_i as:

$$D_K^G(\boldsymbol{u}_i) = f_i D_K(\boldsymbol{u}_i) = f_i \frac{\sum_{j=1}^{K}\sum_{k=1}^{K} d^2(\boldsymbol{x}_j,\boldsymbol{x}_k)}{2K\sum_{j=1}^{K} d^2(\boldsymbol{x}_i,\boldsymbol{x}_j)}. \tag{2.19}$$

The expression of D^G is fundamental because it combines information about the frequencies of occurrence of data samples and their locations in the data space.

2.1.3.4. Unimodal Typicality

In EDA framework, the typicality in the TEDA is redefined and renamed as the unimodal typicality, which is the normalised data density [25]–[27]. The unimodal typicality at $\boldsymbol{x}_i$, denoted by $\tau_K(\boldsymbol{x}_i)$, is given as $(i = 1,2,3, \dots, K)$:

$$\tau_K(\boldsymbol{x}_i) = \frac{D_K(\boldsymbol{x}_i)}{\sum_{k=1}^{K} D_K(\boldsymbol{x}_k)} = \frac{\sum_{j=1}^{K} d^{-2}(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\sum_{j=1}^{K}\sum_{k=1}^{K} d^{-2}(\boldsymbol{x}_j,\boldsymbol{x}_k)}. \quad (2.20)$$

The unimodal typicality resembles the traditional unimodal PMF, but is automatically defined in the data support unlike the PMF which may have nonzero values for infeasible values of the random variable unless being specifically constrained [27].

2.1.3.5. Multimodal typicality

The multimodal typicality is newly introduced in EDA [25]–[27], which is directly derived from the experimental data with the ability of providing multimodal distributions automatically without the need of user decisions or any processing techniques [27]. The multimodal typicality at a unique data sample $\boldsymbol{u}_i$ $(i = 1,2,3,\dots,N)$, denoted by $\tau_K^G(\boldsymbol{u}_i)$, is expressed as a combination of the normalised unimodal density weighted by the corresponding frequency of occurrence, f_i:

$$\tau_K^G(\boldsymbol{u}_i) = \frac{f_i D_K(\boldsymbol{u}_i)}{\sum_{k=1}^{K} f_k D_K(\boldsymbol{u}_k)} = \frac{\sum_{j=1}^{K} f_i d^{-2}(\boldsymbol{u}_i,\boldsymbol{u}_j)}{\sum_{j=1}^{K}\sum_{k=1}^{K} f_k d^{-2}(\boldsymbol{u}_j,\boldsymbol{u}_k)}. \quad (2.21)$$

The multimodal typicality has the following properties [27]:

1) Sums up to 1;

2) The value is within [0, 1];

3) Provides a closed analytic form;

4) No requirement for prior assumptions as well as any user- or problem- specific thresholds and parameters;

5) Its value calculated on infeasible data is always zero.

2.1.3.6. Recursive Expressions

The recursive calculation expressions of the nonparametric EDA quantities play a significant role in streaming data processing. They ensure the processing techniques to be of one-pass type, and thus, minimise both the memory- and computation- loads.

A. General case

The general recursive expressions of the EDA quantities are given as follows [55]:

$$q_K(\boldsymbol{x}_i) = q_{K-1}(\boldsymbol{x}_i) + d^2(\boldsymbol{x}_i, \boldsymbol{x}_K); \quad (2.22)$$

$$\sum_{j=1}^{K} q_K(\boldsymbol{x}_j) = \sum_{j=1}^{K-1} q_{K-1}(\boldsymbol{x}_j) + 2q_K(\boldsymbol{x}_K). \quad (2.23)$$

With equations (2.22) and (2.23), all the EDA quantities given in the previous section can be recursively calculated for all types of distance metric/dissimilarity. If the Euclidean distance, Mahalanobis distance, cosine dissimilarity or some other types of distances/dissimilarity are used, one can have more elegant recursive expressions.

B. Euclidean distance case

Using Euclidean distance, defined as $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j)}$, the recursive expression of $q_K(\boldsymbol{x}_i)$ and $\sum_{j=1}^{K} q_K(\boldsymbol{x}_j)$ are given as:

$$q_K(\boldsymbol{x}_i) = K(\|\boldsymbol{x}_i - \boldsymbol{\mu}_K\|^2 + X_K - \|\boldsymbol{\mu}_K\|^2); \tag{2.24}$$

$$\sum_{j=1}^{K} q_K(\boldsymbol{x}_j) = 2K^2(X_K - \|\boldsymbol{\mu}_K\|^2), \tag{2.25}$$

where $\boldsymbol{\mu}_K$ and X_K are the means of $\{\boldsymbol{x}\}_K$ and $\{\|\boldsymbol{x}\|^2\}_K$, respectively, and both of them can be updated recursively as:

$$\boldsymbol{\mu}_K = \frac{K-1}{K}\boldsymbol{\mu}_{K-1} + \frac{1}{K}\boldsymbol{x}_K; \tag{2.26}$$

$$X_K = \frac{K-1}{K}X_{K-1} + \frac{1}{K}\|\boldsymbol{x}_K\|^2. \tag{2.27}$$

C. Mahalanobis distance case

Using Mahalanobis distance [56], defined as $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{\Sigma}_K^{-1}(\boldsymbol{x}_i - \boldsymbol{x}_j)}$, the recursive calculation expressions of $q_K(\boldsymbol{x}_i)$ and $\sum_{j=1}^{K} q_K(\boldsymbol{x}_j)$ are given as:

$$\begin{aligned} q_K(\boldsymbol{x}_i) &= K((\boldsymbol{x}_i - \boldsymbol{\mu}_K)^T \boldsymbol{\Sigma}_K^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_K) + X_K - \boldsymbol{\mu}_K^T \boldsymbol{\Sigma}_K^{-1}\boldsymbol{\mu}_K) \\ &= K((\boldsymbol{x}_i - \boldsymbol{\mu}_K)^T \boldsymbol{\Sigma}_K^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_K) + M)\ ; \end{aligned} \tag{2.28}$$

$$\sum_{j=1}^{K} q_K(\boldsymbol{x}_j) = 2K^2(X_K - \boldsymbol{\mu}_K^T \boldsymbol{\Sigma}_K^{-1}\boldsymbol{\mu}_K) = 2K^2 M\ ; \tag{2.29}$$

where $\boldsymbol{\mu}_K$ is the mean of $\{\boldsymbol{x}\}_K$; $\boldsymbol{\Sigma}_K$ is the covariance matrix, $\boldsymbol{\Sigma}_K = \frac{1}{K-1}\sum_{k=1}^{K}(\boldsymbol{x}_k - \boldsymbol{\mu}_K)(\boldsymbol{x}_k - \boldsymbol{\mu}_K)^T$; $X_K = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{x}_k^T \boldsymbol{\Sigma}_K^{-1}\boldsymbol{x}_k$; $X_K - \boldsymbol{\mu}_K^T \boldsymbol{\Sigma}_K^{-1}\boldsymbol{\mu}_K = M$[51].

$\boldsymbol{\Sigma}_K$ can be updated recursively as:

$$\mathbf{X}_K = \frac{K-1}{K}\mathbf{X}_{K-1} + \frac{1}{K}\boldsymbol{x}_K\boldsymbol{x}_K^T; \tag{2.30}$$

$$\boldsymbol{\Sigma}_K = \frac{K}{K-1}(\mathbf{X}_K - \boldsymbol{\mu}_K\boldsymbol{\mu}_K^T). \tag{2.31}$$

D. Cosine dissimilarity case

Using cosine dissimilarity, defined as $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{2 - 2cos\left(\theta_{\boldsymbol{x}_i,\boldsymbol{x}_j}\right)} = \left\| \frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} - \frac{\boldsymbol{x}_j}{\|\boldsymbol{x}_j\|} \right\|$ [32], [33], the recursive calculation expressions of $q_K(\boldsymbol{x}_i)$ and $\sum_{j=1}^{K} q_K(\boldsymbol{x}_j)$ are given as:

$$q_K(\boldsymbol{x}_i) = K\left(\left\| \frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} - \overline{\boldsymbol{\mu}}_K \right\|^2 + \bar{X}_K - \|\overline{\boldsymbol{\mu}}_K\|^2\right); \tag{2.32}$$

$$\sum_{j=1}^{K} q_K(\boldsymbol{x}_j) = 2K^2(\bar{X}_K - \|\overline{\boldsymbol{\mu}}_K\|^2), \tag{2.33}$$

where $\overline{\boldsymbol{\mu}}_K$ and $\bar{X}_K$ are the means of $\left\{\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\right\}_K$ and $\left\{\left\|\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\right\|^2\right\}_K$, respectively, and both of them can be updated recursively as:

$$\overline{\boldsymbol{\mu}}_K = \frac{K-1}{K}\overline{\boldsymbol{\mu}}_{K-1} + \frac{1}{K}\frac{\boldsymbol{x}_K}{\|\boldsymbol{x}_K\|}; \tag{2.34}$$

$$\bar{X}_K = \frac{K-1}{K}\bar{X}_{K-1} + \frac{1}{K}\left\|\frac{\boldsymbol{x}_K}{\|\boldsymbol{x}_K\|}\right\|^2 = 1. \tag{2.35}$$

E. Direction-Aware distance case [33]

The direction-aware distance is a recently introduced distance metric combining the advantages of Euclidean distance and cosine similarity in the Euclidean space domain. The direction-aware distance consists of a magnitude component and an angular component and has the following expression [33]:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\lambda_M d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) + \lambda_A d_A^2(\boldsymbol{x}_i, \boldsymbol{x}_j)}, \tag{2.36}$$

where $d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ and $d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{1 - cos\left(\theta_{\boldsymbol{x}_i,\boldsymbol{x}_j}\right)} = \frac{1}{\sqrt{2}}\left\| \frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} - \frac{\boldsymbol{x}_j}{\|\boldsymbol{x}_j\|} \right\|$; λ_M and λ_A are a pair of scaling coefficients, and there are $\lambda_M > 0$ and $\lambda_A > 0$.

In [33], the direction-aware distance is proven to be a full metric which satisfies the following properties for $\forall \boldsymbol{x}_i, \boldsymbol{x}_j$ [57]:

1) Non-negativity: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$;

2) Identity of indiscernibles: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0 \ iff \ \boldsymbol{x}_i = \boldsymbol{x}_j$;

3) Symmetry: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$;

4) Triangle inequality: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) + d(\boldsymbol{x}_i, \boldsymbol{x}_k) \geq d(\boldsymbol{x}_j, \boldsymbol{x}_k)$.

With the direction-aware distance, the recursive calculation expressions of $q_K(\boldsymbol{x}_i)$ and $\sum_{j=1}^{K} q_K(\boldsymbol{x}_j)$ are given as:

$$q_K(\boldsymbol{x}_i) = K\left((\|\boldsymbol{x}_i - \boldsymbol{\mu}_K\|^2 + X_K - \|\boldsymbol{\mu}_K\|^2) + \frac{1}{2}\left(\left\|\frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} - \overline{\boldsymbol{\mu}}_K\right\|^2 + 1 - \|\overline{\boldsymbol{\mu}}_K\|^2\right)\right); \tag{2.37}$$

$$\sum_{j=1}^{K} q_K(\boldsymbol{x}_j) = K^2(2(X_K - \|\boldsymbol{\mu}_K\|^2) + 1 - \|\overline{\boldsymbol{\mu}}_K\|^2), \tag{2.38}$$

where $\boldsymbol{\mu}_K$, X_K and $\overline{\boldsymbol{\mu}}_K$ can be updated recursively using equations (2.26), (2.27) and (2.34).

2.1.3.7. Chebyshev Inequality

The well-known Chebyshev inequality in the traditional probability theory and statistics [52] describes the probability that a certain data sample $\boldsymbol{x}$, is more than $n\sigma$ distance away from the mean, $\boldsymbol{\mu}$, where σ denotes the standard deviation. With the Euclidean distance used, the Chebyshev inequality can be reformulated as [2]–[4]:

$$P(\|\boldsymbol{x} - \boldsymbol{\mu}\|^2 < n^2\sigma^2) > 1 - \frac{1}{n^2}, \tag{2.39}$$

and the possibility of the point $\boldsymbol{x}$ to be an outlier is given by:

$$P(\|\boldsymbol{x} - \boldsymbol{\mu}\|^2 \geq n^2\sigma^2) \leq \frac{1}{n^2}. \tag{2.40}$$

It can be proven that exactly the same result can be provided within EDA through the standardised eccentricity for the Euclidean distance [49]:

$$P(\varepsilon_K(\boldsymbol{x}_i) < 1 + n^2) > 1 - \frac{1}{n^2}; \tag{2.41}$$

$$P(\varepsilon_K(\boldsymbol{x}_i) \geq 1 + n^2) \leq \frac{1}{n^2}. \tag{2.42}$$

Similarly, the Chebyshev inequality in the form of density is expressed as [26]:

$$P\left(D_K(\boldsymbol{x}_i) > \frac{1}{1+n^2}\right) > 1 - \frac{1}{n^2}; \tag{2.43}$$

$$P\left(D_K(\boldsymbol{x}_i) \leq \frac{1}{1+n^2}\right) \leq \frac{1}{n^2}. \tag{2.44}$$

One can see that the attractiveness of equations (2.41)-(2.44) in comparison with equations (2.39)-(2.40) is that no prior assumptions are required within EDA on the nature of the data (random or deterministic), the generation model, the amount of data and their independence. In addition, the results are more elegant and similar expressions can be derived for Mahalanobis distance, cosine dissimilarity as well as other types of distance and dissimilarity [26], [49].

2.1.3.8. Properties of the EDA quantities

The EDA framework is entirely based on the ensemble properties and mutual distribution of the empirically observed data. Compared with the existing statistical approaches, there are a few outstanding unique properties within EDA quantities [25]–[27]:

1) They are entirely based on the empirically observed experimental data and their mutual distribution in the data space;
2) They do not require any user- or problem-specific thresholds and parameters to be predefined;
3) They do not require any model of data generation to be assumed (random or deterministic);
4) Individual data samples (observations) do not need to be independent or identically distributed; on the contrary, their mutual dependence is taken into account directly through the mutual distance between the data points/samples;
5) They also do not require infinite number of observations and can work with as little as 2 data samples;
6) They can be calculated recursively for many types of distance metrics.

2.2. Computational Intelligence Methodologies Survey

Computational intelligence is a set of nature-inspired computational methodologies and approaches to address complex real-world problems to which mathematical or traditional modelling struggles. The main approaches of computational intelligence include fuzzy systems, artificial neural networks (ANNs), evolutionary computation (EC), etc.

This section gives a review focusing on fuzzy systems and ANNs. Deep learning, as the later development of ANNs, will be also covered. A brief review on EC will be also presented.

2.2.1 Fuzzy Sets and Systems

Fuzzy sets theory and fuzzy rule-based (FRB) systems were firstly introduced in the seminal paper by Professor Lotfi Zadeh [53] over 50 years ago. The FRB systems are a set of fuzzy rules. The antecedent parts of the fuzzy rules are determined by fuzzy sets, which are defined by parameterised scalar membership functions. In this section, three types of fuzzy rules (Zadeh-Mamdani type [58], Takagi-Sugeno type [59] and AnYa type [60]) are reviewed. However, it has to be stressed that there are other types of fuzzy systems (relational

[61], etc.), only the most widely used and representative ones are considered in this thesis. The FRB system identification process is reviewed briefly in this section as well.

2.2.1.1. Zadeh-Mamdani Type, Takagi-Sugeno Type and AnYa Type Fuzzy Rules

A fuzzy rule of Zadeh-Mamdani type has the following expression [58]:

$$\begin{gathered} IF\ (x_1\ is\ L_{i,1})\ AND(x_2\ is\ L_{i,2})\ AND\ \ldots AND\ (x_M\ is\ L_{i,M}) \\ THEN\ (y_i\ is\ L_{i,out}) \end{gathered}, \tag{2.45}$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_M]^T$; $L_{i,j}$ is the jth reference value of the ith fuzzy rule; y_i is the outcome of the ith fuzzy rule.

A fuzzy rule of Takagi-Sugeno type has the following expression [11], [59], [62]:

$$\begin{gathered} IF\ (x_1\ is\ L_{i,1})\ AND(x_2\ is\ L_{i,2})\ AND\ \ldots AND\ (x_M\ is\ L_{i,M}) \\ THEN\ (y_i = [1, \boldsymbol{x}^T]\boldsymbol{a}_i) \end{gathered}, \tag{2.46}$$

where $\boldsymbol{a}_i$ is the $(M+1)\times 1$ dimensional parameterised vector of the ith fuzzy rule for linear regression.

One can see that, the Zadeh-Mamdani type and Takagi-Sugeno type fuzzy rules share the same type of antecedent (IF) part, but differ in the consequent (THEN) part. Usually, to build the antecedent (IF) parts of the two types of fuzzy rules, a number of ad hoc choices have to be made [63], which include:

1) The types of membership functions, i.e. triangular type, Gaussian type, bell type, etc.;

2) Linguistic terms for each rule;

3) The area of influence of each rule, i.e. hyper-rectangle, -sphere, -ellipsoid;

4) The prototypes for the fuzzy sets;

5) The parameters for the membership functions.

In contrast, as a recently introduced type of fuzzy rules, the AnYa type has a different, simplified antecedent (IF) part, which can be viewed as a generalisation of the two predecessors. A 0-order AnYa type fuzzy rule is expressed as [60]:

$$IF\ (\boldsymbol{x} \sim \boldsymbol{p}_i)\ \ THEN\ (y_i\ is\ L_{i,out}), \tag{2.47}$$

and a 1st order AnYa type fuzzy rule is expressed as [60]:

$$IF\ (\boldsymbol{x} \sim \boldsymbol{p}_i)\ \ THEN\ (y_i = [1, \boldsymbol{x}^T]\boldsymbol{a}_i), \tag{2.48}$$

where "~" denotes similarity, which can also be seen as a fuzzy degree of satisfaction/membership [54], [60] or typicality [26]; $\boldsymbol{p}_i$ is the prototype of the i^{th} fuzzy rule, which is also the only decision required to be made by human experts, but it is still optional as the prototype can also be identified via the data-driven approaches [54].

The AnYa type fuzzy rule simplifies the antecedent (IF) part of the traditional fuzzy rule into a prototype [54], [60], which is a vector representing the focal point of the nonparametric, shape-free data cloud consisting of data samples associated with this focal point resembling Voronoi tessellation [64]. Compared with the antecedent (IF) part of the traditional (Zadeh-Mamdani type and Takagi-Sugeno type) fuzzy rules, which (although the structure and some of the parameters can be learnt from the data) requires heavy involvements of human experts and prior knowledge of the problems to formulate the whole rule, this simplification of the AnYa type significantly reduces the efforts of human experts and, at the same time, largely enhances the objectiveness of the FRB system [60]. The comparison between the three types of fuzzy rules is tabulated in Table 1 for clarity [55], [60].

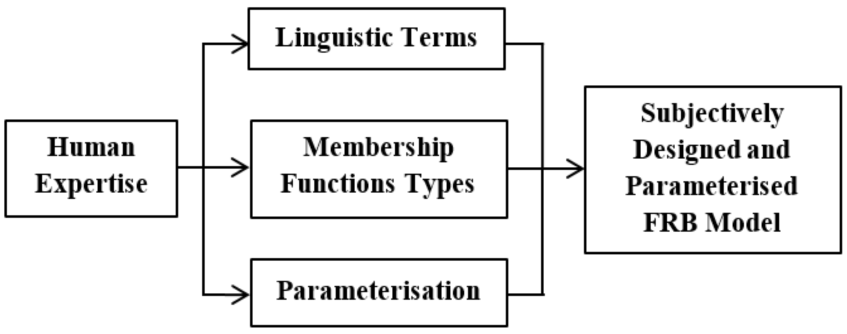
Table 1. A comparison between three types of fuzzy rules

Type		Antecedent (IF) part	Consequent (THEN) Part	De-fuzzification
Zadeh-Mamdani		Scalar, parameterised fuzzy sets	Scalar, parameterised fuzzy sets	Central of gravity
Takagi-Sugeno			Functional (usually linear)	Fuzzily weighted sum (average)
AnYa	0-order	Prototypes, data clouds	Scalar, parameterised fuzzy sets	Winner takes all
	1st order		Functional (usually linear)	Fuzzily weighted sum (average)

2.2.2.2. FRB System Identification

Initially, the fuzzy sets theory was introduced to approximate the data distribution by the subjectivist definition of uncertainty, which completely departed from objective observation and, instead, relies on the human experts' knowledge [27]. The main issue in the design of the fuzzy sets and FRB systems is how to define the membership functions by which they are defined in first place [54].

The main procedure of the traditional way of designing FRB systems, namely, the subjective approach, is summarised in Figure 1 [54].



Linguistic Terms
Human Expertise
Membership Functions Types
Subjectively Designed and Parameterised FRB Model
Parameterisation

Figure 1. Main procedure of the subjective approach for FRB system identification.

The subjective approach has its own very strong rationale in the two-way process of:

1) Formalising expert knowledge and representing it in a mathematical form through the membership functions;

2) Extracting and representing from data human-intelligible and understandable, transparent linguistic information in the form of IF …THEN… rules [55].

However, the following issues appear during the process:

1) Defining a membership function requires many ad hoc decisions;

2) Membership functions often differ significantly from the real data distribution.

Moreover, the so-called "curse of dimensionality" may result from handcrafting traditional FRB systems for high dimensional problems because of the exponential growth of the number of fuzzy sets required.

In 1990s, the so-called data-driven design approach (the objective one) started to be popular and was developed [55]. The main procedure of the objective approach for FRB system identification is depicted in Figure 2 [54].

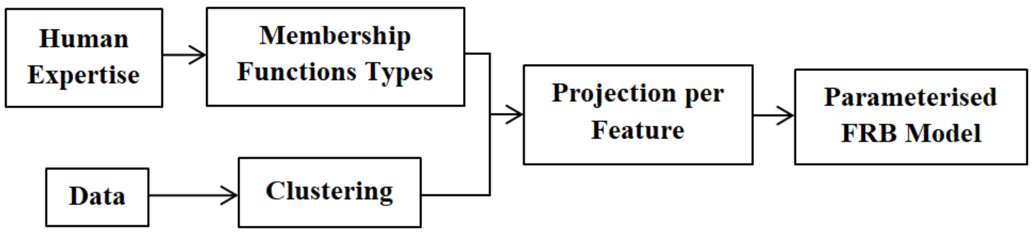


Figure 2. Main procedure of the objective approach for FRB system identification.

Nonetheless, it is practically very difficult and controversial to define membership functions both from experts and from data. This is also related to the more general issue of assumptions made and handcrafting that machine learning (including statistical methods) are facing. Therefore, in [54], an alternative membership function based on the Cauchy type data density (equation (2.18)) [26], [27] is introduced to the AnYa type FRB system, which frees the FRB system from ad hoc decisions and prior assumptions. In this thesis, the AnYa type FRB system with the Cauchy type membership function is employed.

2.2.2. Artificial Neural Networks

Artificial neural networks (ANNs) are the computing systems inspired by the biological neural networks of the animal brains aiming to resolve perception and recognition problems. They are capable of approximating nonlinear relationships between inputs and outputs.

The basic elements of the ANNs are neurons, which receive input, change their internal state (activation) correspondingly, and produce output depending on the input and activation. ANNs are formed by connecting the output of certain neurons to the input of other neurons forming a directed, weighted graph. The adaptive weights along paths between neurons and the functions that compute the activation can be modified (adapted) by learning algorithms. However, unlike the biological neural networks, once an ANN is formed, the connections between artificial neurons are not usually added or removed.

One of the earliest and best known computational models for neural networks based on mathematics and algorithms was introduced by Warren McCulloch and Walter Pitts in 1943 [65] called threshold logic, however, the technology available at that time were insufficient for them to work on practical problems. In 1970s and 1980s, with the development of the computational resources, a number of new, more complex ANNs started to emerge.

Nowadays, deep learning neural networks (DLNNs) have gained a lot of popularity in both the academic circles and the general public [19], [66]. In fact, deep learning is the latest name of ANNs [66]. However, DLNNs have also gone beyond the original neuroscientific perspective, but appear to be a more general principle of learning multiple levels of composition, which can be applied in machine learning frameworks that are not necessarily naturally inspired [66]. Currently, the popular variants of the ANNs include, but not limited to 1) feedforward neural networks [4]; 2) deep convolutional neural networks (DCNNs) [21], [24]; 3) recurrent neural networks (long short-term memory) [67]; 4) deep belief networks [68] and 5) spiking neural networks [69], etc.

As the feedforward neural networks (NNs) and DCNNs are directly related to the main topic of this thesis, this section will focus on reviewing these two particular types.

2.2.2.1. Feedforward Neural Network

Feedforward NNs are a class of ANNs whose neurons form an acyclic graph where information moves only in one direction from input to output. They are extensively used in pattern recognition. The general architecture of a multilayer feedforward NN is depicted in Figure 3.

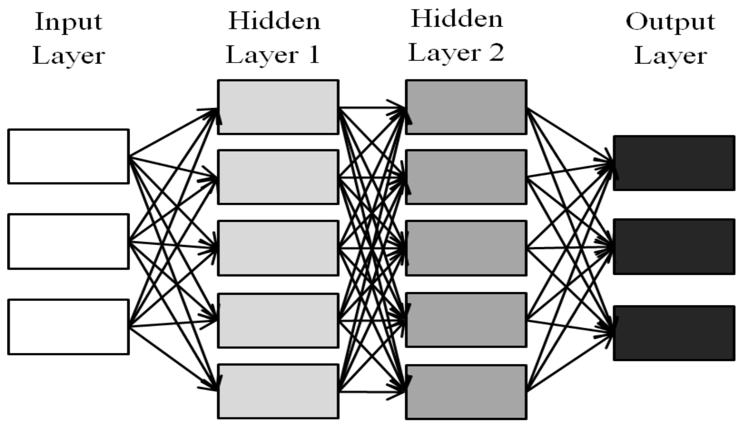


Figure 3. General architecture of a multilayer feedforward NN.

A typical multilayer feedforward neural network consists of three types of layers: input layer, one or more hidden layers and output layer. Each layer is a group of neurons receiving connections from the neurons of the previous layer. Neurons inside a layer are not connected to each other.

Input layer is the first layer of the network and it receives no connections from other layers, but instead, uses input vector as its activation. Input layer is fully connected to the first hidden layer. Each hidden layer is fully connected to the next hidden layer, and the last hidden layer is fully connected to output layer. The activation of output units is considered to be the output of the feedforward neural network. The output of the network is the result of the transformations of input data through neurons and layers in a form of distributed representation consisting of a huge number of weighted local representations across the entire networks.

Backpropagation procedure (backward propagation of errors) is the most widely used supervised learning algorithm for adapting connection weights of feedforward NNs [19]. Weights of the network are tuned to minimise square error between the system output and the target value:

$$\varepsilon = \sum_i (y_i - t_i)^2 \,, \tag{2.49}$$

where ε stands for overall square error; y_i is the system output corresponding to the i^{th} input and t_i is the respective target value.

Backpropagation is nothing more than a practical application of the chain rule for derivatives. The key insight is that the derivative of the objective function with respect to the input of a layer can be computed by working backwards from the gradient with respect to the output of that layer. The backpropagation equation can be applied repeatedly to propagate gradients through all modules, starting from the output layer all the way to the input layer. Once these gradients have been computed, it is straightforward to compute the gradients with respect to the weights of each module [19].

Although feedforward NNs are very powerful, they suffer from the following drawbacks:

1) The structure is complex and considered as a black box;

2) The structure identification requires a number of ad hoc decisions, i.e. number of neurons, number of layers;

3) The training process is computationally expensive and once it is finished, the parameters of the NNs cannot be updated and requires a full retraining if new data samples are given.

2.2.2.2. Deep Convolutional Neural Network

Convolutional neural network (CNN) was firstly introduced by Kunihiko Fukushima [70] almost 50 years ago and was significantly improved later [71].

Currently, DCNNs are the state-of-the-art approaches in the field of computer vision. A number of publications have demonstrated that DCNNs can produce highly accurate results in various image processing problems including, but not limited to, handwritten digits recognition [22], [24], [72]–[74], object recognition [21], [23], [75], [76], human action recognition [77], [78], remote sensing image classification [79]–[82], etc. Some publications suggest that the DCNNs can match the human performance on the handwritten digits recognition problems [22], [24], [73], [74].

Except the input and output layers, a typical CNN consists of a number of hidden layers, which can be a combination of the following four types:

1) Convolution layer;

2) Pooling layer;

3) Normalisation layer;

4) Fully connected layer.

The convolutional and pooling layers in DCNNs are directly inspired by the classic notions of simple cells and complex cells in visual neuroscience [83].

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The role of the convolutional layer is to detect local conjunctions of features from the previous layer.

Pooling layer is for merging semantically similar features into one. A typical max pooling unit (which is the most commonly used one) calculates the maximum of a local patch of units in each sub-region of the image. Neighbouring pooling units take input from patches that are shifted by more than one row or column resulting in the reduction of the dimensionality of the representation and the increase of robustness to small shifts and distortions.

Normalisation layer is useful when using neurons with unbounded activations (e.g. rectified linear neurons), because it permits the detection of high-frequency features with a big neuron response, while damping responses that are uniformly large in a local neighbourhood.

Fully connected layer connects every neuron in the previous layer to every neuron in it, which is in principle the same as the feedforward NN as described in subsection 2.2.2.1. However, one major difference between the fully connected layer in the DCNN and the one in the feedforward NN is that the fully connected layer of the DCNN only connects to a small region of the input volume, while the fully connected layer of feedforward NN connected to all the neurons of the previous layer.

Recent DCNN architectures have 10 to 20 layers of rectified linear neurons, hundreds of millions of weights, and billions of connections between units. Thanks to the very large progress in hardware, software and algorithm parallelisation, the training times can be only a few hours if enough computational resources are provided, which could be extremely expensive. The performance of DCNN-based vision systems has caused most major technology companies, including Google, Facebook, Microsoft, Baidu to initiate research and development projects and to deploy DCNN-based image understanding products and services [19].

Nonetheless, ANNs/DCNNs still have a number of deficiencies and shortcomings:

1) The computational burden of training using huge amount of data is still very heavy;

2) The training process is opaque, and the classifier has low or no human interpretability (black box type);

3) The training process is limited to offline and requires re-training for samples with feature properties different than the observed samples, as well as for samples from unseen classes;

4) Its internal structure identification involves a number of ad hoc decisions, i.e. number of layers, the order of the layers, the types of convolutional kernel, the type of pooling.

2.2.3. Evolutionary Computation

Evolutionary computation (EC) is a class of randomised search and optimisation algorithms inspired by the principles of evolutional and natural genetics [84], [85]. The origins of EC can be dated back to the late 1950's [86]–[88], but the works from John Holland [86], Ingo Rechenberg [89] and Lawrence Fogel [90] laid the foundation for its popularity today.

Currently, the main components of EC include genetic algorithms [91], [92], evolutionary strategies [93], genetic programming [94] and particle swarm optimisation [95], [96]. However, this thesis focuses on the fuzzy systems and ANNs, and, thus, a detailed review on the EC is not conducted. Nonetheless, one can find the more detailed, systematic introductions to EC in [84], [85], [97].

2.3. Machine Learning Techniques Survey

In this section, the machine learning techniques including clustering, classification, regression and anomaly detection will be reviewed.

2.3.1. Cluster Algorithms

Clustering, alternatively, data partitioning, has a variety of goals, all related to grouping or segmenting a collection of data into subsets or "clusters" such that data samples within the same cluster are more closely related to each other than other data samples assigned to different clusters [4].

Clustering algorithms have long been considered as unsupervised machine learning techniques for finding out the underlying groups and pattern within the data. Based on their operating mechanism, clustering algorithms can be divided into the following main types [98]:

1) Hierarchical clustering;

2) Centroid-based clustering;

3) Model-based clustering;

4) Density-based clustering;

5) Distribution-based clustering;

6) Soft-computing clustering.

Since there are a huge number of clustering algorithms published, it is impossible to cover all the published algorithms within this thesis. In this section, only the most typical and representative clustering algorithms of the six types are reviewed, and their later variants are also given.

2.3.1.1. Hierarchical Clustering

A hierarchical clustering algorithm [99] produces a dendrogram representing nested groupings of patterns and similarity levels at different granularities, which offers more flexibility for exploratory analysis. The clustering result is achieved by cutting the dendrogram at the desired similarity level [98]. Some studies suggest that hierarchical algorithms can produce better-quality clusters [100]. There are two major types based on their bottom-up or top-down fashion:

1) Agglomerative hierarchical clustering [101], [102]

The methods treat each data sample as a cluster of its own initially, and merge them successively until obtain the desired cluster structure [98].

2) Divisive hierarchical clustering [103], [104]

The methods achieve the clustering result via a contrary direction. They treat all the data samples as a single cluster and successively divide the cluster into sub-clusters until the desired clustering structure is obtained [98].

More recently, a new type of hierarchical clustering approaches named affinity propagation was introduced in [105], which can achieve the desired cluster structure without cutting the dendrogram. This algorithm takes as input measures of similarity between pairs of data points and simultaneously considers all data samples as potential exemplars. Real-valued messages, whose magnitude represents the affinity of one data sample for choosing another data sample as its cluster centre, are exchanged between data samples until a high-quality set of exemplars and corresponding clusters gradually emerge. Nonetheless, this new approach, in fact, optimises the dendrogram cutting by hardcoding the mathematical rules for achieving the optimal partitions and predefining parameters.

In general, the hierarchical clustering approaches tend to maintain good performance on datasets with non-isotropic clusters, including well-separated, chain-like and concentric ones.

The main drawbacks of the hierarchical approaches are:

1) The computation- and memory- efficiency of the approaches deteriorates fast with the increase of the scale of the data;

2) They do not have back-tracking capability;

3) They require prior knowledge of the problem, which means the performance of the hierarchical approaches is not guaranteed in real cases where the prior knowledge is not available.

2.3.1.2. Centroid-based Clustering

Centroid-based clustering methods start from an initial partitioning and relocate instances by moving them from one cluster to another. The methods require an exhaustive enumeration process of all possible partitions and use certain greedy heuristics for iterative optimisation.

The basic idea of the centroid-based clustering algorithms is to find a clustering structure that minimises a certain error criterion that measures the distance of each data sample to its representative value, and the process is called error minimisation. The most well-known criterion is the sum of squared error.

The simplest and most commonly used algorithm is the k-means algorithm [5]. The k-means algorithm starts by randomly initialise k cluster centres, and then, the algorithm iteratively assigns data samples to the closest centres and updates the centres until some predefined termination condition is satisfied [98]. There are also other versions of k-means algorithms including online k-means [106], batch k-means [107], etc.

Another method that attempts to minimise the sum of squared errors is the k-medoids [108]. The k-medoids algorithm differs from the k-means in its representation of the different clusters. Each cluster is represented by the most centric data sample in the cluster instead of using the mathematical mean that may not belong to the cluster [108].

The centroid-based clustering approaches tend to work well with isolated and compact clusters, and are the most intuitive and frequently used methods. However, they also have the following drawbacks:

1) The number of clusters, which is k, needs to be defined in advance, which requires prior knowledge of the problem;

2) The optimisation process is very time-consuming and exhaustive, and the computation- and memory- efficiency of the approaches further deteriorates with the increase of the scale of the data.

2.3.1.3. Model-based Clustering

The model-based approaches attempt to optimise the fit between the given data and some mathematical models. Approaches of this kind not only identify the groups of data samples but also find characteristic descriptions for each group [98].

The most frequently used method is the self-organising map (SOM) [109], [110], which represents each cluster by a neuron. This algorithm constructs a single-layered network through a learning process with the “winner takes all” strategy.

SOM algorithm is a useful approach for clustering analysis and it can visualise the clustering results of high-dimensional data in 2D or 3D space. However, its performance is sensitive to the initial selection of weight vectors and the free parameters including learning rate, neighbourhood radius as well as the net size.

2.3.1.4. Density-based Clustering

Density-based clustering approaches assume that clusters exist in areas of higher density of the data space. Each cluster is characterised by a local mode or maximum of the density function [98].

One of the most popular density based clustering method is density-based spatial clustering of applications with noise (DBSCAN) [6]. The main idea of the DBSCAN algorithm is to group data samples that are very close together in the data space, and mark data samples that lie alone in low-density as outliers.

DBSCAN requires two parameters: the maximum radius of the neighbourhood and the minimum number of points required to form a dense region [6]. The algorithm starts with an arbitrary sample that has not been visited before. The neighbourhood of this starting sample is extracted and checked to see if this area contains a sufficient number of data samples. If so, a cluster is started, otherwise, it is labelled as noise. If a data sample is found to be a dense part of a cluster, its neighbouring area is also a part of this cluster, and, thus, all the data samples located in that area are added to the cluster. The process continues until the density-connected cluster is completely built. Then, an unvisited data sample is retrieved and processed to form a further cluster or be identified as noise.

There are a number of modified DBSCAN algorithms published including: ST-DBSCAN [111], ST-DBSCAN [112], P-DBSCAN [113], etc.

Mean shift algorithm [114]–[116] is also a popular density-based clustering approach built upon the concept of kernel density estimation (KDE). In statistics, KDE is a non-parametric way to estimate the PDF of a random variable. Mean shift algorithm implements the KDE idea by iteratively shifting each data sample to the densest area in its vicinity until all the data samples converge to local maxima of density.

eClustering algorithm [11] is the most popular online density-based clustering approach for streaming data processing, which can self-evolve its structure and update its parameters in a dynamic way. It is able to successfully handle the drifts and shifts of the data pattern in the data streams [117]. eClustering algorithm opens the door for the evolving clustering approaches and a number of modifications have been introduced later, i.e., evolving local mean clustering (ELMC) algorithm [12], data density-based clustering with automated radii (DDCAR) algorithm [10], clustering of evolving data streams (CEDS) algorithm [118]. The eClustering algorithm is also one of the theoretical bases of the self-organising transparent machine learning techniques described in this thesis in the later chapters.

The density-based clustering approaches can efficiently detect arbitrary-shaped clusters and do not require the number of clusters to be predefined. However, the main drawbacks of the density-based clustering approaches are as follows:

1) They require free parameters to be predefined, i.e. radius, window size, and if the free parameters are not set properly, the performance and efficiency of the algorithms are not guaranteed;

2) They usually assume the distribution model of the data, i.e. mixtures of Gaussians, which is often not the case in real problems.

2.3.1.5. Distribution-based Clustering

Distribution-based methods assume that the points that belong to each cluster are generated from a specific probability distribution, and the overall distribution of the data is assumed to be a mixture of several distributions. Thus, these approaches are closely related to statistics [98].

One prominent method is known as mixture models [119]–[123]. These approaches assume the generalisation model of data to be a mixture of Gaussian distributions. They randomly initialise a number of Gaussian distributions and iteratively optimise the parameters to fit the data model.

The distribution-based clustering is able to produce complex clustering results that can capture correlation and dependence between different features. However, there are clear drawbacks of these approaches:

1) The prior assumptions made by the distribution-based clustering approaches are too strong for real cases;

2) They require parameters to be set by users;

3) The computation- and memory- efficiency of these approaches are very low.

2.3.1.6. Fuzzy Clustering

Traditional clustering approaches generate partitioning, in which each data sample belongs to one and only one cluster. Thus, the clusters are disjointed. Fuzzy clustering extends this notion and suggests a soft clustering schema [98], which means a data sample can belong to different clusters at the same time.

The most representative fuzzy clustering approach is the well-known fuzzy c-means (FCM) algorithm [124]–[126]. FCM algorithm is based on the minimisation of the following equation:

$$F = \sum_{i=1}^{K} \sum_{j=1}^{C} \upsilon_{i,j}^{m} \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_j \right\|^2, \tag{2.50}$$

where C is the number of clusters; $\boldsymbol{\mu}_j$ is the j^{th} cluster centre (j=1,2,…, C); $\upsilon_{i,j}$ is the degree of membership of $\boldsymbol{x}_i$ in the j^{th} cluster; m is fuzzy partition matrix exponent for controlling the degree of fuzzy overlap. The general procedure of the fuzzy c-means algorithm is quite similar to the k-means approach.

In the FCM algorithm [124]–[126] and its later modifications [124], [127], [128], each cluster is a fuzzy set of all the patterns. Larger membership degrees suggest higher confidence in the assignment and, vice versa. A non-fuzzy clustering result can also be achieved by applying a threshold of the membership degrees.

The fuzzy clustering approaches are, generally, better than non-fuzzy centroid-based approaches in avoiding local maxima. However, the major drawbacks of the fuzzy clustering approaches are:

1) The number of clusters, which is "*c*", needs to be defined in advance, which requires prior knowledge of the problem;

2) The design of membership functions requires ad hoc decisions and prior knowledge of the problem as well.

2.3.2. Classification Algorithms

Classification is the task of assigning a class label to an input data sample. The class label indicates one of a given set of classes. In contrast with clustering, classification is usually considered as a supervised or semi-supervised learning technique [129]. In this section, the most widely used and representative fully supervised classification approaches are reviewed, which includes:

1) Naïve Bayes classifier;

2) K-nearest neighbour (KNN) classifier;

3) Support vector machine (SVM) classifier;

4) Decision tree (DT) classifier;

5) eClass classifier;

The popular semi-supervised classification approaches are also briefly reviewed.

2.3.2.1. Naïve Bayes Classifier

Naive Bayes classifier is one of the most widely studied and used classification approaches deeply rooted in the traditional probability theory and statistics [130]. The naïve Bayes classifier is based on the PDFs derived from training samples on the prior assumption that different features of the data are statistically independent.

Under this assumption, the conditional PDF for a data sample $\boldsymbol{x}$ belonging to the class $\mathbb{C}_i$ $(i = 1,2,\dots,C)$ is written as [130]:

$$P(\boldsymbol{x}|\mathbb{C}_i) = \prod_{j=1}^{M} P\left(x_j|\mathbb{C}_i\right), \tag{2.51}$$

and the class label of $\boldsymbol{x}$, denoted by $y(\boldsymbol{x})$, is given as:

$$\mathrm{y}(\boldsymbol{x}) = \underset{i=1,2,\dots C}{\operatorname{argmax}}\left(P(\boldsymbol{x}|\mathbb{C}_i)\right). \tag{2.52}$$

With certain types of PDFs, the naïve Bayes classifier can be trained very efficiently and it only requires a small number of training data for the training. However, the drawbacks of the naïve Bayes classifier are also obvious:

1) Its prior assumption, although simple, is often not held in real cases;

2) The choice of the PDFs requires prior knowledge and can influence the efficiency and accuracy of the classifier if it is not properly set;

3) Its model is over simple, which makes it insufficient in dealing with complex problems.

2.3.2.2. KNN Classifier

Nearest neighbour rule [131] is the simplest nonparametric decision procedure for deciding the label of an unlabelled data sample, $\boldsymbol{x}_o$. The mathematical expression of the nearest neighbour rule is as follows:

$$\mathrm{y}(\boldsymbol{x}_o) = \underset{x\in\{x\}_K}{\operatorname{argmin}}\left(d(\boldsymbol{x}_o,\boldsymbol{x})\right), \tag{2.53}$$

where $y(\boldsymbol{x}_o)$ is the estimated label of $\boldsymbol{x}_o$, which comes from the label of the data sample $\boldsymbol{x}_n \in \{\boldsymbol{x}\}_K$ that is closest to $\boldsymbol{x}_o$, $d(\boldsymbol{x}_o,\boldsymbol{x}_n) = \min_{\boldsymbol{x}\in\{\boldsymbol{x}\}_K}\left(d(\boldsymbol{x}_o,\boldsymbol{x})\right)$.

KNN algorithm [131]–[133] is the most representative algorithm employing the nearest neighbour rule directly. The algorithm is also among the simplest of all machine learning algorithms. The label of a particular data sample is decided by the labels of its k nearest neighbours based on the voting mechanism. The KNN algorithm mainly conducts computation during the classification stage.

KNN classifier has been widely used in different areas, i.e. biology [134], remote sensing [135], etc., and has a number of modifications being published [136]–[138]. KNN is also one of the two most widely used classifiers (the other one is the SVM classifier, which will be described in the next subsection) in the transfer learning approaches based on pre-trained DCNNs and is able to produce highly accurate classification results [77], [139]–[141]. However, the main drawbacks of the KNN classifier are:

1) The best choice of k is data dependent, which means that it requires prior knowledge to decide, otherwise, the performance of the KNN approach is not guaranteed;

2) KNN classifier is also sensitive to the structure of the data. Its performance severely deteriorates by the noisy, irrelevant features or unbalanced feature scales.

2.3.2.3. SVM Classifier

SVM [142] is one of the most popular classification approaches and has been widely used in various areas including biology [143], economy [144] and natural language processing [145]. SVM is also the other most widely used classifier in the transfer learning approaches based on pre-trained DCNNs and is able to produce highly accurate classification results [77], [139]–[141].

In essence, SVM is an algorithm for maximising a particular mathematical function with respect to a given collection of data [146]. There are four very important basic concepts within the SVM classifier [146].

1) Separating hyperplane;

Given a training set for a binary classification problem, denoted by $\{\boldsymbol{x}\}_K = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_K\}$ ($\boldsymbol{x}_i \in \mathbf{R}^M$) and the corresponding label, $\{y\}_K = \{y_1, y_2, \dots, y_K\}$ ($y_i \in \{-1, 1\}$). If the two classes are linearly separable, one can find some dimensional hyperplanes that separates the two classes [130], [146]. The points lie on the hyperplanes satisfying [147]:

$$\boldsymbol{x}^T\boldsymbol{w} + b = 0, \tag{2.54}$$

where $\boldsymbol{w}$ is a vector that is perpendicular to the separating hyperplane, namely, the normal.

2) Maximum-margin hyperplane;

As there exist many hyperplanes in the data space that can separate the data from two classes, the SVM selects the hyperplane with the maximum distance from it to the nearest data point on each side, which is also known as the maximum-margin hyperplane [130], [146], [148]. This is formulated as [147]:

$$\begin{array}{ll} \boldsymbol{x}_i^T\boldsymbol{w} + b \geq 1 & if\ y_i = 1 \\ \boldsymbol{x}_i^T\boldsymbol{w} + b \leq -1 & if\ y_i = -1 \end{array}. \tag{2.55}$$

One can find the maximum margin by minimising $\|\boldsymbol{w}\|$. The constrains (equation (2.55)) can be reformulated in Lagrangian expressions [147], which will be much easier to handle by constrains on the Lagrange multiples:

$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{K} \alpha_i y_i (\boldsymbol{x}_i^T \boldsymbol{w} + b) + \sum_{i=1}^{K} \alpha_i; \tag{2.56}$$

$$L_D = \sum_{i=1}^{K} \alpha_i - \frac{1}{2}\sum_{i=1}^{K} \sum_{j=1}^{K} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j, \tag{2.57}$$

where the subscript P stands for primal and D for dual; α_i $(i = 1,2,\dots,K)$ are the positive Lagrange multiples, one for each of the constrains (2.55). Equations (2.56) and (2.57) are raised from the same objective function but with different constrains, and the solution is found by minimising L_P or maximising L_D.

It has proven that selecting the maximum-margin hyperplane is the key to the success of the SVM as it maximises the SVM's ability to predict the correct classification of previously unseen examples [148].

3) Soft margin;

Although the data from two classes were assumed to be linearly separable in the previous derivation, data cannot be separated as cleanly in many cases in reality. In order to handle such cases, the SVM is modified by adding a soft margin, which can be done by introducing slack variables, denoted by ϑ_i $(i = 1,2,\dots,K)$, to the constrains (2.55) [147]:

$$\begin{matrix} \boldsymbol{x}_i^T \boldsymbol{w} + b \geq 1 - \vartheta_i & if\ y_i = 1 \\ \boldsymbol{x}_i^T \boldsymbol{w} + b \leq -1 + \vartheta_i & if\ y_i = -1 \\ \vartheta_i \geq 0\ \forall i & \end{matrix} . \tag{2.58}$$

With this modification, the SVM is able to deal with errors in the data by allowing a few anomalies to fall on the wrong side of the separating hyperplane. Essentially, this allows some data samples to push their way through the margin of the separating hyperplane without affecting the final result [146]. Meanwhile, it is necessary to limit the number of misclassifications of the SVM to prevent the deterioration of the performance, therefore, a user-specified parameter, denoted by ϱ, is introduced for controlling the training errors. A lager ϱ corresponds to a higher penalty to errors, and vice versa.

As a result, extra conditions are imposed on the solution for maximising dual Lagrangian L_D (equation (2.57)) [147]:

$$\begin{matrix} 0 \leq \alpha_i \leq \varrho \\ \sum_{i=1}^{K} \alpha_i y_i = 0 \end{matrix} . \tag{2.59}$$

For primal Lagrangian L_P (equation (2.56)), the formulation becomes [147]:

$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 + \varrho \sum_{i=1}^{K} \vartheta_i - \sum_{i=1}^{K} \alpha_i y_i (\boldsymbol{x}_i^T \boldsymbol{w} + b) - \sum_{i=1}^{K} \iota_i \vartheta_i, \tag{2.60}$$

where ι_i $(i = 1,2,\dots,K)$ are the positive Lagrange multiples introduced to enforce the positive of $\vartheta_i (i = 1,2,\dots,K)$.

4) Kernel function.

For a nonlinear separable case, it is impossible to find the maximum margin separating hyperplane, and the soft margin is not going to help as well. To solve this problem, the kernel function, which itself is a mathematical trick, is applied to the maximum margin hyperplanes. The kernel function provides a solution to the nonlinear separable problems by adding an additional dimension to the data and it projects the data from a low dimensional space to a higher dimensional space [146].

Some common kernels include [147]:

Polynomial: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j + \boldsymbol{1})^p$;

Gaussian radical basis function: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$ and

Hyperbolic tangent: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\kappa \boldsymbol{x}_i^T \boldsymbol{x}_j - \delta)$.

There is also a TEDA kernel recently introduced into the SVM [149].

Although the SVM classifier is one of the most widely used classifiers and is able to exhibit very good performance in various classification problems, there are still a few major drawbacks [130]:

1) It requires prior knowledge for choosing the kernel function. If the kernel function is not correctly chosen, it performance is not guaranteed;

2) Its computational efficiency drops quickly in large-scale problems;

3) It requires a full retraining if more training samples are provided later;

4) It is less efficient in handling multi-class classification problems.

2.3.2.4. Decision Tree Classifier

Decision tree classifier is a commonly used nonlinear classification approach [150]–[153] by mapping the input vectors in the data space to the output labels. The system is organised into a tree structure that each node represents an elementary classification algorithms and the leaves represent the output class labels. The system splits the entire data space into unique regions corresponding to the classes in a sequential manner [130], [154], [155]. An illustrative example of the decision tree is given in Figure 4[130].

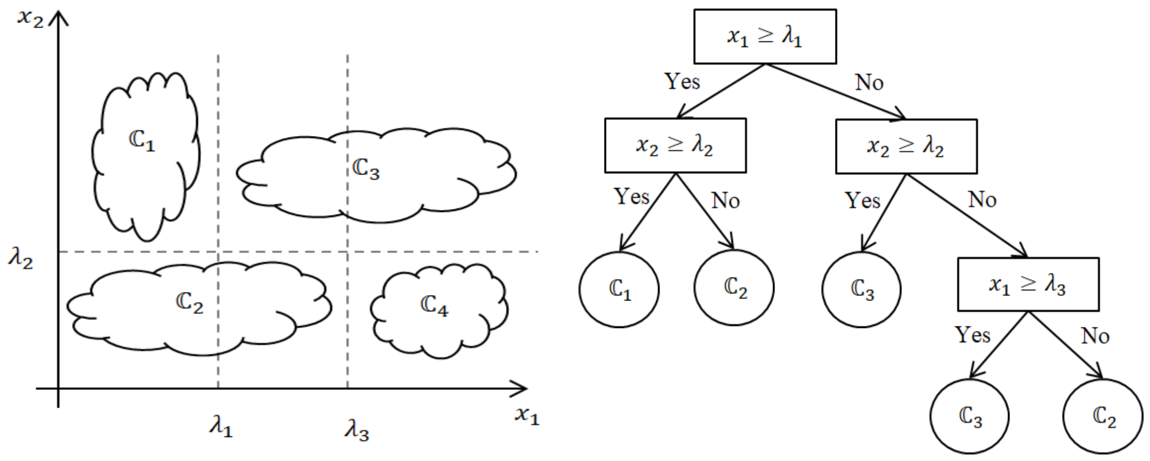


Figure 4. Example of a decision tree.

Initially, the decision tree approach is an expert-based approach [130], later, the recursive partitioning from the statistics makes it a data-driven approach [156]. Recursive partitioning creates a decision tree by splitting the training data into subsets based on several dichotomous independent variables. The process is termed recursive because each subset may be split for an indefinite number of times until a particular stopping criterion is reached.

The most popular approaches for learning a decision tree include:

1) Iterative Dichotomiser 3 (ID3) [157], which calculates the information entropy of the attributes to select the most appropriate one as the nodes of the tree;

2) C4.5 [158], which is an extension of ID3 algorithm with its open source Java implementation named as J48.

However, the major drawbacks of the decision tree approach are as follows:

1) It is less sufficient in dealing with complex problems;

2) Its performance relies heavily on the stopping criterion, which requires prior knowledge of the problems;

3) It has very high variance; a very small change in the dataset will lead to an entirely different tree structure.

2.3.2.5. eClass Classifier

As it was mentioned in the previous section that FRB systems were initially designed for expert-based systems, which seriously restricted the applications. After the effective idea of learning automation is introduced, evolving FRB systems started to be applied for the classification purposes with the implementation of automatic fuzzy rules generation [159]. The most successful and widely used evolving fuzzy classifier is the eClass [160].

eClass classifier [160] is a classification approach for streaming data processing. It is able to self-evolve its structure and update its meta-parameters on a sample-by-sample basis. There are a lot of evolving classifiers introduced on the basis of the eClass classifier [160] including: FLEXFIS-Class [161], simpl_eClass [162], autoClass [163], TEDAClass [51], etc.

eClass classifier [160] has two versions, the first one is eClass0, which uses the Zadeh-Mamdani-type fuzzy rules (equation (2.45)) and the other one is eClass1, which employs the Takagi-Sugeno type fuzzy rules (equation (2.46)).

The structure update mechanisms of the eClass0 and eClass1 are based on the local and global potentials (equations (2.61) and (2.62)) introduced within recursive density estimation (RDE) [62], respectively:

$$P_{K,i}^{L}(\boldsymbol{x}_K) = \frac{1}{1+\frac{\sum_{x\in\mathbb{C}_i} d(x_K,x)}{S_i}}; \tag{2.61}$$

$$P_K(\boldsymbol{x}_K) = \frac{1}{1+\frac{\sum_{i=1}^{K-1} d(x_K,x_i)}{K-1}}. \tag{2.62}$$

eClass0 uses eClustering algorithm [62] to partition the data into clusters, and adds new fuzzy rules to the rule base if

$$P_{K,i}^{L}(\boldsymbol{x}_K) > P_{K,i}^{L}(\boldsymbol{x}_i^*) \;\; \forall i = 1,2,\dots,C, \tag{2.63}$$

where $\boldsymbol{x}_i^*$ is the prototype of the i^{th} cluster.

While in eClass1, a new fuzzy rule is added to the rule base on condition that:

$$P_K(\boldsymbol{x}_K) > P_K(\boldsymbol{x}_i^*) \;\; \forall i = 1,2,\dots,C. \tag{2.64}$$

The difference between equations (2.63) and (2.64) is that the global potential is used in eClass1 and the local potential is used in eClass0.

For eClass0, the output ($y(\boldsymbol{x}_K)$, which is the label) for $\boldsymbol{x}_K$ with unknown data label is given following the “winner takes all” principle:

$$y(\boldsymbol{x}_K) = \text{argmax}_{i=1,2,\dots,C}\left(\lambda_{K,i}\right), \tag{2.65}$$

where $\lambda_i(\boldsymbol{x}_K)$ is the firing level of the *i*th fuzzy rule calculated by:

$$\lambda_{K,i} = \prod_{j=1}^{M} e^{-\frac{d^2(x_K, x_i^*)}{2r_{i,j}^2}}, \tag{2.66}$$

where $r_{i,j}$ is the spread of the *j*th fuzzy set of the *i*th fuzzy rule, which will be recursively updated during the learning stage [160].

For eClass1, the overall system output for $\boldsymbol{x}_K$ with unknown data label is given as the fuzzily weighted average:

$$y(\boldsymbol{x}_K) = \sum_{i=1}^{C} \frac{\lambda_{K,i}\bar{\beta}_{K,i}}{\sum_{j=1}^{C}\lambda_{K,j}}, \tag{2.67}$$

where $\bar{y}_{K,i}$ is the normalised output of the *i*th fuzzy rule defined as:

$$\bar{\beta}_{K,i} = \sum_{j=1}^{C} \frac{\beta_{K,i}}{\beta_{K,j}}, \tag{2.68}$$

and there is: $\beta_{K,i} = \bar{\boldsymbol{x}}_K^T \boldsymbol{a}_{K,i}$ ($\bar{\boldsymbol{x}}_K^T = [1, \boldsymbol{x}_K^T]$, $\boldsymbol{a}_{K,i} = \left[a_{K,i,0}, a_{K,i,1}, \dots, a_{K,i,M}\right]^T$). The label of $\boldsymbol{x}_K$ is chosen as: $y(\boldsymbol{x}_K) = \text{argmax}_{i=1,2,\dots,C}\left(\bar{\beta}_{K,i}\right)$.

The consequent parameter vector , $\boldsymbol{a}_{K,i}$ is updated using the fuzzily weighted recursive least squares (FWRLS) [11]:

$$\begin{aligned} \boldsymbol{a}_{K,i} &= \boldsymbol{a}_{K-1,i} + \boldsymbol{\Theta}_{K,i}\lambda_i\bar{\boldsymbol{x}}_K\left(y_K - \bar{\boldsymbol{x}}_K^T\boldsymbol{a}_{K-1,i}\right); & \boldsymbol{a}_{1,i} &= \boldsymbol{0}_{(M+1)\times 1} \\ \boldsymbol{\Theta}_{K,i} &= \boldsymbol{\Theta}_{K-1,i} - \frac{\lambda_i\boldsymbol{\Theta}_{K-1,i}\bar{\boldsymbol{x}}_K\bar{\boldsymbol{x}}_K^T\boldsymbol{\Theta}_{K-1,i}}{1+\lambda_i\bar{\boldsymbol{x}}_K^T\boldsymbol{\Theta}_{K-1,i}\bar{\boldsymbol{x}}_K}; & \boldsymbol{\Theta}_{1,i} &= \Omega\mathbf{I}_{(M+1)\times(M+1)} \end{aligned}, \tag{2.69}$$

where $\boldsymbol{\Theta}_{K,i}$ is the corresponding covariance matrix of the *i*th fuzzy rule; $\mathbf{I}_{(M+1)\times(M+1)}$ is the $(M+1)\times(M+1)$ dimensional identical matrix; Ω is a large constant.

2.3.2.6. Semi-Supervised Classifiers

Semi-supervised machine learning approaches [164]–[169] consider both the labelled and unlabelled data. The goal of the semi-supervised learning is to use the unlabelled data to improve the generalisation.

Cluster assumption states that the decision boundary should not cross high density regions, but lie in low density regions [166]. Virtually all the existing successful semi-supervised approaches rely on the cluster assumption in a direct or indirect way by estimating or optimising a smooth classification function over labelled and unlabelled data [170], [171].

Currently, there are two major branches of semi-supervised approaches, SVM-based and graph-based approaches [168], [172].

Semi-supervised SVMs [173]–[175] are extensions of the traditional SVMs [142] to a semi-supervised scenario. Traditional SVMs maximise the separation between classes based on the training data via a maximum-margin hyperplane [142], while semi-supervised classifiers balance the estimated maximum-margin hyperplane with a separation of all the data through the low-density regions. Well-known SVM-based semi-supervised classifiers include: transductive support vector machine (TSVM) [173], ∇TSVM [166], Laplacian SVM classifier [176], [177], local and global consistency based SVM [165] etc.

Graph-based approaches [168], [172], [178] use the labelled and unlabelled data as vertices in a graph and build pairwise edges between the vertices weighted by similarities. Well-known graph-based semi-supervised classifiers include: Gaussian fields and harmonic functions based approaches [164], AnchorGraph-based classifier [179] and greedy gradient max-cut based classifier [168], etc.

In general, both types of semi-supervised approaches share the same drawbacks [172]:

1) They are computationally expensive and they consume a lot of computer memory;

2) They are not applicable to the out-of-sample data;

3) They require full retraining when more training samples are given.

2.3.3. Regression Algorithms

Regression is a statistical process for estimating the relationships among variables and it is commonly used for prediction and forecasting in various areas including engineering [180], [181], biology [4], [182], economy and finance [183], [184]. The most widely used regression algorithm should be the linear regression [4]. Linear regression is a simple, linear, offline algorithm which has been studied rigorously, and used extensively in practical applications in the precomputer area of statistics [185], however, even now, it is still the predominant empirical tool in economics [4].

Adaptive-network-based fuzzy inference system (ANFIS) was introduced in 1993 [186] as a kind of artificial neural network that is based on Takagi-Sugeno fuzzy inference system. Since it integrates both neural networks and fuzzy logic principles, it has potential to capture the benefits of both in a single framework and has the learning capability to approximate nonlinear functions, and therefore, it is considered as a universal estimator.

Nowadays, due to the fact that we are faced not only with large datasets, but also with huge data streams, the traditional simple offline algorithms are not sufficient to meet the need. The more advanced evolving intelligent systems start to be developed and widely applied for the purpose of prediction [187]. The two most representative algorithms to learn evolving intelligent systems are the dynamic evolving neural-fuzzy inference system (DENFIS) [188] and evolving Takagi-Sugeno (ETS) fuzzy model [11], [189].

In this section, the four well-known algorithms, linear regression, ANFIS, DENFIS and ETS, are reviewed.

2.3.3.1. Linear Regression

A linear regression model assumes that the regression function is linear in the inputs. The linear model is one of the most important tools in the area of statistics. Linear models are quite simple, but can provide an adequate and interpretable description of the relationship between the inputs and outputs [4].

Using $\boldsymbol{x} = [x_1, x_2, \dots, x_M]^{\boldsymbol{T}}$ as the input vector, the linear model to predict the output y is expressed as:

$$y = a_0 + \sum_{i=1}^{M} x_i a_i = \overline{\boldsymbol{x}}^T \boldsymbol{a}, \tag{2.70}$$

where $\overline{\boldsymbol{x}}^T = [1, x_1, x_2, \dots, x_M]$ and $\boldsymbol{a} = [a_0, a_1, a_2, \dots, a_M]^T$.

A number of approaches can be used to decide the parameters $\boldsymbol{a}$ and fix the linear model, but by far, the most popular approach is the least square method [4], which is written as:

$$\widehat{\boldsymbol{a}} = (\overline{\boldsymbol{x}}^T \overline{\boldsymbol{x}})^{-\mathbf{1}} \overline{\boldsymbol{x}}^T y. \tag{2.71}$$

There are also many different modifications on the linear regression algorithm, one of the most representative one is the sliding window linear regression, which has been widely used in the finance and economy [190].

Despite the fact that linear regression model is one of the most popular regression algorithms due to its simplicity and stability, however, its major drawback is the oversimplification of the problems, which makes it insufficient in dealing with complex and large-scale problems.

2.3.3.2. ANFIS

ANFIS [186] is a simple data learning technique that uses fuzzy logic to transform given inputs into a desired output through interconnected neural network processing elements and information connections [191]. For clarity, the general architecture of ANFIS is presented in Figure 5, where two fuzzy rules of Takagi-Sugeno type are considered [59], [62]:

$$\begin{aligned} &IF\ (x_1\ is\ L_{1,1})\ AND\ (x_2\ is\ L_{1,2}) \quad THEN\ (y_1 = [1, \boldsymbol{x}^T]\boldsymbol{a}_1) \\ &IF\ (x_1\ is\ L_{2,1})\ AND\ (x_2\ is\ L_{2,2}) \quad THEN\ (y_2 = [1, \boldsymbol{x}^T]\boldsymbol{a}_2) \end{aligned}, \tag{2.72}$$

where $\boldsymbol{x} = [x_1, x_2]^T$, $\boldsymbol{a}_1 = [a_{1,0}, a_{1,1}, a_{1,2}]^T$ and , $\boldsymbol{a}_2 = [a_{2,0}, a_{2,1}, a_{2,2}]^T$.

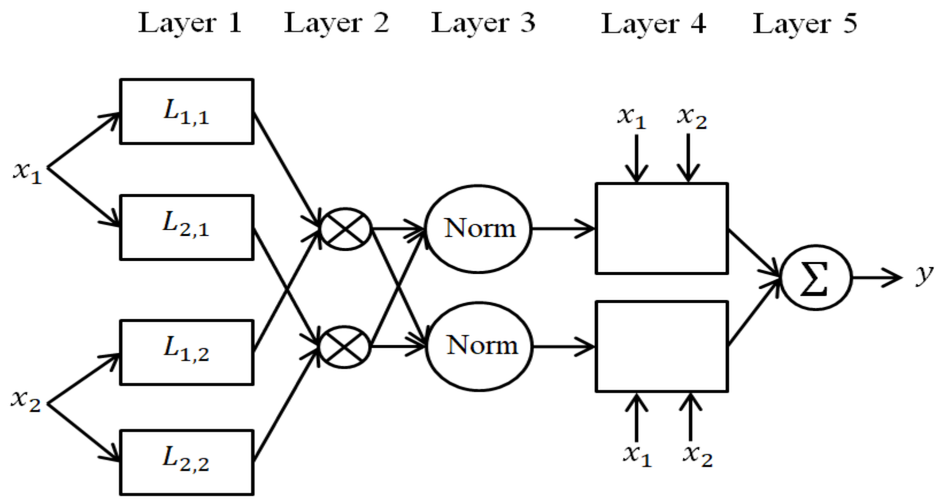


Figure 5. Architecture of the ANFIS.

Layer 1 consists of a number of adaptive nodes. The outputs of Layer 1 are the fuzzy membership grades of the inputs, and the membership function can be of any type. The outputs of this layer are denoted by $\text{MF}_{i,j}(x_j)$, in this case, $i, j = 1,2$.

Layer 2 involves fuzzy operators; it uses the multiply operator to fuzzify the inputs, and the outputs of this layer are denoted by $\lambda_i = \text{MF}_{i,1}(x_1) \cdot \text{MF}_{i,2}(x_2)$ $(i = 1,2)$.

Layer 3 plays a normalisation role to the firing strengths from the previous layer, $\bar{\lambda}_i = \lambda_i/(\lambda_1 + \lambda_2)$.

Layer 4 calculates the product of the normalised firing strength and a first order polynomial (for a first order Takagi-Sugeno model), $y_i = \bar{\lambda}_i[1, \boldsymbol{x}^T]\boldsymbol{a}_i$ $(i = 1,2)$.

Layer 5 performs the summation of all incoming signals. The overall output of the model is given by:

$$y = \sum_{i=1}^{2} y_i = \sum_{i=1}^{2} \bar{\lambda}_i [1, \boldsymbol{x}^T] \boldsymbol{a}_i = \sum_{i=1}^{2} \frac{\lambda_i [1, \boldsymbol{x}^T] \boldsymbol{a}_i}{\sum_{j=1}^{2} \lambda_{1j}}. \tag{2.73}$$

The training process of the ANFIS is a combination of gradient descent and least squares methods. In the forward pass, the outputs of the nodes within the network go forward until Layer 4 and the consequent parameters are determined by the least squares. In the backward pass, the error signals propagate backward and the premise parameters are updated using gradient descendent [191].

ANFIS was developed in the era that the datasets are static and not complicated. However, the ANFIS system is insufficient for the real applications nowadays due to the following drawbacks:

1) The structure of the fuzzy inference system needs to be predefined, which requires prior knowledge and a large number of ad hoc decisions;

2) Its structure is not self-evolving and its parameters cannot be updated online.

2.3.3.3. DENFIS

DENFIS is one of the two most widely used approaches for learning an evolving intelligent system [188]. DENFIS is able to generate a linear neural fuzzy model through an efficient adaptive online or offline learning process, and conduct accurate dynamic time series prediction.

Its online learning is achieved by the evolving clustering method (ECM), which, essentially, can be viewed as an online k-means algorithm with a mechanism of incrementally gaining new clusters [188]. Its offline learning process is also very similar to the k-means algorithm, which requires the number of clusters to be predetermined [188].

DENFIS [188], both online and offline models, uses Takagi-Sugeno type inference engine (equation (2.46)) [59], [62]. At each time moment, the output of DENFIS is calculated based on q-most activated fuzzy rules, which are dynamically chosen from a fuzzy rule set.

In both DENFIS online and offline models, all fuzzy membership functions are triangular type functions, which depend on three parameters as given by the following equation:

$$\mathrm{MF}(x) = \begin{cases} (x-a)/(b-a) & a < x \leq b \\ (c-x)/(c-b) & b < x \leq c \\ 0 & x \leq a \vee c < x \end{cases}, \tag{2.74}$$

The output of the DENFIS system is formulated as:

$$y(\boldsymbol{x}_K) = \sum_{i=1}^{q} \frac{\lambda_{K,i}\bar{\boldsymbol{x}}_K^T \boldsymbol{a}_{K,i}}{\sum_{j=1}^{q} \lambda_{K,j}}, \tag{2.75}$$

where $\lambda_{K,i} = \prod_{j=1}^{M} \mathrm{MF}_j\left(x_{K,j}\right)$; and the consequent parameters, $\boldsymbol{a}_{K,i}$, are updated using the weighted recursive linear least-square estimator (RLSE):

$$\begin{aligned} \boldsymbol{a}_{K,i} &= \boldsymbol{a}_{K-1,i} + \boldsymbol{\Theta}_{K,i}\lambda_i\bar{\boldsymbol{x}}_K\left(y_K - \bar{\boldsymbol{x}}_K^T\boldsymbol{a}_{K,i}\right) \\ \boldsymbol{\Theta}_{K,i} &= \frac{1}{\varepsilon}\left(\boldsymbol{\Theta}_{K-1,i} - \frac{\lambda_i\boldsymbol{\Theta}_{K-1,i}\bar{\boldsymbol{x}}_K\bar{\boldsymbol{x}}_K^T\boldsymbol{\Theta}_{K-1,i}}{\varepsilon+\bar{\boldsymbol{x}}_K^T\boldsymbol{\Theta}_{K-1,i}\bar{\boldsymbol{x}}_K}\right), \end{aligned} \tag{2.76}$$

where $\boldsymbol{a}_{K,i}$ and $\boldsymbol{\Theta}_{K,i}$ are initialised from the first few data samples; ε is a forgetting factor which typical value range is [0.8,1] [188].

Despite of being widely used, the major drawbacks of the DENFIS algorithm are:

1) It requires prior assumptions and predefined parameters, i.e. number of initial rules, parameters of the membership function;

2) As an online algorithm, it requires offline training and cannot start "from scratch".

2.3.3.4. ETS

The ETS system was firstly introduced in [192], [193] and ultimately in [11]. Nowadays, it is the other one of the two most widely used approaches for learning an evolving intelligent system. The learning mechanism of the ETS system [11] is computationally very efficient because it is fully recursive. The two phases include:

1) Data space partitioning and based on this, form and update the fuzzy rule-base structure;

2) Learning parameters of the consequent part of the fuzzy rules.

Data space partitioning is achieved by the eClustering algorithm [11] as presented in section 2.3.1.4. However, the data space partitioning within ETS serves for a different purpose compared to the eClustering. In ETS, there are outputs and the aim is to find such (perhaps overlapping) clustering of the input-output joint data space that fragments the input-output relationship into locally valid simpler (possibly linear) dependences. In eClustering, the aim is to cluster the input data space into a number of sub-regions. The consequent parameters of the ETS system are learned by the FWRLS, which has been described in section 2.4.5.

Due to its genetic nature, the ETS system has been widely applied to different problems including, but not limited to clustering, time series prediction, control.

2.3.4. Anomaly Detection Algorithms

Anomaly detection is an important problem of statistical analysis [194]. Anomaly detection techniques mainly target at discovering rare events [49]. In many real situations and applications, i.e. detecting criminal activities, forest fire, human body monitoring, etc., the rare cases play a key role. Anomaly detection is also closely linked to clustering process since the members of a cluster are rather routine, normal or typical [49] and, thus, data either belong to a cluster or are anomalous.

Traditional anomaly detection is based on statistical analysis [3], [195]. It relies on a number of prior assumptions about the data generation models and requires certain degree of prior knowledge [3]. However, these prior assumptions are only true in the ideal/theoretical situations, i.e. Gaussian, independently and identically distributed data, and the prior knowledge is more often unavailable in reality.

There are some supervised anomaly detection approaches published in the recent decades [152], [196], [197]. These techniques require the labels of the data samples to be known in advance, which allows the algorithms to learn in a supervised way and generate the desired output after training. The supervised approaches are usually more accurate and effective in detecting outliers compared with the statistical methods. However, in real applications, the labels of the data are usually unknown. The existing unsupervised anomaly detection approaches [198]–[200], however, require a number of user inputs to be predefined, i.e. threshold, error tolerance, number of nearest neighbours, etc. Selection of the proper user inputs requires good prior knowledge; otherwise, the performance of these approaches is affected.

2.4. Conclusion

This chapter contains the separate surveys for data analysis, computational intelligence and machine learning covering the scope of the research work presented in this thesis.

Traditional data analytics approach (the classical probability theory and statistics) provides the very solid mathematical foundation for the traditional data machine learning techniques. However, the very strong prior assumptions the probability theory and statistics rely on also open the door for many failures in real situations.

Traditional machine learning techniques suffer from various problems including 1) strong prior assumptions, 2) predefined user- and problem- specific parameters, 3) ad hoc decisions, etc., which undermine their applicability in large-scale, complex real problems.

Artificial neural networks (or the so-called deep learning) are the state-of-the-art approaches in the fields of machine learning and computer vision. However, their structures lack transparency, and they suffer from the problems of too many ad hoc decisions and very heavy computational burden, all of which hinder them in wider applications in real world.

On the other hand, traditional fuzzy rule-based classifiers were successfully used for classification [160], [201] offering transparent, interpretable structure, but could not reach the levels of performance achieved by deep learning classifiers. Their design also requires handcrafting membership functions, assumptions to be made and parameters to be selected.

The prototype-based nature of the recently introduced AnYa type fuzzy rules simplifies the antecedent (IF) part of the traditional fuzzy rule. Meanwhile, the new data analytics methodology, EDA, gives a strong alternative to the traditional statistics and probability theory, but is free from their paradoxes and deficiencies. Both of them provide a data-centred theoretical basis for the new generation of self-organising, transparent, nonparametric machine learning algorithms and deep learning networks, which will be presented in chapter 3, chapter 4 and chapter 5, respectively.

3. Unsupervised Self-Organising Machine Learning Algorithms

As the major unsupervised machine learning technique, clustering, alternatively, data partitioning plays a very important role in data analysis and pattern recognition. However, most of the data clustering and partitioning approaches, as described in section 2.2.1, share the three deficiencies:

1) They rely on strong prior assumptions on the model of data generation;

2) They require prior knowledge for defining free parameters;

3) Their performance and computational efficiency deteriorates very fast on large-scale and complex problems.

In principle, clustering and data partitioning are closely related and very similar, both of them aim to partition the data into smaller groups using certain types of algorithmic procedures. The only difference between clustering and data partitioning is that a data partitioning algorithm firstly identifies the data distribution peaks/modes and uses them as focal points [27] to associate other points with them to form data clouds [60] that resemble Voronoi tessellation [64]. Data clouds [60] can be generalised as a special type of clusters but with many distinctive differences. They are nonparametric and their shape is not predefined or predetermined by the type of the distance metric used. Data clouds directly represent the local ensemble properties of the observed data samples. In contrast, a clustering algorithm derives from data clusters with pre-determined shapes. The shape of clusters formed using Euclidean distance is always hyper-spherical; clusters formed using Mahalanobis distance are always hyper-ellipsoidal, etc.

In this chapter, the newly introduced self-organising data partitioning/clustering techniques within the EDA framework are presented. In contrast to the traditional clustering approaches, the techniques included in this chapter have the following features:

1) They employ the nonparametric EDA quantities as the operators to achieve data processing;

2) They are autonomous, self-organising and entirely data-driven;

3) They are free from user- and problem- specific parameters;

4) They are based on the ensemble properties and mutual distribution of empirically observed data.

This chapter is organised as follows. Section 3.1 introduces the autonomous data-driven clustering algorithm of three different versions (offline, evolving and parallel computing) as presented in [28]–[30]. Section 3.2 presents the offline and online versions of the hypercube-based data partitioning algorithm. The autonomous data partitioning algorithm and self-organising direction-aware data partitioning algorithm are given in section 3.3 and section 3.4, respectively. Section 3.5 summarises this chapter.

3.1. Autonomous Data-Driven Clustering Algorithm

The autonomous data-driven (ADD) clustering algorithm is a novel method based entirely on the empirical observations (the discrete data) and their ensemble properties (standardised eccentricity and unimodal density). It has three different versions, 1) offline version, 2) evolving version and 3) parallel computing online version.

3.1.1. Offline ADD Algorithm

The offline ADD clustering algorithm was initially introduced in [28]. As the computational efficiency of the original version is not high enough and is less effective in handling datasets within which data samples from different classes are not separable, in this section, the modified offline algorithm is presented. It has three stages: 1) preparation; 2) prototypes identification and 3) cluster fusion. The main procedure of the offline algorithm is described as follows.

3.1.1.1. Stage 1: Preparation

In this stage, for every unique data sample $\boldsymbol{u}_i \in \{\boldsymbol{u}\}_N$, $\{\boldsymbol{u}\}_N \subseteq \{\boldsymbol{x}\}_K$, its local unimodal density D_L is calculated:

$$D_L(\boldsymbol{u}_i) = \frac{\sum_{d(\boldsymbol{x},\boldsymbol{u}_i)\leq \bar{d}} q_L(\boldsymbol{x})}{2N_i q_L(\boldsymbol{u}_i)}, \tag{3.1}$$

where $q_L(\boldsymbol{x})$ is the cumulative proximity calculated locally for all the data samples located in the hypersphere with $\boldsymbol{u}_i$ as its centre and $\bar{d}$ as its radius; N_i is the number of data samples located within this hypersphere; $\bar{d}$ is the half of the average square distance between the data samples within $\{\boldsymbol{x}\}_K$ and is calculated as:

$$\bar{d}^2 = \frac{\sum_{j=1}^{K} q_K(\boldsymbol{x}_j)}{2K^2} = \frac{\sum_{j=1}^{K}\sum_{k=1}^{K} d^2(\boldsymbol{x}_j,\boldsymbol{x}_k)}{2K^2}. \tag{3.2}$$

3.1.1.2. Stage 2: Prototypes Identification

The clusters formation begins with the data sample with the maximum D_L:

$$\boldsymbol{u}_m = \text{argmax}_{i=1,2,\dots,N}(D_L(\boldsymbol{u}_i))\ . \tag{3.3}$$

Then, all the data samples within the hypersphere with $\boldsymbol{u}_m$ as the centre and $r = \frac{\bar{d}}{4}$ as the radius are found out as the initial member of the first cluster $\mathbb{C}_1$, and they are ranked based on their distances to $\boldsymbol{u}_m$ in an ascending order, which means: $d(\boldsymbol{z}_1, \boldsymbol{u}_m) = 0 \leq d(\boldsymbol{z}_2, \boldsymbol{u}_m) \leq \cdots \leq d\left(\boldsymbol{z}_{S_1}, \boldsymbol{u}_m\right)$ $(\boldsymbol{z}_i \in \mathbb{C}_1)$, and the number of members within $\mathbb{C}_1$ is denoted by S_1. The descending speed of D_L at $\mathbb{C}_1$ is calculated as:

$$D'_L(\boldsymbol{z}_i) = D_L(\boldsymbol{z}_1) - D_L(\boldsymbol{z}_i);\ \ i = 2,3 \dots, S_1. \tag{3.4}$$

The following condition is checked in regards to D'_L:

$$Condition\ 1: \begin{array}{c} IF\ \left(D'_L(\boldsymbol{z}_i) \leq \text{E}\left(D'_L(\boldsymbol{z})\right) + std\left(D'_L(\boldsymbol{z})\right), \forall i = 2,3,..,S_1\right) \\ THEN\ (D_L(\boldsymbol{z})\ decreases\ smoothly) \\ ELSE\ THEN\ (D_L(\boldsymbol{z})\ decreases\ sharply\ at\ a\ certain\ point) \end{array}\ . \tag{3.5}$$

If Condition 1 is met, it means that $\mathbb{C}_1$ is not fully spread yet and the radius of the hypersphere around $\boldsymbol{u}_m$ is enlarged to allow more data samples $(r \leftarrow r \times 1.1)$ to be included in $\mathbb{C}_1$. Then the process repeats until Condition 1 is unsatisfied.

Once Condition 1 is unsatisfied, it means that $D_L(\boldsymbol{z}_i)$ decreases sharply at the knee point, denoted by $\boldsymbol{u}_k$ (there may be multiple keen points as well). In such condition, the hypersphere around $\boldsymbol{u}_m$ includes data samples from two or more clusters, and $d(\boldsymbol{u}_k, \boldsymbol{u}_m)$ is the maximum radius of the hypersphere around $\boldsymbol{u}_m$ which includes data samples from the same cluster. By finding out all the data samples in $\{\boldsymbol{x}\}_K$ within the range of $d(\boldsymbol{u}_k, \boldsymbol{u}_m)$ around $\boldsymbol{u}_m$, $\mathbb{C}_1$ is fully formed: $\mathbb{C}_1 \leftarrow \{\boldsymbol{x} | d(\boldsymbol{x}, \boldsymbol{u}_m) \leq d(\boldsymbol{u}_k, \boldsymbol{u}_m), \boldsymbol{x} \in \{\boldsymbol{x}\}_K\}$. After the formation of $\mathbb{C}_1$, all its members are excluded from $\{\boldsymbol{u}\}_N$, $\{\boldsymbol{x}\}_K$ and the formation process starts again by finding out the next $\boldsymbol{u}_m$. The formation process will not stop until $\{\boldsymbol{u}\}_N = \emptyset$.

During the formation process, there may be some data samples spatially isolated from the majority, which means that $d(\boldsymbol{u}, \boldsymbol{u}_m) > \frac{\bar{d}}{4}$ $(\boldsymbol{u} \in \{\boldsymbol{u}\}_N\ and\ \boldsymbol{u} \neq \boldsymbol{u}_m)$, for this kind of $\boldsymbol{u}_m$, it forms a cluster by itself.

3.1.1.3. Stage 3: Cluster Fusion

As the previous stage may create too many subtle clusters, in this stage, the underlying overlapping clusters are merged together. The fusion operation starts from the cluster with the smallest support and ends up with the one with the largest support if no interruption.

Starting from the smallest cluster, $\mathbb{C}_1$, Condition 2 is checked, which also involves the Chebyshev inequality in the form of standardised eccentricity (equations (2.41) and (2.42)) [50]:

$$Condition\ 2: \begin{array}{c} IF\ \left(\varepsilon_{L,i}(\boldsymbol{\mu}_j) < \varepsilon_o\right) OR\ \left(\varepsilon_{L,j}(\boldsymbol{\mu}_i) < \varepsilon_o\right) \\ THEN\ \left(\mathbb{C}_i\ and\ \mathbb{C}_j\ are\ merged\ together\right) \end{array}, \tag{3.6}$$

where $i = 1,2,\ldots,C-1$ and $j = i+1, i+2, \ldots, C$; $\varepsilon_o = 5$, which corresponds to the 2σ rule; $\varepsilon_{L,i}(\boldsymbol{\mu}_j)$ and $\varepsilon_{L,j}(\boldsymbol{\mu}_i)$ are the standardised eccentricities calculated locally within the i^{th} and j^{th} clusters, respectively, and are expressed as ($i \neq j$):

$$\varepsilon_{L,i}(\boldsymbol{\mu}_j) = \frac{\sum_{\boldsymbol{x}\in\mathbb{C}_i} q_{L,i}(\boldsymbol{x}) + 2\sum_{\boldsymbol{x}\in\mathbb{C}_i} d^2(\boldsymbol{x},\boldsymbol{\mu}_j)}{2(S_i+1)\sum_{\boldsymbol{x}\in\mathbb{C}_i} d^2(\boldsymbol{x},\boldsymbol{\mu}_j)}. \tag{3.7}$$

Once two clusters, for instance, $\mathbb{C}_i$ and $\mathbb{C}_j$, are merged together ($\mathbb{C}_i \leftarrow \mathbb{C}_i + \mathbb{C}_j$), the centre and support of the new cluster are calculated. If $\mathbb{C}_i$ requires to be merged with multiple clusters, it is merged with the nearest one. Then, all the existing $C-1$ clusters are re-ranked in the descending order in terms of their supports and Condition 2 is checked again for another round.

After all the potentially overlapping clusters have been merged together, the remaining clusters are regarded as the main modes of the data pattern and the offline algorithm uses the clusters as the final output.

3.1.1.4. Complexity Analysis

In the first stage of the offline ADD clustering algorithm, the computational complexity of calculating $\bar{d}$ is $O(K^2)$. The computational complexity for calculating the local unimodal density, D_L is decided by the calculation of local cumulative proximity, q_L, and the computational complexity of which is: $\sum_{i=1}^{N} O(N_i)$. This is because that the distances between any two data samples have been calculated when $\bar{d}$ was calculated.

The second and third stages of the offline ADD algorithm mainly operate on the calculated local cumulative proximity, q_L and local unimodal density, D_L in the first stage, and, thus, the computational complexity of both stages is decided by the number of unique data samples, namely, $O(N)$.

Therefore, the overall computational complexity of the offline ADD algorithm is $O(K^2)$.

Since the complexity analysis of the proposed algorithms can be performed in a similar way as presented in this subsection, the computational complexity analysis for the rest of the algorithms presented in this thesis is not conducted. Nonetheless, one can use the same principles to get the conclusion.

3.1.1.5. Algorithm Summary

The main procedure of the offline ADD clustering algorithm is summarised in the form of a flowchart presented in Figure 6.

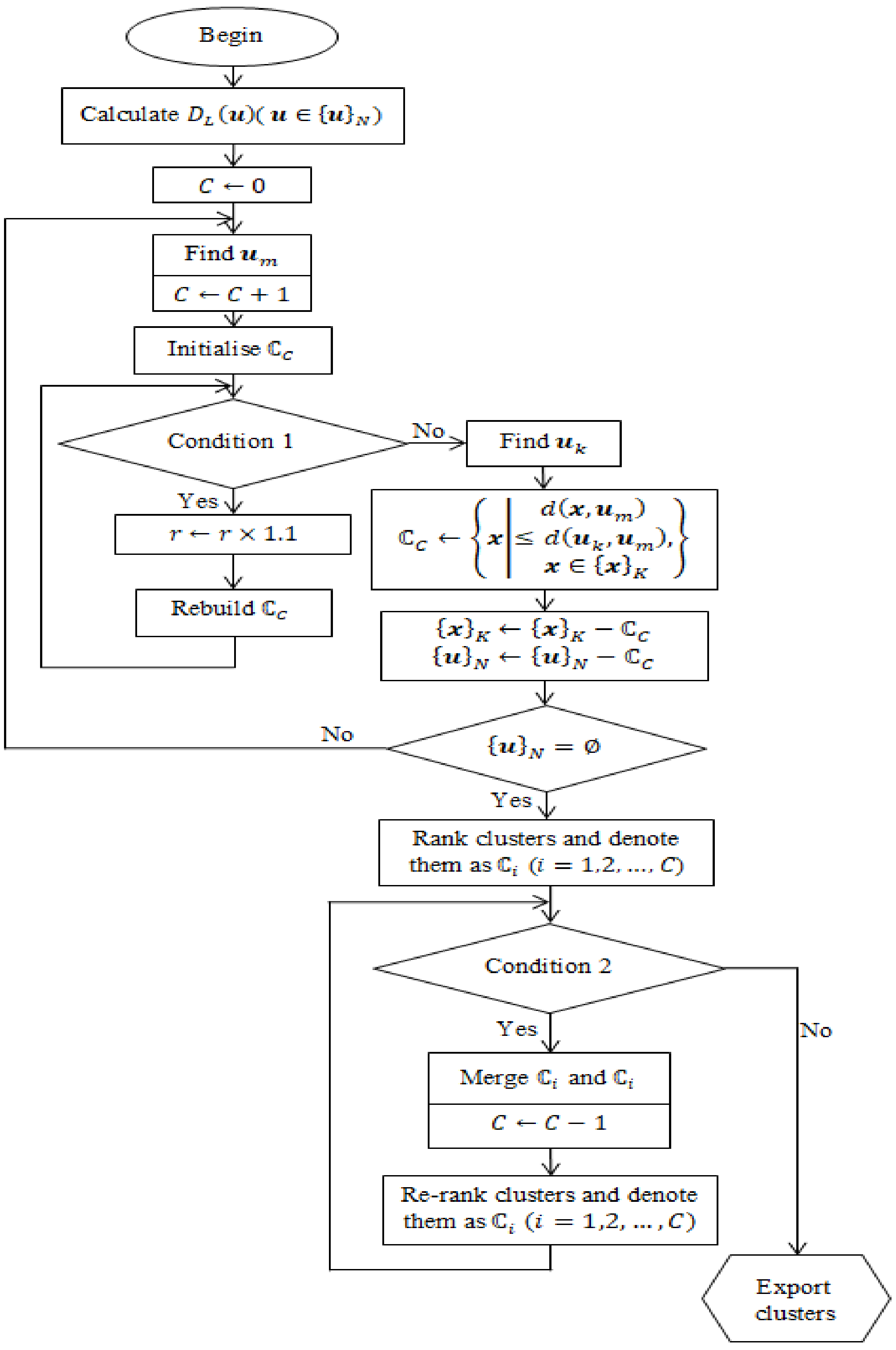

Begin
Calculate $D_L(\boldsymbol{u})(\boldsymbol{u} \in \{\boldsymbol{u}\}_N)$
$C \leftarrow 0$
Find $\boldsymbol{u}_m$
$C \leftarrow C + 1$
Initialise $\mathbb{C}_C$
Condition 1
No
Find $\boldsymbol{u}_k$
Yes
$r \leftarrow r \times 1.1$
Rebuild $\mathbb{C}_C$
$\mathbb{C}_C \leftarrow \left\{ \boldsymbol{x} \middle| \begin{array}{c} d(\boldsymbol{x}, \boldsymbol{u}_m) \\ \leq d(\boldsymbol{u}_k, \boldsymbol{u}_m), \\ \boldsymbol{x} \in \{\boldsymbol{x}\}_K \end{array} \right\}$
$\{\boldsymbol{x}\}_K \leftarrow \{\boldsymbol{x}\}_K - \mathbb{C}_C$
$\{\boldsymbol{u}\}_N \leftarrow \{\boldsymbol{u}\}_N - \mathbb{C}_C$
No
$\{\boldsymbol{u}\}_N = \emptyset$
Yes
Rank clusters and denote them as $\mathbb{C}_i$ $(i = 1,2, \dots, C)$
Condition 2
Yes
No
Merge $\mathbb{C}_i$ and $\mathbb{C}_j$
$C \leftarrow C - 1$
Re-rank clusters and denote them as $\mathbb{C}_i$ $(i = 1,2, \dots, C)$
Export clusters

Figure 6. Main procedure of the offline ADD clustering algorithm.

3.1.2. Evolving ADD Algorithm

In this section, the evolving version of the ADD clustering algorithm is described for the streaming data processing [29]. During the clustering process, there are only a few meta-parameters that have to be kept in memory and are recursively updated, which ensures the computation- and memory- efficiency of the evolving algorithm.

Because the recursive expressions of the algorithmic meta-parameters are involved in the evolving clustering algorithm, the most widely used Euclidean distance is used for simpler derivation and visual clarity, however, as it has been demonstrated in section 2.1.3, other types of distance/dissimilarity can also be considered.

The main stages are described as follows.

3.1.2.1. Stage 1: Initialisation

The evolving ADD clustering algorithm is initialised by the first data sample of the stream, $\boldsymbol{x}_1$. The global meta-parameters of the algorithm are set as:

$$K \leftarrow 1; \quad C \leftarrow 1; \quad \boldsymbol{\mu} \leftarrow \boldsymbol{x}_1; \quad X \leftarrow \|\boldsymbol{x}_1\|^2, \tag{3.8}$$

where K denotes the current time instance; C is the number of existing clusters; $\boldsymbol{\mu}$ and X are the global mean and average scalar product of the data stream $\{\boldsymbol{x}\}_K$.

The meta-parameters of the first cluster, $\mathbb{C}_1$, are set as:

$$\mathbb{C}_1 \leftarrow \{\boldsymbol{x}_1\}; \quad S_1 \leftarrow 1; \quad \boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_1; \quad X_1 \leftarrow \|\boldsymbol{x}_1\|^2, \tag{3.9}$$

3.1.2.2. Stage 2: Clusters Update

For each newly arrived data sample $\boldsymbol{x}_K$ ($K \leftarrow K+1$), the global meta-parameters ($\boldsymbol{\mu}$ and X) are firstly updated using equations (2.26) and (2.27).

Then the unimodal density is calculated at the centres of the existing clusters $\boldsymbol{\mu}_i$ ($i = 1,2,\dots,C$) and $\boldsymbol{x}_K$ using equations (2.18), (2.24) and (2.25). And Condition 3 is checked [11]:

$$Condition\ 3: \quad \begin{array}{c} IF\ \left(D_K(\boldsymbol{x}_K) < \min_{i=1,2,\dots,C}(D_K(\boldsymbol{\mu}_i))\right) \\ OR\ \left(D_K(\boldsymbol{x}_K) > \max_{i=1,2,\dots,C}(D_K(\boldsymbol{\mu}_i))\right). \\ THEN\ (\boldsymbol{x}_K\ creates\ a\ new\ cluster) \end{array} \tag{3.10}$$

If Condition 3 is satisfied, a new cluster is created around $\boldsymbol{x}_K$ ($C \leftarrow C+1$) because of the change of data pattern, and there are:

$$\mathbb{C}_C \leftarrow \{\boldsymbol{x}_K\} \quad S_C \leftarrow 1; \quad \boldsymbol{\mu}_C \leftarrow \boldsymbol{x}_K; \quad X_C \leftarrow \|\boldsymbol{x}_K\|^2. \tag{3.11}$$

Otherwise, $\boldsymbol{x}_K$ is assigned to the nearest cluster using equation (3.12):

$$nearest\ cluster\ = \operatorname{argmin}_{i=1,2,\dots,C}(\|\boldsymbol{x}_K - \boldsymbol{\mu}_i\|). \tag{3.12}$$

Assuming $\boldsymbol{x}_K$ is supposed to be assigned to the n^{th} cluster, $\mathbb{C}_n$. If the support $S_n = 1$, the parameters of $\mathbb{C}_n$ are updated as follows.

$$\mathbb{C}_n \leftarrow \{\mathbb{C}_n, \boldsymbol{x}_K\}; \quad S_n \leftarrow S_n + 1; \quad \boldsymbol{\mu}_n \leftarrow \frac{S_n - 1}{S_n}\boldsymbol{\mu}_n + \frac{1}{S_n}\boldsymbol{x}_K; \quad X_n \leftarrow \frac{S_n - 1}{S_n}X_n + \frac{1}{S_n}\|\boldsymbol{x}_K\|^2. \tag{3.13}$$

If $S_n > 1$, Condition 4 needs to be checked first:

$$Condition\ 4: \quad \begin{array}{c} IF\ (\|\boldsymbol{x}_K - \boldsymbol{\mu}_n\|^2 > r_n^2) \\ THEN\ (\boldsymbol{x}_K\ creates\ a\ new\ cluster) \end{array}, \tag{3.14}$$

where r_n is the radius of the n^{th} cluster and can be derived based on Chebyshev inequality, and r_n^2 is expressed as:

$$r_n^2 = 2(X_n - \|\boldsymbol{\mu}_n\|^2), \tag{3.15}$$

which indicates that the areas of influences are within $\sqrt{2}$ standard deviations around the centres of the clusters,

If Condition 4 is met, a new cluster is created around $\boldsymbol{x}_K$ ($C \leftarrow C + 1$) and the meta-parameters of $\mathbb{C}_C$ can be set using equations (3.11).

3.1.2.3. Stage 3: Clusters Adjustment

In this stage, all the existing clusters are ranked in terms of their radii in a descending order, and are still denoted as $\mathbb{C}_i$ ($i = 1,2,\ldots,C$), but there is $r_1^2 \geq r_2^2 \geq \cdots \geq r_C^2$. They will be examined and adjusted to avoid the possible overlap.

Condition 5 is checked first and the operation starts from the cluster with the largest radius and end with the one with the smallest radius if no interrupt:

$$Condition\ 5: \quad \begin{array}{c} IF\ \left(\left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_{j_1}\right\|^2 < \max\left(r_i^2, r_{j_1}^2\right)\right)\ AND\ \ldots \\ AND\ \left(\left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_{j_L}\right\|^2 < \max\left(r_i^2, r_{j_L}^2\right)\right) \\ THEN\ (\mathbb{C}_i\ is\ split) \end{array}, \tag{3.16}$$

where $i < \min_{l=1,2,\ldots,L}(j_k)$ and $L \geq 2$.

If Condition 5 is satisfied for cluster $\mathbb{C}_i$, it means that there are two or more other clusters sharing the same influence areas with it, thus, $\mathbb{C}_i$ needs to be split according to the following rule ($C \leftarrow C - 1$):

$$\begin{array}{cc} \mathbb{C}_{j_l} \leftarrow \{\mathbb{C}_{i_l}, \mathbb{C}_{j_l}\}; & S_{j_l} \leftarrow S_{j_l} + s_{i_l}; \\ \boldsymbol{\mu}_{j_l} \leftarrow \frac{S_{j_l} - s_{i_l}}{S_{j_l}}\boldsymbol{\mu}_{j_l} + \frac{s_{i_l}}{S_{j_l}}\boldsymbol{\mu}_i; & X_{j_l} \leftarrow \frac{S_{j_l} - s_{i_l}}{S_{j_l}}X_{j_l} + \frac{s_{i_l}}{S_{j_l}}X_i, \end{array} \tag{3.17}$$

where $l = 1,2,\dots,L$; $\mathbb{C}_i = \sum_{l=1}^{L} \mathbb{C}_{i_l}$; $s_{i_l} = round\left(\frac{\left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_{j_l}\right\|^{-1}}{\sum_{k=1}^{L}\left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_{j_k}\right\|^{-1}} S_i\right)$, $round(\cdot)$ denotes round to the nearest integer and there is: $S_i = \sum_{l=1}^{L} s_{i_l}$.

Once a cluster is split to the clusters nearby, the meta-parameters of the existing clusters are updated, they are re-ranked in terms of their radii again and the cluster split operation starts again from the largest cluster if no interrupt.

After there is no cluster satisfying Condition 5, Condition 6 is checked to see whether there are any clusters needed to merged:

$$Condition\ 6\text{:}\quad \begin{array}{c} IF\ \left(\left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|^2 < \max\left(r_i^2, r_j^2\right)\right) \\ THEN\ \left(\mathbb{C}_i\ and\ \mathbb{C}_j\ are\ merged\ together\right) \end{array}. \tag{3.18}$$

Once clusters $\mathbb{C}_i$ and $\mathbb{C}_j$ meet Condition 6, they are very close to each other and should be merged together as :

$$\begin{array}{ll} \mathbb{C}_i \leftarrow \{\mathbb{C}_i, \mathbb{C}_j\}; \quad C \leftarrow C - 1; & S_i \leftarrow S_i + S_j; \\ \boldsymbol{\mu}_i \leftarrow \frac{S_i - S_j}{S_i}\boldsymbol{\mu}_i + \frac{S_j}{S_i}\boldsymbol{\mu}_j; & X_i \leftarrow \frac{S_i - S_j}{S_i} X_i + \frac{S_j}{S_i} X_j. \end{array} \tag{3.19}$$

Similarly, the cluster merge operation starts with the largest cluster and every time a merge operation is performed, the remaining clusters are re-ranked based on their radii and the merge operation starts again until no cluster satisfying Condition 6.

Then, the algorithm goes back to Stage 2 if there are new data samples available or goes to Stage 4 to export the clusters.

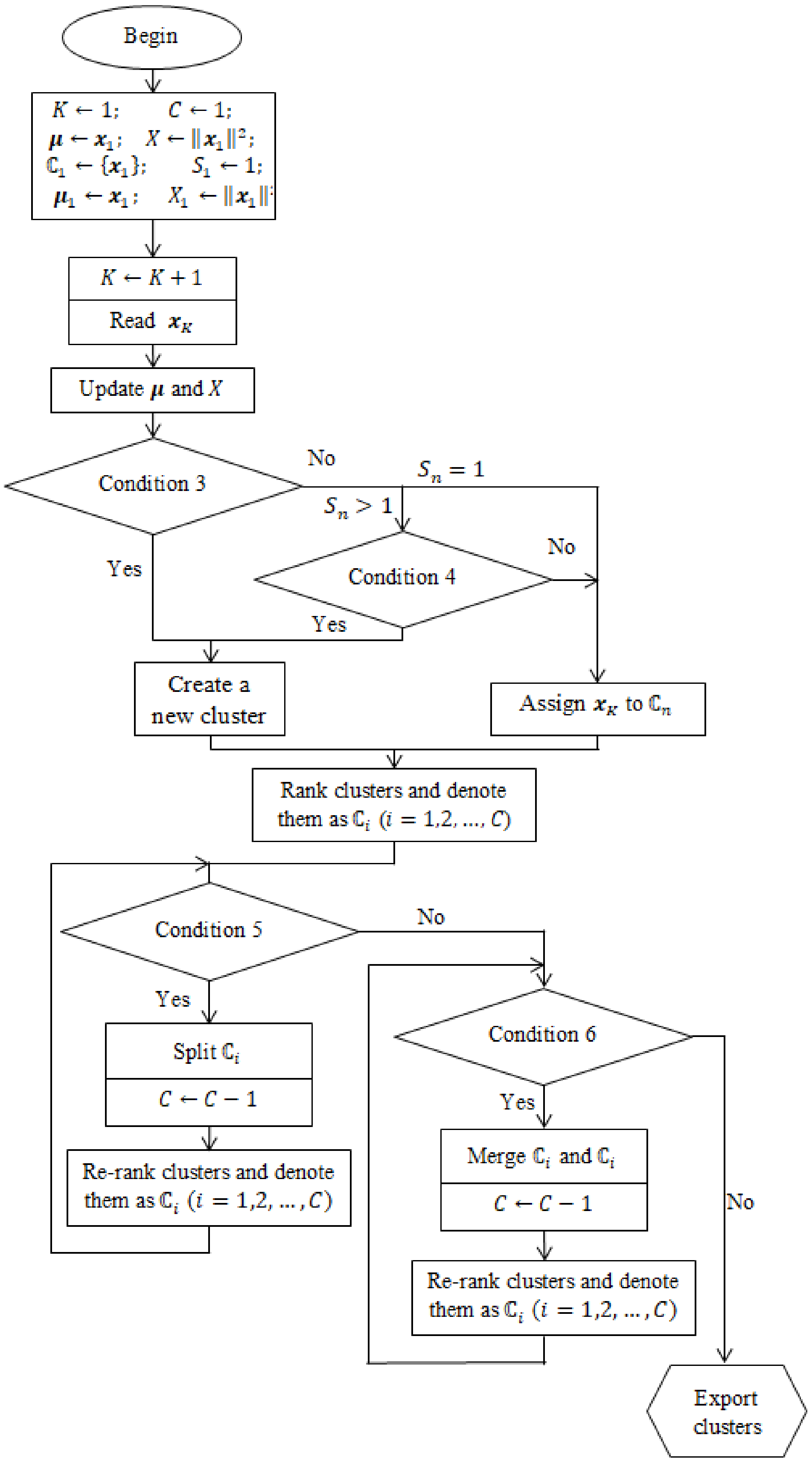
3.1.2.4. Stage 4: Exporting Main Clusters

In this stage, as there is no new data sample anymore, the evolving algorithm uses Condition 7 to filter out the clusters with small supports to get the more elegant output:

$$Condition\ 7\text{:}\quad IF\ \left(S_i > \frac{K}{2C}\right)\quad THEN\ (\ \mathbb{C}_i\ is\ one\ of\ the\ main\ clusters). \tag{3.20}$$

3.1.2.5. Algorithm Summary

The main procedure of the evolving ADD clustering algorithm is summarized in the form of a flowchart presented in Figure 7.

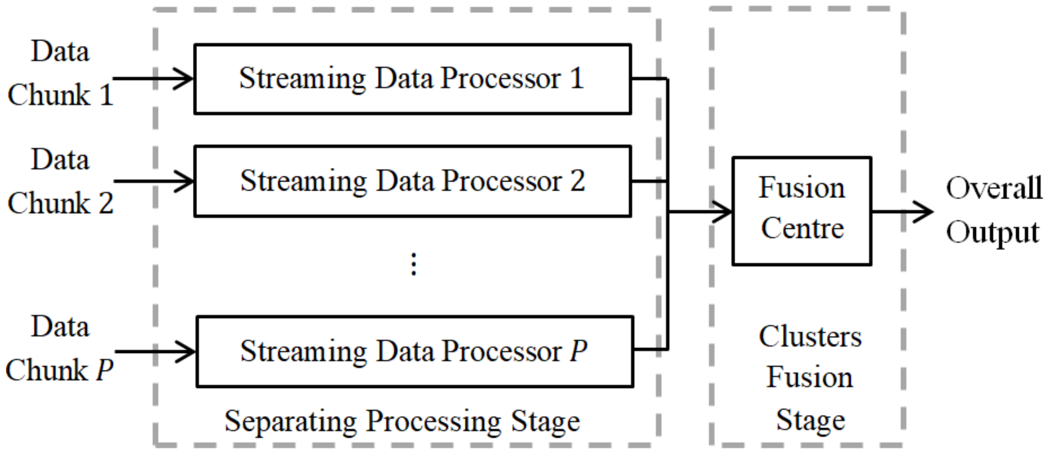


Begin
$K \leftarrow 1$; $C \leftarrow 1$;
$\boldsymbol{\mu} \leftarrow \boldsymbol{x}_1$; $X \leftarrow \|\boldsymbol{x}_1\|^2$;
$\mathbb{C}_1 \leftarrow \{\boldsymbol{x}_1\}$; $S_1 \leftarrow 1$;
$\boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_1$; $X_1 \leftarrow \|\boldsymbol{x}_1\|$
$K \leftarrow K + 1$
Read $\boldsymbol{x}_K$
Update $\boldsymbol{\mu}$ and X
Condition 3
No
$S_n = 1$
$S_n > 1$
Yes
Condition 4
No
Yes
Create a new cluster
Assign $\boldsymbol{x}_K$ to $\mathbb{C}_n$
Rank clusters and denote them as $\mathbb{C}_i$ $(i = 1,2, ..., C)$
Condition 5
No
Yes
Split $\mathbb{C}_i$
$C \leftarrow C - 1$
Re-rank clusters and denote them as $\mathbb{C}_i$ $(i = 1,2, \dots, C)$
Condition 6
Yes
Merge $\mathbb{C}_i$ and $\mathbb{C}_i$
$C \leftarrow C - 1$
Re-rank clusters and denote them as $\mathbb{C}_i$ $(i = 1,2, \dots, C)$
No
Export clusters

Figure 7. Main procedure of the evolving ADD clustering algorithm.

3.1.3. Parallel Computing ADD Algorithm

The parallel computing version of the ADD clustering algorithm was introduced for high frequency streaming data clustering [30]. Within this version, a number of streaming data processors are involved, which work on the chunks of the data stream and collaborate with each other efficiently to achieve parallel computation as well as a much higher processing speed. A fusion centre is involved to gather the key information from the processors and generate the overall output. The architecture of the parallel computing ADD clustering algorithm is depicted in Figure 8.



Data Chunk 1
Data Chunk 2
Data Chunk P
Streaming Data Processor 1
Streaming Data Processor 2
⋮
Streaming Data Processor P
Separating Processing Stage
Fusion Centre
Clusters Fusion Stage
Overall Output

Figure 8. Architecture of the parallel computing ADD clustering algorithm.

The main procedure of the parallel computing ADD clustering algorithm is as follows. As the recursive expressions of the algorithmic meta-parameters are used, Euclidean distance is used for illustration, however, other types of distance/dissimilarity can be considered as well.

3.1.3.1. Stage 1: Separate Processing

Assuming that there are P streaming data processors with the input chunk size of Q, and, at the current time instance, there are PQ data samples observed, the observed data samples are firstly separated into P different chunks according to the time instances at which they arrived:

$$\begin{gathered} \boldsymbol{chunk}\ 1=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\dots,\boldsymbol{x}_Q\} \\ \boldsymbol{chunk}\ 2=\{\boldsymbol{x}_{Q+1},\boldsymbol{x}_{Q+2},\dots,\boldsymbol{x}_{2Q}\} \\ \vdots \\ \boldsymbol{chunk}\ P=\{\boldsymbol{x}_{(P-1)Q+1},\boldsymbol{x}_{(P-1)Q+2},\dots,\boldsymbol{x}_{PQ}\} \end{gathered}, \tag{3.21}$$

After the data chunks are separated from the data stream, they are processed separately in the corresponding processors in the form of a tributary data stream on a sample-by-sample basis.

The *i*th streaming data processor ($i = 1,2,\ldots,P$) is initialised with the first data sample $\boldsymbol{x}_{(i-1)Q+1}$ and the meta-parameters of processor are set as:

$$I^i \leftarrow 1; \quad C^i \leftarrow 1, \tag{3.22}$$

where I^i denotes the number of data samples the *i*th processor has processed (time instance). The meta-parameter of the first cluster initialised by $\boldsymbol{x}_{(i-1)Q+1}$ are set as:

$$\mathbb{C}_1^i \leftarrow \{\boldsymbol{x}_{(i-1)Q+1}\}; \quad S_1^i \leftarrow 1; \quad \boldsymbol{\mu}_1^i \leftarrow \boldsymbol{x}_{(i-1)Q+1}; \quad X_1^i \leftarrow \left\|\boldsymbol{x}_{(i-1)Q+1}\right\|^2; \quad A_1^i \leftarrow 0, \tag{3.23}$$

where A_1^i is the age of the clusters [55], [60]. The age of a particular cluster (the *c*th one) is defined as follows [55], [60]:

$$A_c^i = I^i - \frac{\sum_{j=1}^{S_c^i} t_{c,j}^i}{S_c^i}, \tag{3.24}$$

where $t_{c,j}^i$ is time instance at which the *j*th member of the *c*th cluster is assigned.

For the next data sample ($I^i \leftarrow I^i + 1$), $\boldsymbol{x}_{(i-1)Q+I^i}$, Condition 7 is checked:

$$Condition\ 7: \quad \begin{array}{c} IF\ \left(\varepsilon_{L,j}^i\left(\boldsymbol{x}_{(i-1)Q+I^i}\right) \leq \varepsilon_o\right) \\ THEN\ \left(\boldsymbol{x}_{(i-1)Q+I^i}\ is\ associated\ with\ \mathbb{C}_j^i\right) \end{array}, \tag{3.25}$$

where $\varepsilon_o = 5$, which is the same as section 3.1.1.3; $\varepsilon_{L,j}^i\left(\boldsymbol{x}_{(i-1)Q+I^i}\right)$ is the local standardised eccentricity of $\boldsymbol{x}_{(i-1)Q+I^i}$ recursively calculated at the *c*th cluster as:

$$\varepsilon_{L,j}^i\left(\boldsymbol{x}_{(i-1)Q+I^i}\right) = \frac{\left(S_j^i\right)^2\left\|\boldsymbol{\mu}_j^i - \boldsymbol{x}_{(i-1)Q+I^i}\right\|^2 + \left(S_j^i+1\right)\left(S_j^i X_j^i + \left\|\boldsymbol{x}_{(i-1)Q+I^i}\right\|^2\right) - \left\|S_j^i\boldsymbol{\mu}_j^i + \boldsymbol{x}_{(i-1)Q+I^i}\right\|^2}{\left(S_j^i+1\right)\left(S_j^i X_j^i + \left\|\boldsymbol{x}_{(i-1)Q+I^i}\right\|^2\right) - \left\|S_j^i\boldsymbol{\mu}_j^i + \boldsymbol{x}_{(i-1)Q+I^i}\right\|^2}. \tag{3.26}$$

If $\boldsymbol{x}_{(i-1)Q+I^i}$ is associated with multiple clusters at the same time, it is assigned to the cluster based on the following rule:

$$\mathbb{C}_n^i \leftarrow \{\mathbb{C}_n^i, \boldsymbol{x}_{(i-1)Q+I^i}\}; \quad n = \operatorname{argmin}_{j=1,2,\ldots,C^i}\left(\varepsilon_{L,j}^i\left(\boldsymbol{x}_{(i-1)Q+I^i}\right)\right). \tag{3.27}$$

And the meta-parameters of $\mathbb{C}_n^i$ are updated as:

$$\begin{array}{cc} S_n^i \leftarrow S_n^i + 1; & \boldsymbol{\mu}_n^i \leftarrow \frac{S_n^i - 1}{S_n^i}\boldsymbol{\mu}_n^i + \frac{1}{S_n^i}\boldsymbol{x}_{(i-1)Q+I^i}; \\ X_n^i \leftarrow \frac{S_n^i - 1}{S_n^i}X_n^i + \frac{1}{S_n^i}\left\|\boldsymbol{x}_{x_{(i-1)Q+I^i}}\right\|^2; & A_n^i \leftarrow I^i - \frac{\left(S_n^i-1\right)\left(I^i-1-A_n^i\right)+I^i}{S_n^i}. \end{array} \tag{3.28}$$

If there is no cluster satisfying Condition 7, $\boldsymbol{x}_{(i-1)Q+I^i}$ creates a new cluster ($C^i \leftarrow C^i + 1$):

$$\mathbb{C}^i_{C^i} \leftarrow \{\boldsymbol{x}_{(i-1)Q+I^i}\}; \quad S^i_{C^i} \leftarrow 1; \quad \boldsymbol{\mu}^i_{C^i} \leftarrow \boldsymbol{x}_{(i-1)Q+I^i}; \quad X^i_{C^i} \leftarrow \left\|\boldsymbol{x}_{(i-1)Q+I^i}\right\|^2; \quad A^i_{C^i} \leftarrow 0. \tag{3.29}$$

For all the other clusters that do not receive new members, their meta-parameters stay the same except the ages:

$$A^i_k \leftarrow A^i_k + 1; \quad k \in other. \tag{3.30}$$

After the structure and the meta-parameters of the system are updated, before the processor begins to handle the next data sample ($I^i \leftarrow I^i + 1$), every cluster is checked to see whether it is out of date using Condition 8 ($k = 1,2,\dots,C^i$)

$$Condition\ 8\!: \quad \begin{matrix} IF\ \left(A^i_k > \mu^i_A + n\sigma^i_A\right)\ AND\ \left(A^i_k > Q\right) \\ THEN\ \left(\mathbb{C}^i_k\ is\ out\ of\ date\right) \end{matrix}, \tag{3.31}$$

where $n = 3$, which corresponds to the "3sigma" rule; μ^i_A and σ^i_A are the mean and standard deviation of the ages of all the existing clusters within the i^{th} processor.

Once the streaming data processor selects out a stale cluster, the cluster is removed automatically because it fails to represent the current data pattern and may have adverse influence on further clustering process [117]. After the cluster cleaning process, the processor will process the next data sample ($I^i \leftarrow I^i + 1$). Once the current chunk is processed, the processor will begin a new round of processing with the next data chunk on the basis of the previous clustering results stored in the memory.

3.1.3.2. Stage 2: Clusters Fusion

Although the P streaming data processors will continue the data processing process one chunk by one chunk automatically, based on the needs of the users, the overall clustering results can be viewed and checked at any time.

Responding to the request of the user, the clustering results of all the processors are passed to the fusion centre. The existing clusters from all the processors are re-denoted as $\mathbb{C}_1, \mathbb{C}_2, \dots, \mathbb{C}_{C^o}$, where $C^o = \sum_{i=1}^{P} C^i$, and there is $S_1 \leq S_2 \leq \cdots \leq S_{C^o}$. The centres and the average scalar products of the clusters are re-denoted as $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_{C^o}$ and $X_1, X_2, \dots, X_{C^o}$ correspondingly.

Each round of the cluster fusion operation starts with the cluster having the smallest support and end with the one with the largest support if no interrupt. The same process as presented in section 3.1.1.3 is performed to detect the two clusters that are required to be merged and equation (3.19) is used to fuse the meta-parameters of both clusters together.

After the clusters fusion process is finished, there may be some trivial clusters (with very small support) left, and they need to be assigned to nearest larger clusters based on Condition 9 to ensure an elegant output ($i = 1,2,\dots,C^o$):

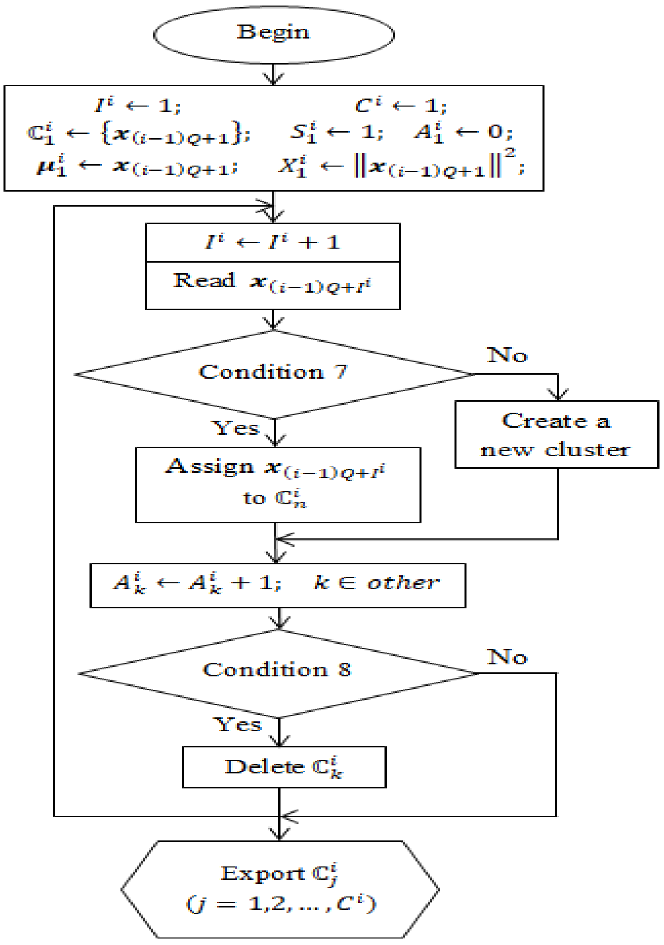
$$Condition\ 9:\quad IF\ \left(S_i \le \frac{\sum_{j=1}^{C^o} S_j}{5C^o}\right)\quad THEN\ (\ \mathbb{C}_i\ is\ a\ trivial\ cluster). \qquad (3.32)$$

And the nearest larger cluster is determined as:

$$\mathbb{C}_n \leftarrow \{\mathbb{C}_i, \mathbb{C}_n\};\quad n = \operatorname{argmin}_{j=1,2,\dots,C^o}\left(\left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|\right). \qquad (3.33)$$

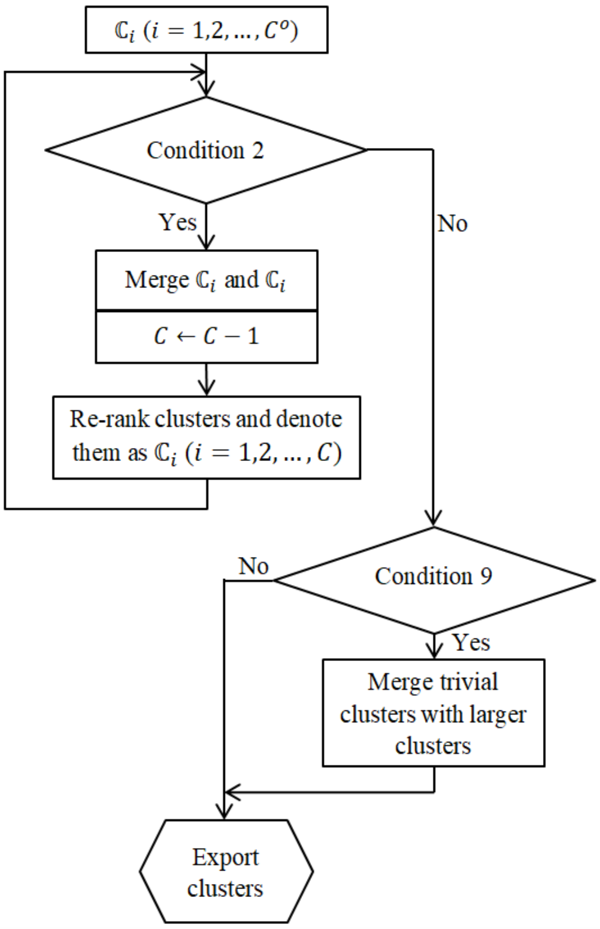
3.1.3.3. Algorithm Summary

The main procedure of the clustering process each streaming data processor performs is presented in the form of a flowchart presented in Figure 9. The fusion process is presented in Figure 10.



Begin
$I^i \leftarrow 1$; $C^i \leftarrow 1$;
$\mathbb{C}_1^i \leftarrow \{\boldsymbol{x}_{(i-1)Q+1}\}$; $S_1^i \leftarrow 1$; $A_1^i \leftarrow 0$;
$\boldsymbol{\mu}_1^i \leftarrow \boldsymbol{x}_{(i-1)Q+1}$; $X_1^i \leftarrow \|\boldsymbol{x}_{(i-1)Q+1}\|^2$;
$I^i \leftarrow I^i + 1$
Read $\boldsymbol{x}_{(i-1)Q+I^i}$
Condition 7
No
Yes
Create a new cluster
Assign $\boldsymbol{x}_{(i-1)Q+I^i}$ to $\mathbb{C}_n^i$
$A_k^i \leftarrow A_k^i + 1$; $k \in other$
Condition 8
No
Yes
Delete $\mathbb{C}_k^i$
Export $\mathbb{C}_j^i$ ($j = 1,2,\dots,C^i$)

Figure 9. Main procedure of the clustering process of the i^{th} streaming data processor



$\mathbb{C}_i \ (i = 1,2,\ldots,C^o)$
Condition 2
Yes
No
Merge $\mathbb{C}_i$ and $\mathbb{C}_i$
$C \leftarrow C - 1$
Re-rank clusters and denote them as $\mathbb{C}_i \ (i = 1,2,\ldots,C)$
Condition 9
No
Yes
Merge trivial clusters with larger clusters
Export clusters

Figure 10. Main procedure of the fusion process.

3.2. Hypercube-Based Data Partitioning Algorithm

The ADD clustering algorithm extracts all the needed information from the observed data directly and then continues to filter out the less important information from the main one, in addition, it has complicated operation mechanism, and thus, it is relatively slow. In this section, we will introduce an alternative algorithm for data partitioning, namely the hypercube-based.

This hypercube-based data partition (HCDP) algorithm involves a regular grid in the data space resulting in a number of hyper-cubes completely filling in the whole space, which simplifies the calculation and, thus, speed up the whole algorithm. The concept of hypercube is borrowed from [202] that a group of hyper-cubes perfectly divide the entire data space, $\mathbf{R}^M$. Within the proposed algorithm, every observed data sample will be projected into a hyper-cube, and the prototypes representing the local modes of the data pattern will be identified automatically.

The HCDP algorithm partitions the data into nonparametric, shape-free data clouds with the prototypes as the focal points attracting the data samples around them resembling Voronoi tessellation [64], which objectively represent the local modes of the data distribution. The proposed algorithm is deliberately designed to be memory-efficient based on the fact that the data samples normally will not be distributed everywhere in the data space.

In the following two subsections, the two versions (offline and evolving) of the hypercube-based partitioning algorithm are introduced separately.

3.2.1. Offline HCDP Algorithm

The offline HCDP algorithm involves the multimodal density, D^G, to identify the focal points of the data clouds in the data space, $\mathbf{R}^M$. The identification is conducted with the hyper-cubes and only involves the unique data sample set $\{\boldsymbol{u}\}_N$ and the corresponding frequencies of occurrence $\{f\}_N$.

The main stages of the proposed offline hypercube-based partitioning algorithm are as follows.

3.2.1.1. Stage 1: Hyper-cubes projection

Firstly, the multimodal densities $D_K^G(\boldsymbol{u}_i)$ $(\boldsymbol{u}_i \in \{\boldsymbol{u}\}_N)$ at all the data samples are calculated using equation (2.19). Then, all the unique data samples $\{\boldsymbol{u}\}_N$ are normalised into the range between [0,1], re-denoted as $\{\boldsymbol{v}\}_N$, and the whole data space is converted into a M dimensional hypercube with the value range of [0,1] in each dimension.

Then, $\{\boldsymbol{v}\}_N$ are projected into the γ^M smaller hyper-cubes that separate the data space, where γ is the granularity of the segmentation, and γ is a positive integer. The following equation is used to find the hypercube for a particular unique data sample $\boldsymbol{v}_i$ belonging to:

$$\boldsymbol{m}_i = [m_{i,1}, m_{i,2}, \dots, m_{i,M}]^T \ , \tag{3.34}$$

where $\boldsymbol{m}_i$ indicates the coordinate of the hypercube in the data space; $m_{i,d} = \text{argmin}_{m=1,2,\dots,\gamma}\left(\left|v_{i,d} - \frac{m-1}{\gamma}\right| + \left|v_{i,d} - \frac{m}{\gamma}\right|\right)$ and $d = 1,2,\dots,M$. Based on $\{\boldsymbol{m}\}_N$, one can find out the corresponding data samples in each hypercube.

Assuming that there are H hyper-cubes are actually occupied by at least one data sample ($C \le \gamma^M$), denoted by $\boldsymbol{\mathcal{H}}_i$ $(i = 1,2,\dots,H)$, one can count the support of each hypercube (S_i) :

$$S_i = \sum\nolimits_{\boldsymbol{u}_j \in \boldsymbol{\mathcal{H}}_i} f_j, \tag{3.35}$$

and calculate the sum of multimodal densities of data samples within it:

$$D^G(\boldsymbol{\mathcal{H}}_i) = \sum_{\boldsymbol{u}_j \in \boldsymbol{\mathcal{H}}_i} D_K^G(\boldsymbol{u}_j). \tag{3.36}$$

3.2.1.2. Stage2: Data Clouds Formation

For each hypercube, $D^G(\boldsymbol{\mathcal{H}}_i)$ $(i = 1,2,\dots,H)$ is compared with the same value for other cubes directly connected to it, which are sharing the same edge or point. If $D^G(\boldsymbol{\mathcal{H}}_i)$ is the local maximum, then $\boldsymbol{\mathcal{H}}_i$ is a hypercube that represents one of the local modes of the data pattern, and the collection of such hyper-cubes are denoted as $\{\boldsymbol{\mathcal{H}}^*\}$. If there is no occupied hypercube around $\boldsymbol{\mathcal{H}}_i$, $\boldsymbol{\mathcal{H}}_i$ can also be viewed as a local mode, and thus, there is $\{\boldsymbol{\mathcal{H}}^*\} \leftarrow \{\boldsymbol{\mathcal{H}}^*, \boldsymbol{\mathcal{H}}_i\}$.

However, it is also necessary to filter out the hyper-cubes with smaller supports in $\{\boldsymbol{\mathcal{H}}^*\}$ using Condition 10 because they may actually stand for anomalies $(i = 1,2,\dots,H^*)$:

$$Condition\ 10\text{:}\quad \begin{matrix} IF\ (S_i^* < mean(S^*) - 2 \times std(S^*)) \\ THEN\ (\boldsymbol{\mathcal{H}}_i^*\ is\ removed) \end{matrix}, \tag{3.37}$$

where H^* is the number of hyper-cubes within $\{\boldsymbol{\mathcal{H}}^*\}$; S_i^* is the corresponding support of $\boldsymbol{\mathcal{H}}_i^*$; $mean(S^*)$ and $std(S^*)$ are the average value and standard deviation of S_i^* $(i = 1,2,\dots,H^*)$.

After the filtering operation (equation (3.37)), the focal points of the data clouds can be selected from the remaining hyper-cubes directly as the unique data samples with the highest value of D^G in each hypercube. The focal points are re-denoted as $\{\boldsymbol{u}^*\}$ and based on them, the members of all the data clouds can be selected from $\{\boldsymbol{x}\}_K$ using equation (3.12).

3.2.1.3. Algorithm Summary

The main procedure of the offline HCDP algorithm is summarised in the form of a flowchart presented in Figure 11.

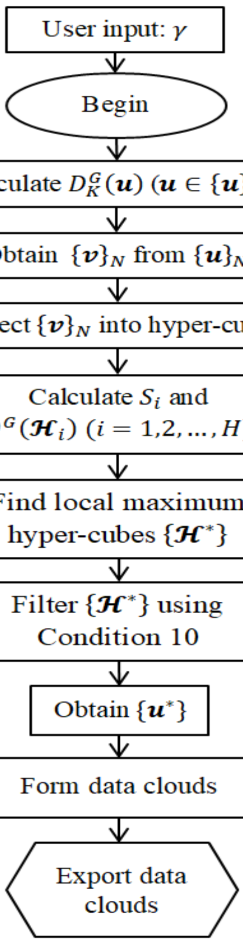

User input: γ
Begin
Calculate $D_K^G(\boldsymbol{u})$ $(\boldsymbol{u} \in \{\boldsymbol{u}\}_N)$
Obtain $\{\boldsymbol{v}\}_N$ from $\{\boldsymbol{u}\}_N$
Project $\{\boldsymbol{v}\}_N$ into hyper-cubes
Calculate S_i and $D^G(\boldsymbol{\mathcal{H}}_i)$ $(i = 1,2, \dots, H)$
Find local maximum hyper-cubes $\{\boldsymbol{\mathcal{H}}^*\}$
Filter $\{\boldsymbol{\mathcal{H}}^*\}$ using Condition 10
Obtain $\{\boldsymbol{u}^*\}$
Form data clouds
Export data clouds

Figure 11. Main procedure of the offline HCDP algorithm.

3.2.2. Evolving HCDP Algorithm

In this subsection, the main procedure of the evolving hypercube-based partitioning algorithm is described.

3.2.2.1. Stage 1: Online Hypercube Projection

The evolving hypercube-based partitioning algorithm requires to partition the data space into γ^M hyper-cubes. However, without knowing the exact value ranges of the attributes of the streaming data, a direct partitioning of the data space is infeasible. Therefore, data online standardisation is necessary for confining the value ranges of the data. For each newly arrived data sample, $\boldsymbol{x}_K$ $(K \leftarrow K + 1)$, the standard deviation of each attribute value is updated as $(d = 1,2, \dots, M)$:

$$\sigma_d^2 \leftarrow \frac{K-1}{K}(\sigma_d^2 + \mu_d^2) + \frac{1}{K}x_{K,d}^2 - \left(\frac{K-1}{K}\mu_d + \frac{1}{K}x_{K,d}\right)^2, \tag{3.38}$$

Then, the global mean $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_M]^T$ is also updated using equation (2.26) and $\boldsymbol{x}_K$ is standardised online as $(d = 1,2, \dots, M)$:

$$v_{K,d} = (x_{K,d} - \mu_d)/\sigma_d \ . \tag{3.39}$$

Using the "3σ" Chebyshev inequality, the data space is manually converted into a M dimensional hypercube with the value range of $[-3, 3]$ in each dimension. This covers the majority of the observed data samples [52]. For the data samples jumping out of the limitation, they are rolled back to the edges of this huge hypercube.

Then, the same approach as described by section 3.2.1.1 (equation (3.34)) is used to find the small hyper-cube with the coordinate $\boldsymbol{m}_K$ that $\boldsymbol{v}_K$ belongs to, and the only difference is that ($d = 1,2, \dots, M$):

$$m_{K,d} = \text{argmin}_{m=1,2,\dots,\gamma}\left(\left|v_{i,d} - n - 2n \times \frac{m-1}{\gamma}\right| + \left|v_{i,d} - n - 2n \times \frac{m}{\gamma}\right|\right), \tag{3.40}$$

where $n = 3$, which corresponds to the "3σ" rule.

Based on the coordinate $\boldsymbol{m}_K$, the hypercube, assuming $\boldsymbol{\mathcal{H}}_n$, that $\boldsymbol{v}_K$ is associated with is identified, and its meta-parameters, namely the number S_n and mean $\boldsymbol{\mu}_n$ of the current data samples within $\boldsymbol{\mathcal{H}}_i$ are updated.

3.2.2.2. Stage 2: Data Clouds Formation

Once there is no new data sample any more, the evolving hypercube-based data partitioning algorithm will perform the focal points identification operation and then generate the final data partitioning output. The algorithm is designed to work automatically and will perform the focal points identification operation anyway unless specifically prompted not to.

Once the focal points identification operation begins, the multimodal densities at the centres of the activated hyper-cubes ($S > 0$) are calculated using equations (2.19), (2.24) and (2.25) with the corresponding supports used as the frequencies ($i = 1,2, \dots, H$):

$$D_K^G(\boldsymbol{\mu}_i) = S_i \Big/ \left(1 + \frac{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}\|^2}{X - \|\boldsymbol{\mu}\|^2}\right). \tag{3.41}$$

Then, by using the multimodal densities $D_K^G(\boldsymbol{\mu}_i)$ as $D^G(\boldsymbol{\mathcal{H}}_i)$, the same process as described in section 3.2.1.2 is applied to identify the focal points from the centres $\boldsymbol{\mu}_i (i = 1,2, \dots, H)$ of the activated hyper-cubes. The selected focal points are re-denoted as $\{\boldsymbol{\mu}^*\}$, and based on them, the corresponding members of all the data clouds can be obtained from $\{\boldsymbol{x}\}_K$ using equation (3.12).

3.2.2.3. Algorithm Summary

The main procedure of the evolving HCDP algorithm is summarised in the form of a flowchart presented in Figure 12.

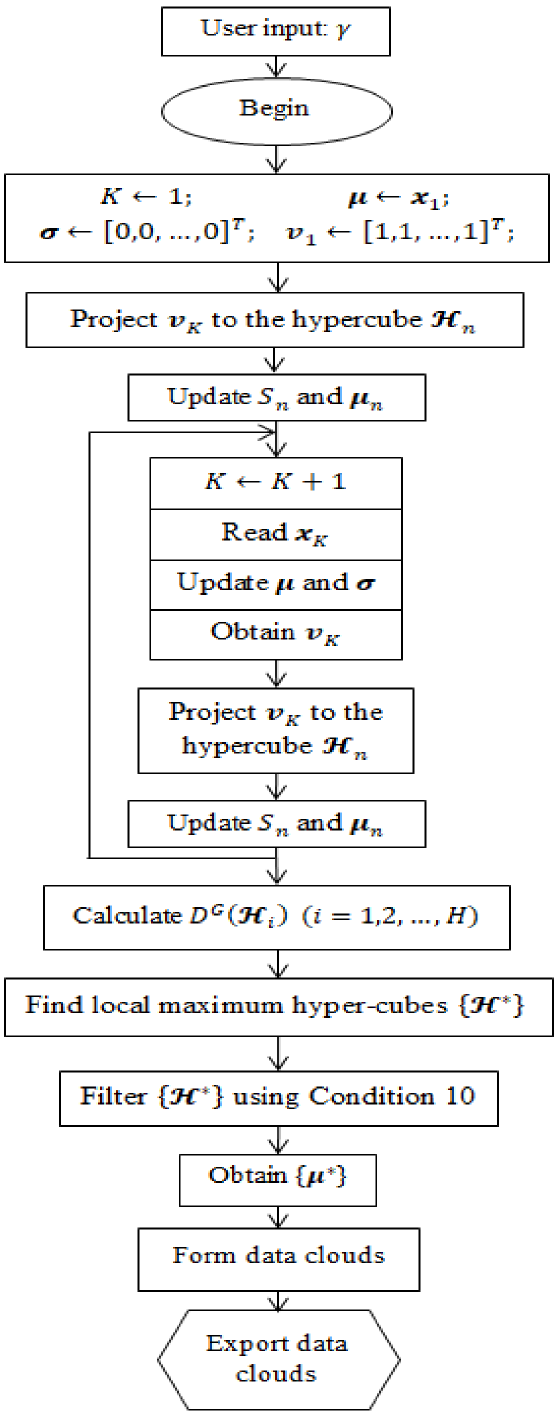

User input: γ
Begin
$K \leftarrow 1$; $\boldsymbol{\mu} \leftarrow \boldsymbol{x}_1$;
$\boldsymbol{\sigma} \leftarrow [0,0,\ldots,0]^T$; $\boldsymbol{v}_1 \leftarrow [1,1,\ldots,1]^T$;
Project $\boldsymbol{v}_K$ to the hypercube $\mathcal{H}_n$
Update S_n and $\boldsymbol{\mu}_n$
$K \leftarrow K+1$
Read $\boldsymbol{x}_K$
Update $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$
Obtain $\boldsymbol{v}_K$
Project $\boldsymbol{v}_K$ to the hypercube $\mathcal{H}_n$
Update S_n and $\boldsymbol{\mu}_n$
Calculate $D^G(\mathcal{H}_i)$ $(i=1,2,\ldots,H)$
Find local maximum hyper-cubes $\{\mathcal{H}^*\}$
Filter $\{\mathcal{H}^*\}$ using Condition 10
Obtain $\{\boldsymbol{\mu}^*\}$
Form data clouds
Export data clouds

Figure 12. Main procedure of the evolving HCDP algorithm.

3.3. Autonomous Data Partitioning Algorithm

The autonomous data partitioning (ADP) algorithm is an advanced data driven approach for data partitioning [31]. The ADP algorithm has the following advantages:

1) Its operation mechanism is simpler, which makes it more computationally efficient and easier for implementation compared with other clustering/data partitioning algorithms presented in this thesis (one can see from section 6.1 and also from [28]–[33]);

2) It is free from user inputs, prior assumptions and predefined problem- and user-specific parameters;

3) It partitions the data into nonparametric, shape-free data clouds, which objectively represent the local modes of the data distribution.

ADP algorithm has two versions, offline and evolving.

3.3.1. Offline ADP Algorithm

The offline ADP algorithm works with the multimodal density, D^G of the observed data samples and it is based on the ranks of them in terms of multimodal densities and mutual distribution [31]. Ranks are very important, but other approaches avoid them because they are nonlinear and discrete operators. And thus, the offline version is more stable and effective in partitioning static datasets.

The main procedure of the offline ADP algorithm consists of four stages as follows [31].

3.3.1.1. Stage 1: Ranking Order Data

The ADP algorithm starts by organising the unique data samples $\{\boldsymbol{u}\}_N$ in an indexing list, denoted by $\{\boldsymbol{z}\}_N$, based on the distance to the global peak of multimodal density.

Firstly, the multimodal densities D^G of all observed unique data samples $\{\boldsymbol{u}\}_N$ are calculated using equation (2.19). The unique data sample with the highest multimodal density is then selected as the first element of $\{\boldsymbol{z}\}_N$:

$$\boldsymbol{z}_1 \leftarrow \boldsymbol{u}_j, j = \operatorname{argmax}_{k=1,2,\dots,N}\left(D_K^G(\boldsymbol{u}_k)\right), \tag{3.42}$$

where $\boldsymbol{z}_1$ is the unique data sample with the global maximum multimodal density. After, $\boldsymbol{z}_1$ is identified, it is set as the reference sample $(\boldsymbol{z}_r \leftarrow \boldsymbol{z}_1)$ and $\boldsymbol{z}_1$ is removed from $\{\boldsymbol{u}\}_N$.

Then, the unique data sample nearest to $\boldsymbol{z}_r$ (denoted by $\boldsymbol{z}_2$) is selected from the rest of $\{\boldsymbol{u}\}_N$ as the new reference sample: $\boldsymbol{z}_r \leftarrow \boldsymbol{z}_2$, and $\boldsymbol{z}_2$ is removed from $\{\boldsymbol{u}\}_N$ as well. The process is repeated until $\{\boldsymbol{u}\}_N = \emptyset$. Then, the ranked unique data samples, denoted as $\{\boldsymbol{z}\}_N$ and their corresponding ranked multimodal density collection: $\{D_K^G(\boldsymbol{z})\}_N$ are obtained.

3.3.1.2. Stage 2: Prototypes Identification

In this stage, the local maxima of $\{D_K^G(\boldsymbol{z})\}_N$ are identified and the corresponding unique data samples with local maximum D^G are used as the prototypes to form clusters.

Condition 11 is used to identify the local maxima from $\{D_K^G(\boldsymbol{z})\}_N$ and all the data samples satisfying Condition 11 are re-denoted as $\{\boldsymbol{u}^*\}_N$:

$$Condition\ 11{:}\quad \begin{array}{c} IF\ \left(D_K^G(\mathbf{z}_{j-1}) < D_K^G(\mathbf{z}_j)\right)\ AND\ \left(D_L(\mathbf{z}_{j+1}) < D_K^G(\mathbf{z}_j)\right) \\ THEN\ \left(\mathbf{z}_j\ is\ a\ local\ maximum\ of\ D^G\right) \end{array}. \tag{3.43}$$

3.3.1.3. Stage 3: Creating Voronoi Tessellations

Once the collection, $\{\boldsymbol{u}^*\}_N$, is identified, its members are used as the focal points of the data clouds representing the local modes of the data pattern. All the data samples within $\{\boldsymbol{x}\}_K$ are then assigned to the nearest focal points using equation (3.12).

After all the data samples within $\{\boldsymbol{x}\}_K$ are assigned to the focal points, they naturally create Voronoi tessellation [64] and form data clouds. Assuming that there are C data clouds formed, these data clouds are ranked in terms of their supports (number of members) in an ascending order, denoted by S_i. The ranked data clouds are denoted as $\boldsymbol{\Xi}_i$ $(i = 1,2,\dots,C)$, where there is $S_1 \leq S_2 \leq \cdots \leq S_C$. The corresponding centres are denoted as $\boldsymbol{\mu}_i$ $(i = 1,2,\dots,C)$.

3.3.1.4. Stage 4: Filtering Local Modes

The data clouds formed in the previous stage may contain some less representative ones, therefore, in this stage, the initial Voronoi tessellations are filtered and combined into larger, more meaningful data clouds.

The multimodal densities of the data clouds centres $\{\boldsymbol{\mu}\}$ are firstly calculated using equation (2.21) with the corresponding supports $\{S\}$ used as the frequencies, denoted by $D_K^G(\boldsymbol{\mu}_i)$ $(i = 1,2,\dots,C)$. In order to identify the centres with the local maxima of multimodal density, the three objectively derived quantifiers of the data pattern are introduced:

$$\eta_K = \sum_{i=1}^{C-1}\sum_{j=i+1}^{C} d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)/\left(C(C-1)\right); \tag{3.44}$$

$$\gamma_K = \sum_{\boldsymbol{x},\boldsymbol{z}\in\{\boldsymbol{\mu}\}, d(\boldsymbol{x},\boldsymbol{z})\leq\eta_K, \boldsymbol{x}\neq\boldsymbol{z}} d(\boldsymbol{x},\boldsymbol{z})/C_\eta; \quad (3.45)$$

$$\lambda_K = \sum_{\boldsymbol{x},\boldsymbol{z}\in\{\boldsymbol{\mu}\}, d(\boldsymbol{x},\boldsymbol{z})\leq\gamma_K, \boldsymbol{x}\neq\boldsymbol{z}} d(\boldsymbol{x},\boldsymbol{z})/C_\gamma. \quad (3.46)$$

η_K is the average distance between any pair of the existing local modes. γ_K is the average distance between any pair of existing local modes with a distance less than η_K, and C_η is the number of such pairs. Similarly, λ_K is the average distance between any pair of existing local modes with a distance less than γ_K, and C_γ is the number of such local modes pairs. Note that, η_K , γ_K and λ_K are not problem-specific, but are parameter-free. The quantifier λ_K can be viewed as the estimation of the distances between the strongly connected data clouds condensing the local information from the whole data set. Moreover, instead of relying on a fixed threshold, which may frequently fail, η_K, γ_K and λ_K derived from the dataset objectively are guaranteed to be meaningful regardless of the distribution of the data.

Each centre $\boldsymbol{\mu}_i$ $(i = 1,2,\ldots,C)$ is compared with the centres of the neighbouring data clouds $\{\Xi\}_i^n$ in terms of the multimodal densities to identify the local maxima following Condition 12:

$$Condition\ 12: \begin{matrix} IF\ \left(D_K^G(\boldsymbol{\mu}_i) = \max(\{\{D_K^G(\boldsymbol{\mu})\}_i^n, D_K^G(\boldsymbol{\mu}_i)\})\right) \\ THEN\ (\boldsymbol{\mu}_i\ is\ one\ of\ the\ local\ maxima) \end{matrix}, \quad (3.47)$$

where $\{D_K^G(\boldsymbol{\mu})\}_i^n$ is the collection of multimodal densities of the neighbouring data cloud centres $\{\Xi\}_i^n$, which satisfy Condition 13 ($j = 1,2,\ldots,C, i \neq j$):

$$Condition\ 13: \begin{matrix} IF\ \left(d(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j) \leq \frac{\lambda_K}{2}\right) \\ THEN\ \left(\Xi_j\ is\ neighbouring\ \Xi_i\right) \end{matrix}. \quad (3.48)$$

The criterion of neighbouring range is defined in this way because two centres with the distance smaller than γ_K can be considered to be potentially relevant in the sense of spatial distance; λ_K is the average distance between the centres of any two potentially relevant data clouds. Therefore, when Condition 13 is satisfied, both $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ are highly influencing each other and, the data samples within the two corresponding data clouds are strongly connected. Therefore, the two data clouds are considered as neighbours. This criterion also guarantees that only small-size (less important) data clouds that significantly overlap with large-size (more important) ones will be removed during the filtering operation.

After the filtering operation, the data cloud centres with local maximum multimodal densities denoted by $\{\boldsymbol{\mu}^*\}$ are obtained. Then, $\{\boldsymbol{\mu}^*\}$ are used as local modes for forming data clouds in stage 3 ($\{\boldsymbol{u}^*\}_N \leftarrow \{\boldsymbol{\mu}^*\}$) and are filtered in stage 4.

Stages 3 and 4 are repeated until all the distances between the existing local modes exceed $\frac{\lambda_K}{2}$. Finally, we obtain the remaining centres with the local maxima of D^G, re-denoted by $\{\boldsymbol{\mu}^*\}$, and use them as the local modes to form data clouds using equation (3.12).

After the data clouds are formed, the corresponding centres, standard deviations, supports, members and other parameters of the formed data clouds can be extracted post factum.

3.3.1.5. Algorithm Summary

The main procedure of the offline ADP algorithm is summarised in the form of a flowchart presented in Figure 13.

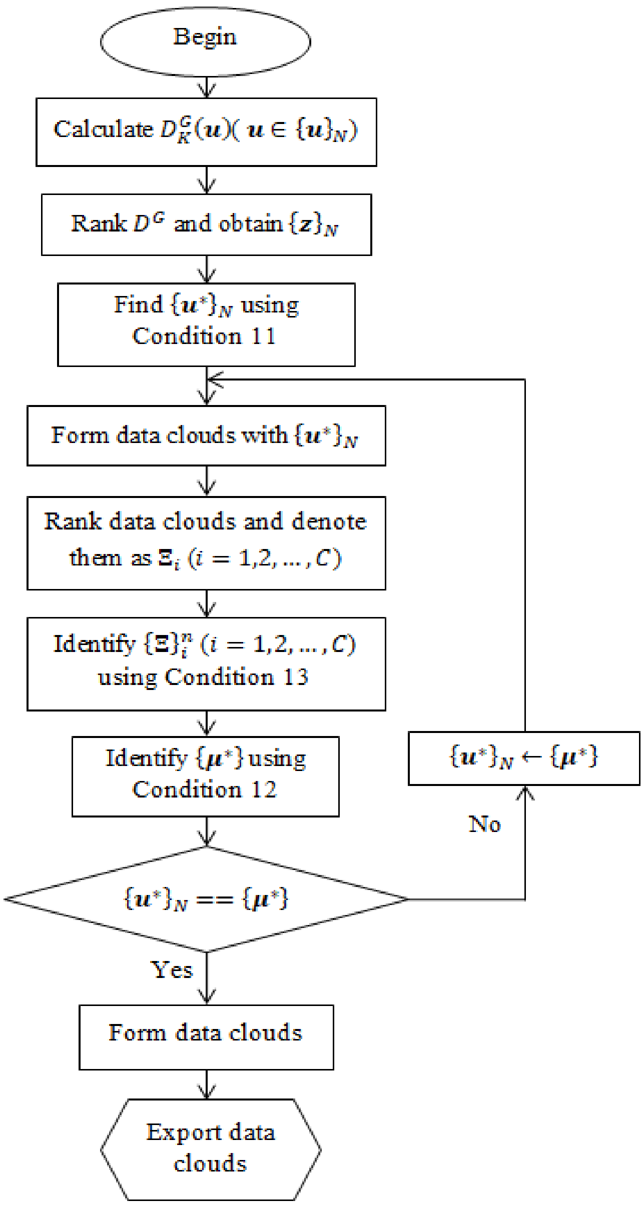

Begin
Calculate $D_K^G(\boldsymbol{u})(\boldsymbol{u} \in \{\boldsymbol{u}\}_N)$
Rank D^G and obtain $\{\boldsymbol{z}\}_N$
Find $\{\boldsymbol{u}^*\}_N$ using Condition 11
Form data clouds with $\{\boldsymbol{u}^*\}_N$
Rank data clouds and denote them as Ξ_i $(i = 1,2,\dots,C)$
Identify $\{\Xi\}_i^n$ $(i = 1,2,\dots,C)$ using Condition 13
Identify $\{\boldsymbol{\mu}^*\}$ using Condition 12
$\{\boldsymbol{u}^*\}_N \leftarrow \{\boldsymbol{\mu}^*\}$
No
$\{\boldsymbol{u}^*\}_N == \{\boldsymbol{\mu}^*\}$
Yes
Form data clouds
Export data clouds

Figure 13. Main procedure of the offline ADP algorithm.

3.3.2. Evolving ADP Algorithm

The evolving ADP algorithm works with the unimodal density D of the streaming data. This algorithm is able to start "from scratch". In addition, a hybrid between the evolving and the offline versions is also possible. The main procedure of the evolving algorithm is as follows [31]. Here the Euclidean distance is used for simpler derivation.

3.3.2.1. Stage 1: Initialisation

The first data sample within the data stream $\boldsymbol{x}_1$ is selected as the first local mode. The global parameters of the ADP algorithm are set as $K \leftarrow 1, C \leftarrow 1, \boldsymbol{\mu} \leftarrow \boldsymbol{x}_1$ and $X \leftarrow \|\boldsymbol{x}_1\|^2$, and the meta-parameters of the first data cloud are set as $\Xi_1 \leftarrow \{\boldsymbol{x}_1\}, S_1 \leftarrow 1$ and $\boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_1$, which are the same as equations (3.8) and (3.9).

The ADP algorithm then starts to self-evolve its structure and update the parameters based on the arriving data samples.

3.3.2.2. Stage 2: System Structure and Meta-Parameters Update

For each newly arriving data sample ($K \leftarrow K+1$), denoted as $\boldsymbol{x}_K$, the global meta-parameters $\boldsymbol{\mu}$ and X are updated firstly using equations (2.26) and (2.27). The unimodal density at $\boldsymbol{x}_K$ and the centres of all the existing data clouds, $D_K(\boldsymbol{x}_K)$ and $D_K(\boldsymbol{\mu}_i)$ ($i = 1,2, \dots, C$) are calculated using equations (2.18), (2.24) and (2.25).

Then, Condition 3 (equation (3.19)) is checked to decide whether $\boldsymbol{x}_K$ will form a new data cloud. If Condition 3 is met, a new data cloud is added with $\boldsymbol{x}_K$ as its local mode as: $C \leftarrow C+1$, $\Xi_C \leftarrow \{\boldsymbol{x}_K\}$, $\boldsymbol{\mu}_C \leftarrow \boldsymbol{x}_K$ and $S_C \leftarrow 1$. Otherwise, the existing local mode closest to $\boldsymbol{x}_K$ is found, denoted as $\boldsymbol{\mu}_n$. Then, Condition 14 is checked before $\boldsymbol{x}_K$ is assigned to the data cloud formed around the nearest data cloud centre $\boldsymbol{\mu}_n$:

$$Condition\ 14\text{:}\quad \begin{matrix} IF(\| \boldsymbol{\mu}_n - \boldsymbol{x}_K\| \leq \eta_K/2) \\ THEN\ (\boldsymbol{x}_K\ is\ assigned\ to\ \ \boldsymbol{\mu}_n) \end{matrix}. \tag{3.49}$$

However, it is not computationally efficient to calculate η_K at each time a new data sample arrives. Since the average distance between all the data samples η_K^d is approximately equal to η_K, $\eta_K \approx \eta_K^d$, η_K can be replaced as:

$$\eta_K \approx \eta_K^d = \sqrt{\frac{\sum_{i=1}^{K}\sum_{j=1}^{K}\|x_j - x_i\|^2}{K^2}} = \sqrt{2(X_K - \|\boldsymbol{\mu}\|^2)}. \tag{3.50}$$

If Condition 14 is satisfied, $\boldsymbol{x}_K$ is associated with the nearest existing local mode $\boldsymbol{\mu}_n$. The meta-parameters of this data cloud Ξ_n, namely S_n and $\boldsymbol{\mu}_n$ are updated using equation (3.13).

If Condition 14 is not satisfied, then $\boldsymbol{x}_K$ starts a new data cloud with the meta-parameters initialised as: $C \leftarrow C + 1$, $\Xi_C \leftarrow \{\boldsymbol{x}_K\}$, $\boldsymbol{\mu}_C \leftarrow \boldsymbol{x}_K$ and $S_C \leftarrow 1$.

The local modes and support of other data clouds that do not get the new data sample stay the same for the next processing cycle. After the update of the system structure and the meta-parameters, the algorithm is ready for the next data sample.

3.3.2.3. Stage 3: Data Clouds Formation

When there are no more data samples, the identified local modes (renamed as $\{\boldsymbol{\mu}^*\}$) are used to rebuild data clouds using equation (3.12). The parameters of these data clouds can be extracted post factum.

3.3.2.4. Algorithm Summary

The main procedure of the evolving ADP algorithm is summarised in the form of a flowchart presented in Figure 14.

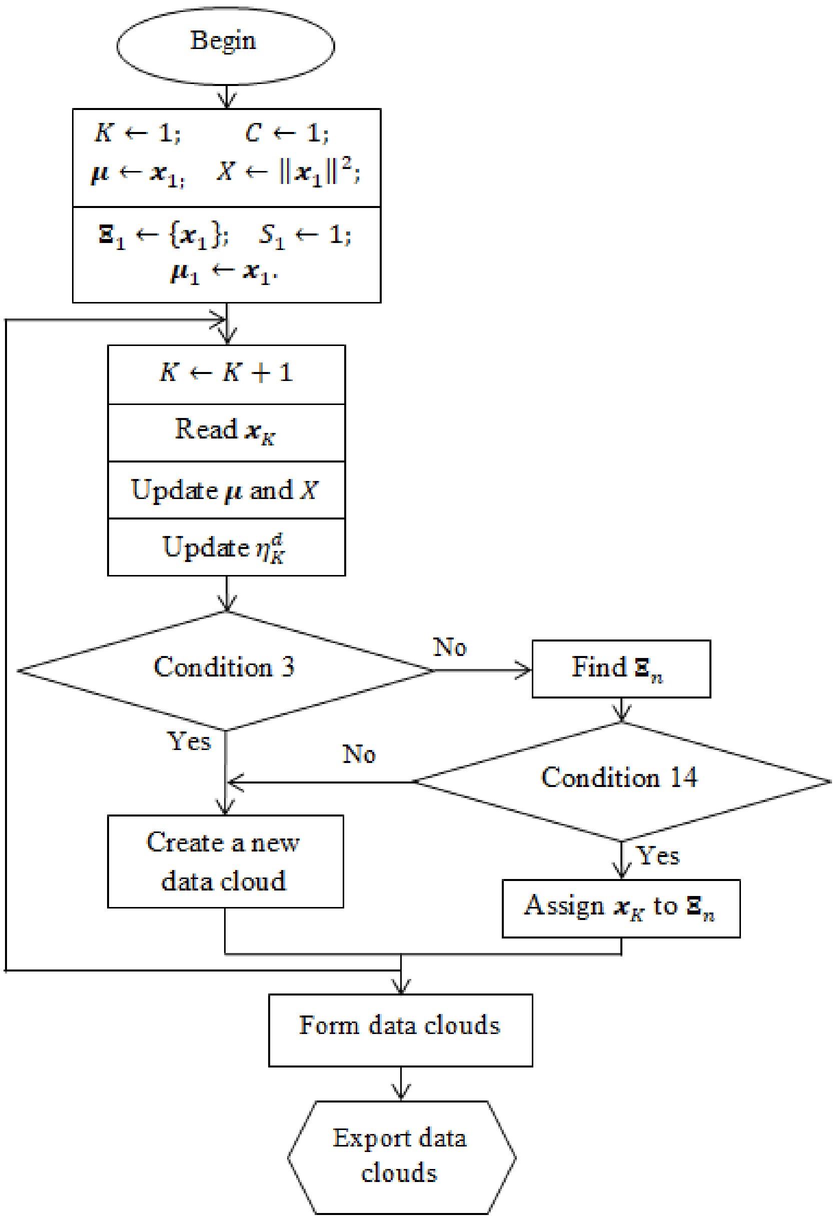

Begin
$K \leftarrow 1$; $C \leftarrow 1$;
$\boldsymbol{\mu} \leftarrow \boldsymbol{x}_1$; $X \leftarrow \|\boldsymbol{x}_1\|^2$;
$\boldsymbol{\Xi}_1 \leftarrow \{\boldsymbol{x}_1\}$; $S_1 \leftarrow 1$;
$\boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_1$.
$K \leftarrow K + 1$
Read $\boldsymbol{x}_K$
Update $\boldsymbol{\mu}$ and X
Update η_K^d
Condition 3
No
Find $\boldsymbol{\Xi}_n$
Yes
No
Condition 14
Create a new data cloud
Yes
Assign $\boldsymbol{x}_K$ to $\boldsymbol{\Xi}_n$
Form data clouds
Export data clouds

Figure 14. Main procedure of the evolving ADP algorithm.

3.3.3. Handling the Outliers in ADP

After the data clouds are formed by all the identified local modes, one may notice some data clouds with support equal to 1, which means that there is no sample associated with these data clouds except for the local modes. This kind of local modes are considered to be outliers. In the ADP algorithm presented in this thesis, the outliers are assigned to the nearest normal data clouds using equation (3.12) and the meta-parameters of the data clouds that receive new members are updated using equation (3.13). Nonetheless, it has to be stressed that these abnormal local modes are ignored from the partitioning results, but they can still be kept in memory in case new data samples arrive.

3.4. Self-Organising Direction-Aware Data Partitioning Algorithm

In this section, an autonomous algorithm named Self-Organised Direction Aware (SODA) data partitioning is presented. The SODA partitioning algorithm employs both a traditional distance metric and a cosine similarity based angular component. The widely used traditional distance metrics, including Euclidean, Mahalanobis, Minkowski distances, mainly measure the magnitude difference between vectors. The cosine similarity, instead, focuses on the directional similarity. The algorithm that takes into consideration both the spatial and the angular divergences results in a deeper understanding of the ensemble properties of the data.

Using EDA operators [26], [27], the SODA algorithm autonomously identifies the focal points (local peaks of the typicality, thus, the most representative points locally) from the observed data based on both, the spatial and angular divergences and, based on the focal points, it is able to disclose the ensemble properties and mutual distribution of the data. The possibility to calculate the EDA quantities incrementally enables us to propose computationally efficient algorithms.

The SODA algorithm consists of two versions, namely, offline and evolving. The offline version of the SODA algorithm is for static data processing, and an extension is also given, which enables the offline algorithm to follow the changing data pattern in an agile manner once primed/initialised with a seed dataset. The evolving SODA algorithm for streaming data employs the recently introduced direction-aware distance as the distance measure, and can start "from the scratch". In this section, Euclidean distance is used to measure the magnitude difference. The magnitude component is expressed as $d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ and the cosine similarity-based angular component is expressed as $d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{1 - cos\left(\theta_{x_i,x_j}\right)} = \left\| \frac{x_i}{\|x_i\|} - \frac{x_j}{\|x_j\|} \right\|$ $(\boldsymbol{x}_i, \boldsymbol{x}_j \in \{\boldsymbol{x}\}_K)$.

3.4.1. Offline SODA Algorithm

In this section, we will describe the proposed SODA algorithm. The main steps of the SODA algorithm include: firstly, form a number of direction-aware planes from the observed data samples using both the magnitude-based and angular-based unimodal densities; secondly, identify focal points; finally, use the focal points to partition the data space into data clouds. The detailed procedure of the proposed SODA partitioning algorithm is as follows.

3.4.1.1. Stage 1: Preparation

At this stage, the average square values between every pair of data samples, $\{\boldsymbol{x}\}_K$ for both, the Euclidean components, d_M and square angular components, d_A, are calculated:

$$\bar{d}_M^2 = \frac{\sum_{i=1}^{K}\sum_{j=1}^{K} d_M^2(\boldsymbol{x}_i,\boldsymbol{x}_j)}{K^2} = \frac{\sum_{i=1}^{K}\sum_{j=1}^{K}\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}{K^2} = 2(X_M - \|\boldsymbol{\mu}_M\|^2); \qquad (3.51)$$

$$\bar{d}_A^2 = \frac{\sum_{i=1}^{K}\sum_{j=1}^{K} d_A^2(\boldsymbol{x}_i,\boldsymbol{x}_j)}{K^2} = \frac{\sum_{i=1}^{K}\sum_{j=1}^{K}\left\|\frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|}-\frac{\boldsymbol{x}_j}{\|\boldsymbol{x}_j\|}\right\|^2}{2K^2} = 1 - \|\boldsymbol{\mu}_A\|^2, \qquad (3.52)$$

where X_M and $\boldsymbol{\mu}_M$ are the means of $\{\|\boldsymbol{x}\|^2\}_K$ and $\{\boldsymbol{x}\}_K$, respectively; $\boldsymbol{\mu}_A$ is the mean of $\left\{\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\right\}_K$.

Then, the multimodal densities D^G of the unique data samples $\{\boldsymbol{u}\}_N$ are calculated using equations (2.19), (2.24), (2.25), (2.32) and (2.33), as $D_K^G(\boldsymbol{u}_i) = f_i\left(\frac{1}{1+\frac{\|\boldsymbol{u}_i-\boldsymbol{\mu}_M\|^2}{X_M-\|\boldsymbol{\mu}_M\|^2}} + \frac{1}{1+\frac{\|\boldsymbol{u}_i/\|\boldsymbol{u}_i\|-\boldsymbol{\mu}_A\|^2}{1-\|\boldsymbol{\mu}_A\|^2}}\right)$. Then, $\{\boldsymbol{u}\}_N$ are ranked in a descending order in terms of D^G, which are re-denoted as $\{\boldsymbol{z}\}_N$.

3.4.1.2. Stage 2: Direction-Aware Plane Projection

The direction-aware projection operation begins with the unique data sample that has the highest multimodal density, namely $\boldsymbol{z}_1$. It is initially set to be the first reference, $\boldsymbol{\mu}_1 \leftarrow \boldsymbol{z}_1$, which is also the origin point of the first direction-aware plane, denoted by $\aleph_1$ ($P \leftarrow 1$, P is the number of existing direction-aware planes in the data space). For the rest of the unique data samples $\boldsymbol{z}_j$ ($j = 2,3,\dots,N$), Condition 15 is checked sequentially:

$$Condition\ 15\colon \quad \begin{matrix} IF\left(\frac{d_M(\boldsymbol{\mu}_i,\boldsymbol{z}_j)}{\bar{d}_M} \le \frac{1}{\gamma}\right)\ AND\ \left(\frac{d_A(\boldsymbol{\mu}_i,\boldsymbol{z}_j)}{\bar{d}_A} \le \frac{1}{\gamma}\right), \\ THEN\left(\aleph_i \leftarrow \{\aleph_i,\boldsymbol{z}_j\}\right) \end{matrix} \qquad (3.53)$$

where $i = 1,2,\dots,P$; γ is set to decide the granularity of the partitioning results and relates to the Chebyshev inequality [52]. If two or more direction-aware planes satisfy Condition 15 at the same time, $\boldsymbol{z}_j$ will be assigned to the nearest of them:

$$n = \operatorname{argmin}_{i=1,2,\dots,P}\left(\frac{d_M(\boldsymbol{\mu}_i,\boldsymbol{z}_j)}{\bar{d}_M} + \frac{d_A(\boldsymbol{\mu}_i,\boldsymbol{z}_j)}{\bar{d}_A}\right). \qquad (3.54)$$

The meta-parameters (mean $\boldsymbol{\mu}_n$, support/number of data samples, S_n and sum of multimodal density, denoted by $D^G(\aleph_n)$) of the n^{th} direction-aware plane are being updated as follows:

$$\begin{aligned} &\aleph_n \leftarrow \{\aleph_n, \boldsymbol{z}_j\}; \qquad \boldsymbol{\mu}_n \leftarrow \frac{S_n}{S_n+1}\boldsymbol{\mu}_n + \frac{1}{S_n+1}\boldsymbol{z}_j; \\ &S_n \leftarrow S_n + 1; \quad D^G(\aleph_n) \leftarrow D^G(\aleph_n) + D_K^G(\boldsymbol{z}_j). \end{aligned} \tag{3.55}$$

If Condition 15 is not met, $\boldsymbol{z}_j$ is set to be a new reference and a new direction-aware plane is set up as follows:

$$\begin{aligned} &P \leftarrow P+1; \quad \aleph_P \leftarrow \{\boldsymbol{z}_j\}; \quad \boldsymbol{\mu}_P \leftarrow \boldsymbol{z}_j; \\ &S_P \leftarrow 1; \quad D^G(\aleph_P) \leftarrow D_K^G(\boldsymbol{z}_j). \end{aligned} \tag{3.56}$$

After all the unique data samples are projected onto the direction-aware planes, the next stage can start.

3.4.1.3. Stage 3: Focal Points Identification

In this stage, for each direction-aware plane, denoted as $\aleph_i$ ($i = 1,2,\dots,P$), Condition 16 is used to find the neighbouring direction-aware planes, denoted by $\{\aleph\}_i^n$:

$$Condition\ 16: \begin{array}{c} IF \left(\frac{d_M(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j)}{\bar{d}_M} \le \frac{2}{\gamma}\right) AND \left(\frac{d_A(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j)}{\bar{d}_A} \le \frac{2}{\gamma}\right), \\ THEN\ (\{\aleph\}_i^n \leftarrow \{\{\aleph\}_i^n, \aleph_j\}) \end{array} \tag{3.57}$$

where $j = 1,2,\dots,P$. Condition 16 can be related to the Chebyshev inequality as well [52].

Then, Condition 17 is used to find the direction-aware planes standing for the local maxima of the data density ($i = 1,2,\dots,P$):

$$Condition\ 17: \begin{array}{c} IF\ (D^G(\aleph_i) > \max(\{D^G(\aleph)\}_i^n)) \\ THEN\ (\aleph_i\ is\ a\ local\ maximum) \end{array}. \tag{3.58}$$

By using Conditions 16 and 17 to examine each existing direction-aware plane, one can find all the modes/peaks of the data density, and the origin points of the local maximum planes are re-denoted as $\{\boldsymbol{\mu}^*\}$.

3.4.1.4. Stage 4: Forming Data Clouds

By using $\{\boldsymbol{\mu}^*\}$ as the focal points, data clouds can be formed using equation (3.59) as a Voronoi tessellation [64] :

$$\begin{aligned} &n = \operatorname{argmin}_{\boldsymbol{\mu}\in\{\boldsymbol{\mu}^*\}}\left(\frac{d_M(\boldsymbol{\mu},\boldsymbol{x})}{\bar{d}_M} + \frac{d_A(\boldsymbol{\mu},\boldsymbol{x})}{\bar{d}_A}\right); \\ &\Xi_n \leftarrow \{\Xi_n, \boldsymbol{x}\}; \end{aligned} \quad \boldsymbol{x} \in \{\boldsymbol{x}\}_K. \tag{3.59}$$

3.4.1.5. Algorithm Summary

The main procedure of the offline SODA algorithm is summarised in the form of a flowchart presented in Figure 15.

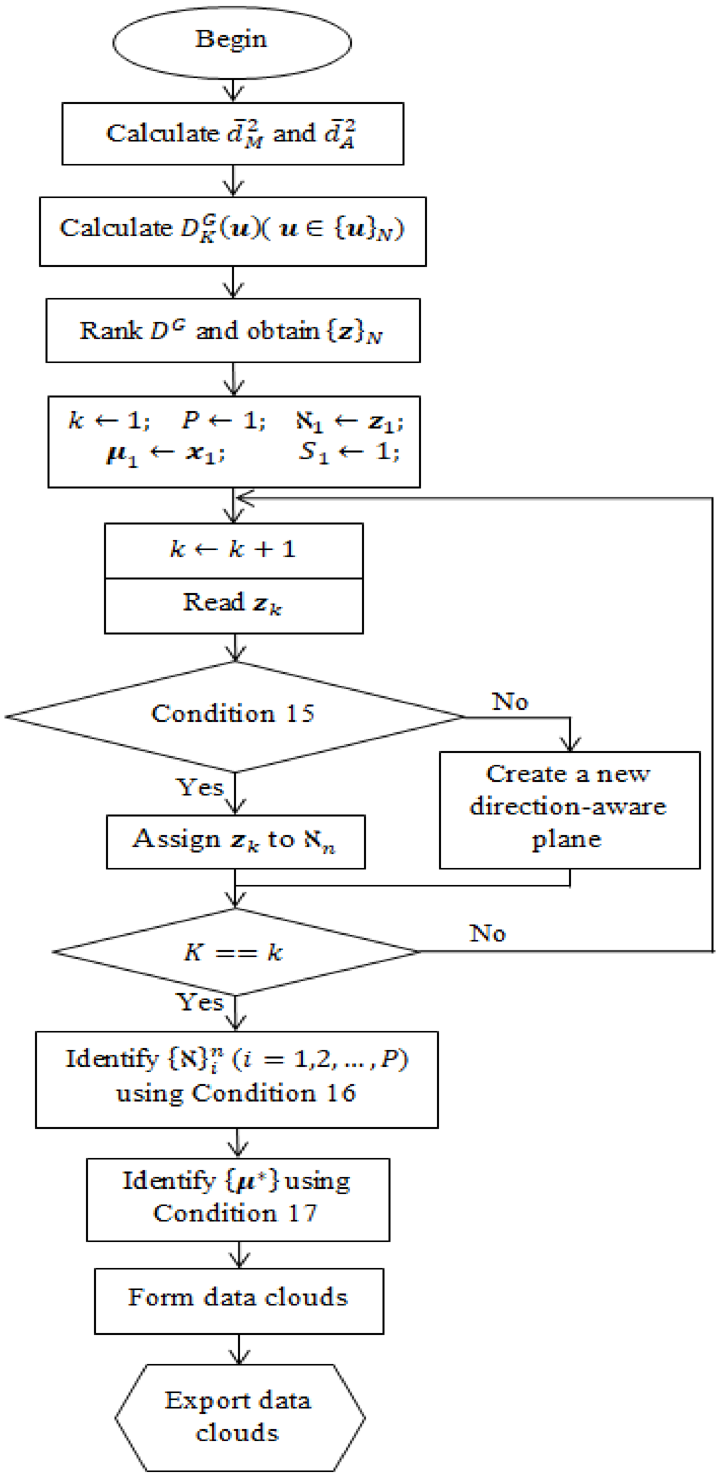

Begin
Calculate $\bar{d}_M^2$ and $\bar{d}_A^2$
Calculate $D_K^G(\boldsymbol{u})(\ \boldsymbol{u} \in \{\boldsymbol{u}\}_N)$
Rank D^G and obtain $\{\boldsymbol{z}\}_N$
$k \leftarrow 1;\ P \leftarrow 1;\ \aleph_1 \leftarrow \boldsymbol{z}_1;\ \boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_1;\ S_1 \leftarrow 1;$
$k \leftarrow k+1$
Read $\boldsymbol{z}_k$
Condition 15
No
Yes
Create a new direction-aware plane
Assign $\boldsymbol{z}_k$ to $\aleph_n$
$K == k$
No
Yes
Identify $\{\aleph\}_i^n\ (i = 1,2,\dots,P)$ using Condition 16
Identify $\{\boldsymbol{\mu}^*\}$ using Condition 17
Form data clouds
Export data clouds

Figure 15. Main procedure of the offline SODA algorithm.

3.4.2. Extension of the Offline SODA Algorithm

In this section, an extension to the offline SODA algorithm will be introduced to allow the algorithm to continue to process the streaming data on the basis of the partitioning results initiated by a static dataset. As a result, the main procedure of this extension for streaming data processing will be built based on a structure initiated by an offline priming (does not start "from scratch").

The main procedure of the extension of the offline algorithm for the streaming data processing is as follows.

3.4.2.1. Stage 1: Meta-parameters Update

After the static dataset has been processed, for each newly arrived data sample ($K \leftarrow K+1$) from the data stream, denoted by $\boldsymbol{x}_K$, $\boldsymbol{\mu}_M$, X_M and $\boldsymbol{\mu}_A$ are updated using equations (2.26), (2.27) and (2.34). The values of the Euclidean components, d_M and the angular components, d_A between $\boldsymbol{x}_K$ and the centres $\boldsymbol{\mu}_i$ $(i = 1,2,\ldots,P)$ of the existing direction-aware planes are calculated, denoted as $d_M(\boldsymbol{x}_K, \boldsymbol{\mu}_i)$ and $d_A(\boldsymbol{x}_K, \boldsymbol{\mu}_i)$ $(i = 1,2,\ldots,P)$. $\bar{d}_M^2$ and $\bar{d}_A^2$ are updated using equations (3.51) and (3.52) as well.

Then, Condition 15 and equation (3.54) are used to find the direction-aware plane $\boldsymbol{x}_K$ is associated with. If Condition 15 is met and $\boldsymbol{x}_K$ is associated with the existing direction-aware plane, assuming $\aleph_n$, $\boldsymbol{x}_K$ is assigned to $\aleph_n$ and the corresponding meta-parameters $\boldsymbol{\mu}_n$ and S_n will be updated using equation (3.55). Otherwise, a new direction-aware plane is set up by $\boldsymbol{x}_K$ ($P \leftarrow P+1, \aleph_P \leftarrow \{\boldsymbol{x}_K\}$) with the meta-parameters ($\boldsymbol{\mu}_P$ and S_P) set up by equation (3.56).

3.4.2.2. Stage 2: Merging Overlapping Direction-Aware Planes

After the Stage 1 is finished, Condition 18 is checked to identify the heavily overlapping direction-aware planes in the data space ($i,j = 1,2,\ldots,L;\ 1 \le i < j \le P$):

$$Condition\ 18:\quad \begin{array}{c} IF\ \left(\frac{d_M(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j)}{\bar{d}_M} \le \frac{1}{2\gamma}\right)\ AND\ \left(\frac{d_A(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j)}{\bar{d}_A} \le \frac{1}{2\gamma}\right) \\ THEN\ \left(\aleph_i\ and\ \aleph_j\ are\ heavily\ overlapping\right) \end{array}. \qquad (3.60)$$

If $\aleph_i$ and $\aleph_j$ ($i,j = 1,2,\ldots,P;\ 1 \le i < j \le P$) meet condition 18, they are merged together on the basis of $\aleph_j$ ($P \leftarrow P-1$) with the meta-parameters $\boldsymbol{\mu}_j$ and S_j updated using equation (3.19). Meanwhile, the meta-parameters of $\aleph_i$ are deleted.

The merging process repeats until all the heavily overlapping direction-aware planes have been merged. Then, the algorithm goes back to Stage 1 and waits for the newly coming data sample. If there is no new data sample anymore, the algorithm goes to the final stage.

3.4.2.3. Stage 3: Forming Data Clouds

Once there are no new data samples available, the SODA algorithm will quickly identify the focal points from the centres of the existing direction-aware planes.

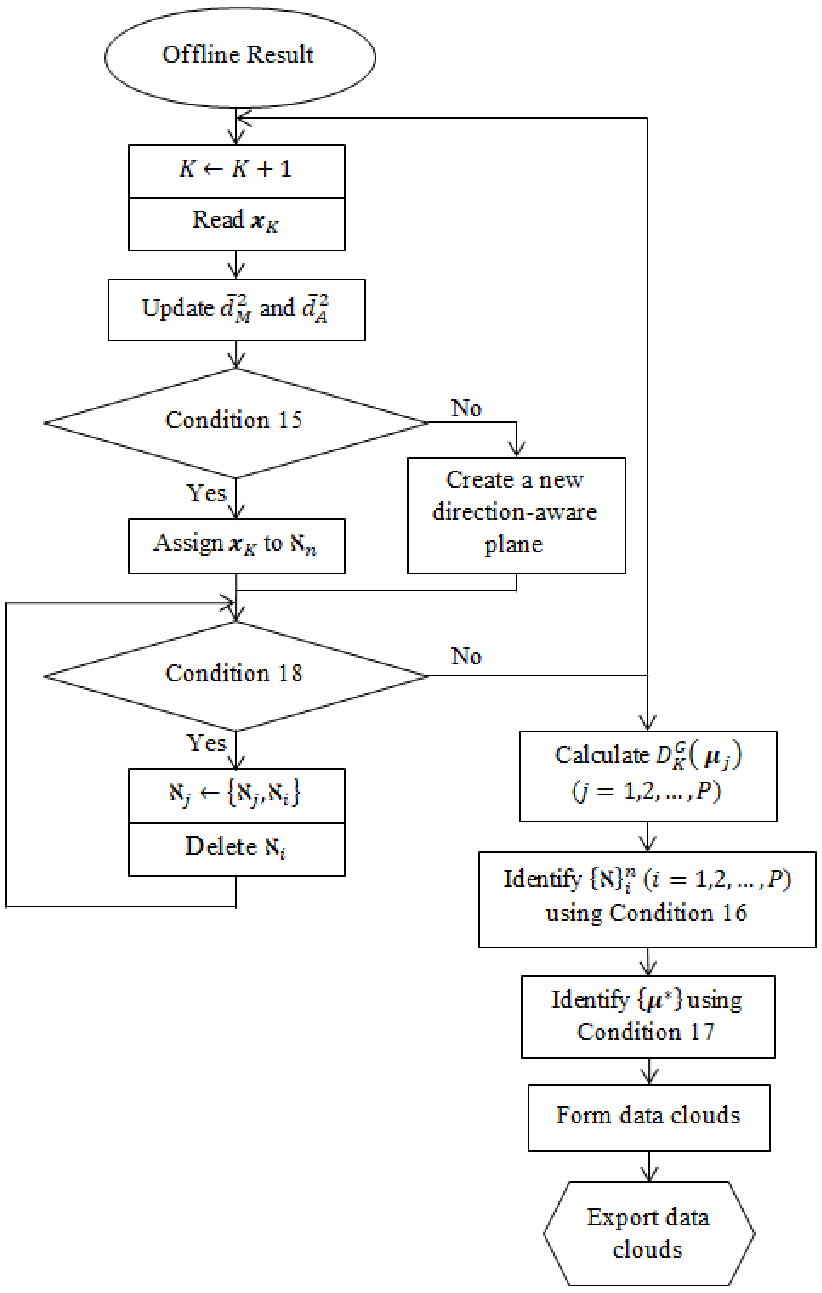
Firstly, the multimodal densities of the centres $\boldsymbol{\mu}_j$ $(j = 1,2,\dots,P)$ of the direction-aware planes are calculated as $D_K^G(\boldsymbol{\mu}_j) = S_j\left(\frac{1}{1+\frac{\|\mu_j-\mu_M\|^2}{X_M-\|\mu_M\|^2}} + \frac{1}{1+\frac{\|\mu_j/\|\mu_j\|-\mu_A\|^2}{1-\|\mu_A\|^2}}\right)$ $(j = 1,2,\dots,P)$, where the support S_j $(j = 1,2,\dots,P)$ of each direction-aware plane is used as the corresponding frequency.

Secondly, for each existing direction-aware plane, $\aleph_i$, Condition 16 is used to find the neighbouring planes around it, denoted as $\{\aleph\}_i^n$. Condition 17 is used to check whether $D_K^G(\boldsymbol{\mu}_j)$ $(j = 1,2,\dots,P)$ is one of the local maxima.

Finally, for all the identified local maxima of D^G, the centres of the corresponding planes, denoted as $\{\boldsymbol{\mu}^*\}$ will serve as the focal points to form the data clouds using equation (3.59).

3.4.2.4. Algorithm Summary

The main procedure of the offline SODA algorithm extension is summarised in the form of a flowchart presented in Figure 16.



Offline Result
$K \leftarrow K+1$
Read $\boldsymbol{x}_K$
Update $\bar{d}_M^2$ and $\bar{d}_A^2$
Condition 15
No
Yes
Create a new direction-aware plane
Assign $\boldsymbol{x}_K$ to $\aleph_n$
Condition 18
No
Yes
$\aleph_j \leftarrow \{\aleph_j, \aleph_i\}$
Delete $\aleph_i$
Calculate $D_K^G(\boldsymbol{\mu}_j)$ $(j = 1,2,\dots,P)$
Identify $\{\aleph\}_i^n$ $(i = 1,2,\dots,P)$ using Condition 16
Identify $\{\boldsymbol{\mu}^*\}$ using Condition 17
Form data clouds
Export data clouds

Figure 16. Main procedure of the offline SODA algorithm extension.

3.4.3. Evolving SODA Algorithm

In this section, the evolving SODA algorithm is presented, which employs the recently introduced direction-aware distance [33] as the distance measure, which is also described in section 2.1.3.6 as well. This algorithm is able to "start from scratch" and consistently evolve its system structure and update the meta-parameters based on the newly arrived data samples. In this evolving version, without a loss of generality, the two scaling coefficients of direction-aware distance, namely λ_M and λ_A are set to be $\lambda_M = \frac{1}{\bar{d}_M^2}$ and $\lambda_A = \frac{1}{\bar{d}_A^2}$, which are derived by equations (3.51) and (3.52) [33].

The main procedure of the proposed algorithm is described as follows.

3.4.3.1. Stage 1: Initialisation

The first data sample $\boldsymbol{x}_1$ in the data stream is used for initialising the first data cloud and its meta-parameters. In the evolving SODA algorithm, the system has the following initialised global meta-parameters: $K \leftarrow 1$, $C \leftarrow 1$, $\boldsymbol{\mu}_M \leftarrow \boldsymbol{x}_1$, $X_M \leftarrow \|\boldsymbol{x}_1\|^2$, $\boldsymbol{\mu}_A \leftarrow \frac{\boldsymbol{x}_1}{\|\boldsymbol{x}_1\|}$ and $X_A \leftarrow 1$. And the meta-parameters of the first data cloud are set as ($\Xi_1 \leftarrow \{\boldsymbol{x}_1\}$):

$$\boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_1; \quad \bar{\boldsymbol{\mu}}_1 \leftarrow \frac{\boldsymbol{x}_1}{\|\boldsymbol{x}_1\|}; \quad X_1 \leftarrow \|\boldsymbol{x}_1\|^2; \quad \bar{X}_1 \leftarrow 1; \quad S_1 \leftarrow 1, \tag{3.61}$$

where $\bar{\boldsymbol{\mu}}_1$ is the normalised mean of $\aleph_1$; $\bar{X}_1$ is the corresponding normalised average scalar product.

After the initialisation of the system, the evolving SODA algorithm starts to update the system structure and meta-parameters with the arrival of each new data samples.

3.4.3.2. Stage 2: System Structure and Meta-Parameters Update

With each newly arrived data sample ($K \leftarrow K + 1$), the system's global meta-parameters, $\boldsymbol{\mu}_M$, X_M and $\boldsymbol{\mu}_A$ are updated with $\boldsymbol{x}_K$ using the equations (2.26), (2.27) and (2.34) [26]. The two scaling parameters, λ_M and λ_A, are updated accordingly using equations (3.51) and (3.52).

Then, the nearest data cloud Ξ_n to $\boldsymbol{x}_K$ with the centre denoted by $\boldsymbol{\mu}_n$ is identified using equation (3.62):

$$\boldsymbol{\mu}_n = \text{argmin}_{i=1,2,\dots,C}\left(d_{DA}(\boldsymbol{x}_K, \boldsymbol{\mu}_i)\right). \tag{3.62}$$

And the direction-aware distance between $\boldsymbol{\mu}_n$ and $\boldsymbol{x}_K$ is obtained as $d_{DA}(\boldsymbol{x}_K, \boldsymbol{\mu}_n)$.

Condition 19 is checked to see whether $\boldsymbol{x}_K$ is associated with a new data cloud:

$$Condition\ 19\!: \begin{array}{c} IF\ \left(d_{DA}(\boldsymbol{x}_K, \boldsymbol{\mu}_M) > \max_{j=1,2,\dots,C}\left(d_{DA}(\boldsymbol{\mu}_j, \boldsymbol{\mu}_M)\right)\right) \\ OR\ \left(d_{DA}(\boldsymbol{x}_K, \boldsymbol{\mu}_M) < \min_{j=1,2,\dots,C}\left(d_{DA}(\boldsymbol{\mu}_j, \boldsymbol{\mu}_M)\right)\right), \\ OR\ (d_{DA}(\boldsymbol{x}_K, \boldsymbol{\mu}_n) > d_o) \\ THEN\ (\boldsymbol{x}_K\ creates\ a\ new\ data\ cloud) \end{array} \tag{3.63}$$

where $d_o = 0.5$.

If Condition 19 is satisfied, a new data cloud is added with $\boldsymbol{x}_K$ as its centre:

$$\begin{aligned} &C \leftarrow C+1; \quad \Xi_C \leftarrow \{\boldsymbol{x}_K\}; \quad \boldsymbol{\mu}_C \leftarrow \boldsymbol{x}_K; \\ &\bar{\boldsymbol{\mu}}_C \leftarrow \frac{\boldsymbol{x}_K}{\|\boldsymbol{x}_K\|}; \quad X_C \leftarrow \|\boldsymbol{x}_K\|^2; \quad \bar{X}_C \leftarrow \|\boldsymbol{x}_K\|^2; \quad S_C \leftarrow 1. \end{aligned} \tag{3.64}$$

In contrast, if Condition 19 is not met, $\boldsymbol{x}_K$ is assigned to the nearest data cloud Ξ_n, and the meta-parameters of Ξ_n are updated as follows ($C \leftarrow C$) [26]:

$$\begin{aligned} &\Xi_n \leftarrow \{\Xi_n, \boldsymbol{x}_K\}; \quad \boldsymbol{\mu}_n \leftarrow \frac{S_n}{S_n+1}\boldsymbol{\mu}_n + \frac{1}{S_n+1}\boldsymbol{x}_K; \quad \bar{\boldsymbol{\mu}}_n \leftarrow \frac{S_n}{S_n+1}\bar{\boldsymbol{\mu}}_n + \frac{1}{S_n+1}\frac{\boldsymbol{x}_K}{\|\boldsymbol{x}_K\|}; \\ &X_n \leftarrow \frac{S_n}{S_n+1}X_n + \frac{1}{S_n+1}\|\boldsymbol{x}_K\|^2; \quad S_n \leftarrow S_n + 1. \end{aligned} \tag{3.65}$$

After the update of the global and local meta-parameters, the system is ready for the arrival of the next data sample and begins a new processing cycle.

3.4.3.3. Stage 3: Filtering Data Clouds

In this stage, all the existing data clouds will be examined and adjusted to avoid the possible overlap. For each existing cloud Ξ_i ($i = 1,2,\dots,C$), firstly, its neighbouring clouds, denoted by $\{\Xi\}_i^n$ based on Condition 20:

$$Condition\ 20: \quad \begin{array}{l} IF\ \left(d_{DA}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) < \frac{\sum_{k=1}^{C}\bar{d}_k}{C}\right) \\ THEN\ \left(\{\Xi\}_i^n \leftarrow \{\{\Xi\}_i^n, \Xi_j\}\right) \end{array}, \tag{3.66}$$

where $\bar{d}_k^2 = \sum_{\boldsymbol{x}\in\Xi_k}\sum_{\boldsymbol{y}\in\Xi_k} d_{DA}^2(\boldsymbol{x},\boldsymbol{y})/S_k^2$ is the average square direction-aware distance between all the members within the *k*th data cloud Ξ_k.

For each cluster centre, $\boldsymbol{\mu}_i$ ($i = 1,2,\dots,C$), its multimodal density is calculated as [26]:

$$D_K^G(\boldsymbol{\mu}_i) = S_i \frac{\sum_{l=1}^{C}\sum_{j=1}^{C} d_{DA}^2(\boldsymbol{\mu}_l,\boldsymbol{\mu}_j)}{2C\sum_{j=1}^{C} d_{DA}^2(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j)}, \tag{3.67}$$

and it is compared with the D^G of its neighbouring data clouds denoted by $\{D^G(\boldsymbol{\mu})\}_i^n$, to identify the local maxima of D^G using Condition 16.

By identifying all the local maxima, denoted by $\{\boldsymbol{\mu}^*\}$ and assigning each data sample to the data cloud with the nearest centre using equation (3.12), the whole partitioning processing is finished. The parameters of the data clouds can be extracted post factum.

3.4.3.4. Algorithm Summary

The main procedure of the evolving SODA algorithm extension is summarised in the form of a flowchart presented in Figure 17.

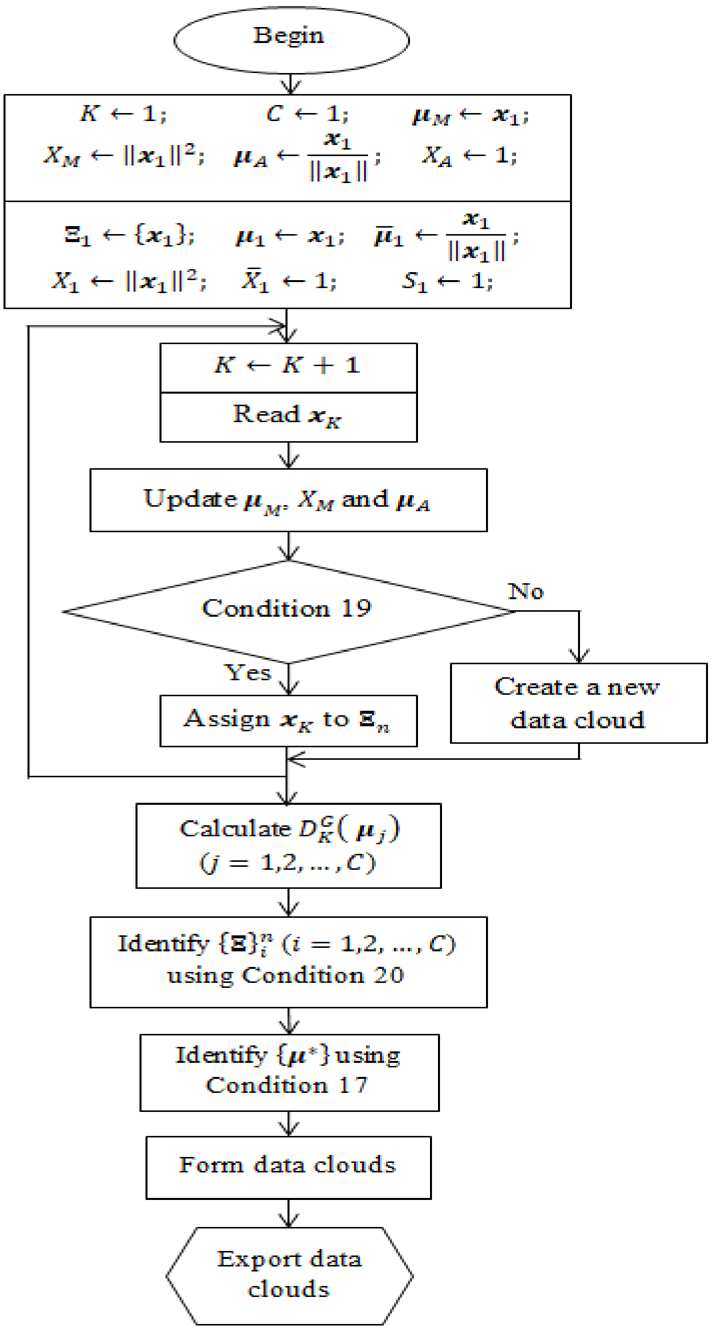

Begin
$K \leftarrow 1$; $C \leftarrow 1$; $\boldsymbol{\mu}_M \leftarrow \boldsymbol{x}_1$;
$X_M \leftarrow \|\boldsymbol{x}_1\|^2$; $\boldsymbol{\mu}_A \leftarrow \frac{\boldsymbol{x}_1}{\|\boldsymbol{x}_1\|}$; $X_A \leftarrow 1$;
$\boldsymbol{\Xi}_1 \leftarrow \{\boldsymbol{x}_1\}$; $\boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_1$; $\bar{\boldsymbol{\mu}}_1 \leftarrow \frac{\boldsymbol{x}_1}{\|\boldsymbol{x}_1\|}$;
$X_1 \leftarrow \|\boldsymbol{x}_1\|^2$; $\bar{X}_1 \leftarrow 1$; $S_1 \leftarrow 1$;
$K \leftarrow K + 1$
Read $\boldsymbol{x}_K$
Update $\boldsymbol{\mu}_M$, X_M and $\boldsymbol{\mu}_A$
Condition 19
No
Yes
Assign $\boldsymbol{x}_K$ to $\boldsymbol{\Xi}_n$
Create a new data cloud
Calculate $D_K^G(\boldsymbol{\mu}_j)$ $(j = 1,2,\ldots,C)$
Identify $\{\boldsymbol{\Xi}\}_i^n$ $(i = 1,2,\ldots,C)$ using Condition 20
Identify $\{\boldsymbol{\mu}^*\}$ using Condition 17
Form data clouds
Export data clouds

Figure 17. Main procedure of the evolving SODA algorithm.

3.5. Conclusion

As the main stream of unsupervised machine learning techniques, clustering algorithms play an important role in data analysis, data mining and pattern recognition. However, traditional approaches suffer from various deficiencies, and they often fail to produce meaningful results on real problems.

In this chapter, four different types of self-organising, data-driven, nonparametric clustering/data partitioning approaches developed within the EDA framework are presented.

Compared with traditional approaches, the novel approaches presented in this chapter are free from the prior assumptions and predefined user- and problem-specific parameters. They are able to perform high quality clustering/partitioning results without any prior knowledge about the problem, and therefore, the scope of their applications can be very wide in the era of big data.

4. Supervised Self-Organising Machine Learning Algorithms

As it was discussed in section 2.3.2 – section 2.3.4, traditional supervised machine learning algorithms suffer from various deficiencies, including:

1) They rely on prior assumptions and predefined parameters for good performance;

2) Their system structures lack the ability of self-evolving.

In this chapter, the newly introduced supervised self-organising machine learning algorithms within the EDA framework are presented, which are autonomous, entirely data-driven and free from prior assumptions and user- and problem- specific parameters.

This chapter is organised as follows. Section 4.1 introduces autonomous learning multi-model system for streaming data processing as presented in [34]. The autonomous learning multi-model system of 0-order [35] is presented in section 4.2, which is very strong for large-scale, complex classification problems (see subsection 6.2.2 and also see [35]). A new type of self-organising fuzzy logic classifier with the ability of performing objective classification under different level of granularities is given in section 4.3. The autonomous anomaly detection algorithm is presented in section 4.4 and this chapter is concluded by section 4.5.

4.1. Autonomous Learning Multi-Model Systems

In this section, the autonomous learning multi-model system for streaming data processing, named ALMMO [34], is presented. The ALMMO system touches the very foundations of the complex learning systems for streaming data processing, and thus, it can be applied in areas including online data analytics, classification, regression, etc. In this section, the general architecture, structure identification and parameter identification of the ALMMO system will be presented. For simpler derivation, the Euclidean distance is used below, however, other types of distance metric and dissimilarity can be considered as well.

4.1.1. General Architecture

In the ALMMO system, the structure is composed of constraints-free data clouds forming Voronoi tessellation [64] in terms of the input and output variables. Its structure identification concerns the identification of the focal points of the data clouds as well as the parameters of output local models. Correspondingly, the parameter identification problem of the proposed approach is to determine the optimal values of the consequent parameters of the local (linear or singleton) models [55], [60]. The structure of the ALMMO system is given in Figure 18.

The ALMMO system can also be viewed as an autonomously self-developing AnYa type FRB system designed with the principles and mechanisms of the Empirical Data Analytics (EDA) computational framework [25]–[27], [33]. Specific characteristics that set ALMMO apart from the existing methods and schemes include:

1) it employs the nonparametric EDA quantities of density and typicality to disclose the underlying data pattern of the streaming data;

2) its system structure is composed of data clouds free from external constrains and self-updating output local models identified in a data-driven way;

3) it further defines and identifies a unimodal density (equation (2.18)) based membership function [54] designed within the EDA framework for the AnYa type FRB system [60];

4) it can, in a natural way, deal with heterogeneous data combining categorical with numerical data [54].

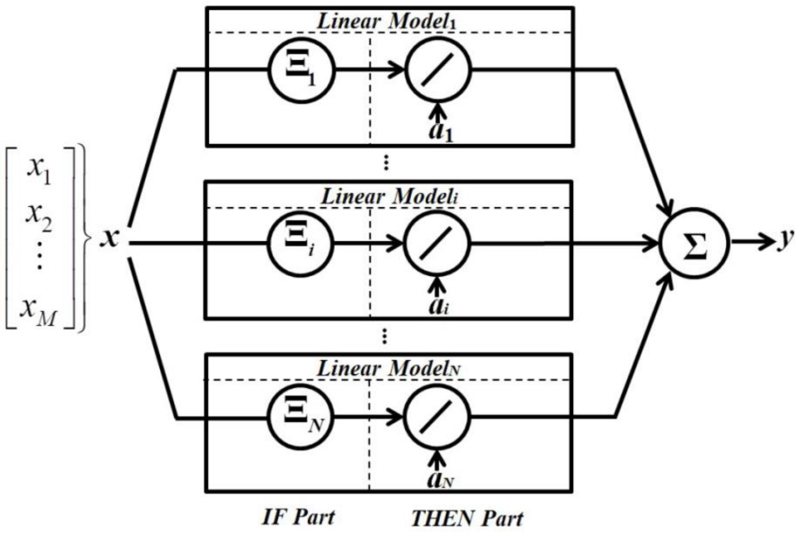


Figure 18. Structure of the ALMMO system.

4.1.2. Structure Identification

In this subsection, the structure identification process of the ALMMO system is described.

4.1.2.1. System Initialisation

For the first data sample, $\boldsymbol{x}_1$, the meta-parameters of the system are initialised as: $K \leftarrow 1$; $\boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_1$; $X_1 \leftarrow \|\boldsymbol{x}_1\|^2$; $N_1 \leftarrow 1$. And the first data cloud within the system is initialised as: $\Xi_1 \leftarrow \{\boldsymbol{x}_1\}$; $\boldsymbol{p}_{1,1} \leftarrow \boldsymbol{x}_1$; $X_{1,1} \leftarrow \|\boldsymbol{x}_1\|^2$; $S_{1,1} \leftarrow 1$.

4.1.2.2. Structure Update

For each newly arrived data sample, $\boldsymbol{x}_K$ ($K \leftarrow K+1$), the global mean and average scalar products $\boldsymbol{\mu}_{K-1}$ and X_{K-1} are updated to $\boldsymbol{\mu}_K$ and X_K using equations (2.26) and (2.27) first.

The unimodal densities of the data sample $\boldsymbol{x}_K$ and all the identified focal points, denoted by $\boldsymbol{p}_{K-1,j}$ ($j = 1,2,\ldots,N_{K-1}$) are calculated using equations (2.18), (2.24) and (2.25).

Then, Condition 3 (equation (3.10)) is checked to see whether $\boldsymbol{x}_K$ will generate a new data cloud and becomes a new prototype added into the fuzzy rule [55]. If Condition 3 (equation (3.10)) is triggered, a new data cloud is being formed around $\boldsymbol{x}_K$.

However, it is also necessary to check whether the newly formed data cloud is overlapping with the existing data clouds, and Condition 21 is used here to avoid possible overlaps ($i = 1,2,\ldots,N_{K-1}$):

$$Condition\ 21: \quad \begin{array}{c} IF\ \left(D_{K,i}(\boldsymbol{x}_K) \geq \frac{1}{1+n^2}\right) \\ THEN\ \left(\begin{array}{c}\Xi_i\ and\ the\ corresponding\ fuzzy\ rule \\ will\ be\ replaced\ by\ the\ new\ one\end{array}\right) \end{array}, \tag{4.1}$$

where $D_{K,i}(\boldsymbol{x}_K)$ is the local unimodal density calculated per data cloud:

$$D_{K,i}(\boldsymbol{x}_K) = \frac{1}{1+\frac{S_{K-1,i}^2\left\|\boldsymbol{x}_K-\boldsymbol{p}_{K-1,i}\right\|^2}{\left(S_{K-1,i}+1\right)\left(S_{K-1,i}X_{K-1,i}+\left\|\boldsymbol{x}_{K+1}\right\|^2\right)-\left\|\boldsymbol{x}_K+S_{K-1,i}\boldsymbol{p}_{K-1,i}\right\|^2}}. \tag{4.2}$$

The rationale to consider $D_{K,i}(\boldsymbol{x}_K) \geq \frac{1}{1+n^2}$ comes from the well-known Chebyshev inequality in the form of unimodal density (equation (2.43)). Here $n = 0.5$ is used, which is equivalent to $D_{K,i}(\boldsymbol{x}_K) \geq 0.8$ for $\boldsymbol{x}_K$ is less than $\sigma/2$ away from the focal point of the *i*th data cloud.

If only Condition 3 is satisfied and Condition 21 is not met, a new data cloud with focal point $\boldsymbol{x}_K$ is added to the system:

$$\begin{array}{c} N_K \leftarrow N_{K-1}+1; \quad \Xi_{N_K} \leftarrow \{\boldsymbol{x}_K\}; \quad \boldsymbol{p}_{K,N_K} \leftarrow \boldsymbol{x}_K; \\ X_{K,N_K} \leftarrow \|\boldsymbol{x}_K\|^2; \quad S_{K,N_K} \leftarrow 1. \end{array} \tag{4.3}$$

In contrast, if Conditions 3 and 21 are both satisfied, then the existing overlapping data cloud (assuming the *i*th one) is being replaced by a new one with the focal point $\boldsymbol{x}_K$ as follows:

$$N_K \leftarrow N_{K-1}; \quad \Xi_i \leftarrow \{\Xi_i, \boldsymbol{x}_K\}; \quad S_{K,i} \leftarrow \left\lceil \frac{S_{K-1,i}+1}{2} \right\rceil;$$
$$\boldsymbol{p}_{K,i} \leftarrow \frac{\boldsymbol{p}_{K-1,i}+\boldsymbol{x}_K}{2}; \quad X_{K,i} \leftarrow \frac{X_{K-1,i}+\|\boldsymbol{x}_K\|^2}{2}. \tag{4.4}$$

Equation (4.4) can stop the ALMMO system from discarding the previously collected information too fast because the new data cloud may be initialised by an abnormal data sample.

If Condition 3 (equation (3.10)) is not satisfied, then the algorithm continues by finding the nearest data cloud Ξ_n to $\boldsymbol{x}_K$, which is identified by equation (3.12). The corresponding meta-parameters of the system and Ξ_n are updated as follows:

$$N_K \leftarrow N_{K-1}; \quad \Xi_n \leftarrow \{\Xi_n, \boldsymbol{x}_K\}; \quad S_{K,n} \leftarrow S_{K-1,n} + 1;$$
$$\boldsymbol{p}_{K,n} \leftarrow \frac{S_{K-1,n}}{S_{K,n}} \boldsymbol{p}_{K-1,n} + \frac{1}{S_{K,n}} \boldsymbol{x}_K; \quad X_{n,K} \leftarrow \frac{S_{K-1,n}}{S_{K,n}} X_{K-1,n} + \frac{1}{S_{K,n}} \|\boldsymbol{x}_K\|^2. \tag{4.5}$$

The meta-parameters of other data clouds stay the same for the next processing cycle. In ALMMO, each data cloud (and the respective focal point) is used as the basis to formulate the antecedent (IF) part of the AnYa type fuzzy rules.

4.1.2.3. Online Quality Monitoring

Since the ALMMO system is for processing streaming data, monitoring the quality of the dynamically evolving structure is necessary in order to guarantee the computation- and memory-efficiency. The quality of the fuzzy rules within the ALMMO system can be characterised by their utility [55]. In ALMMO, utility, $\eta_{K,i}$ of the *i*th data cloud accumulates the weight of the corresponding fuzzy rule contribution to the overall output (activation level) during the life of the rule (from the time instance at which the data cloud was generated till the current time instance). It is the measure of importance of the respective fuzzy rule compared to others ($i = 1,2, \dots, N_K$):

$$\eta_{K,i} = \frac{1}{K-I_i} \sum_{l=I_i}^{K} \lambda_{l,i} = \frac{1}{K-I_i} \sum_{l=I_i}^{K} \frac{D_{l,i}(x_l)}{\sum_{j=1}^{N_l} D_{l,j}(x_l)}; \quad \eta_{I_i,i} = 1, \tag{4.6}$$

where I_i is the time instance at which the *i*th data cloud is established; $\lambda_{l,i} = \frac{D_{l,i}(x_l)}{\sum_{j=1}^{N_l} D_{l,j}(x_l)}$ is the activation level of the *i*th data cloud at the *l*th time instance.

The rule base can be simplified according to Condition 22 by removing the data clouds and their corresponding fuzzy rules with low utility [55], [60]:

$$Condition\ 22: \quad \begin{matrix} IF\ (\eta_{K,i} < \eta_o) \\ THEN\ \left(\begin{matrix}\Xi_i\ and\ the\ corresponding\ fuzzy\ rule \\ is\ removed\end{matrix}\right)\end{matrix}, \tag{4.7}$$

where η_o is a small tolerance constant ($\eta_o = 0.1$ is used).

If Ξ_i satisfies Condition 22, the respective fuzzy rule will be removed from the rule base and its consequent parameters $\boldsymbol{a}_{K,i}$ and $\boldsymbol{\Theta}_{K,i}$ are deleted as well.

4.1.3. Parameter Identification

In this subsection, the parameter identification process of the ALMMO system is described.

4.1.3.1. Parameter Initialisation

As the first data cloud of the system is initialised by the first data sample, $\boldsymbol{x}_1$, the corresponding consequent parameters of the first fuzzy rule within the rule base are set up as: $\boldsymbol{\Theta}_{1,1} \leftarrow \Omega\mathbf{I}_{(M+1)\times(M+1)}$ and $\boldsymbol{a}_{1,1} \leftarrow \mathbf{0}_{1\times(M+1)}$.

4.1.3.2. Parameter Update

If a new fuzzy rule is added by the newly arrived data sample $\boldsymbol{x}_K$ during the structure identification stage, the corresponding consequent parameters are added as follows:

$$\boldsymbol{a}_{K-1,N_K} \leftarrow \frac{1}{N_{K-1}} \sum_{j=1}^{N_{K-1}} \boldsymbol{a}_{K-1,j}\ ; \quad \boldsymbol{\Theta}_{K-1,N_K} \leftarrow \Omega\mathbf{I}_{(M+1)\times(M+1)}. \tag{4.8}$$

If an old fuzzy rule (denoted as the j^{th} rule) is replaced by a new one when Conditions 3 and 21 are both satisfied, the new rule will inherit the consequent parameters of the old one.

After the structure of both the antecedent and consequent parts of the ALMMO system is revised, the FWRLS [11] approach is used to update the consequent parameters ($\boldsymbol{a}_{K,i}$ and $\boldsymbol{\Theta}_{K,i}$, $i = 1,2, \dots, N_K$) of each fuzzy rule locally as equation (2.69).

4.1.3.4. Online Input Selection

In many practical cases, there are a number of inter-correlated attributes within the data. Therefore, it is of great importance to introduce the online input selection, which can further eliminates the waste of the computation- and memory-resources and improve the overall performance.

In this section, Condition 23 is used to deal with this:

$$\mathit{Condition}\ 23: \quad \begin{matrix} IF\ \left(\omega_{K,i,j} < \frac{\varepsilon}{N_K}\sum_{i=1}^{N_K}\omega_{K,i,j}\right) \\ THEN\ (remove\ the\ j^{th}\ set\ from\ the\ i^{th}\ fuzzy\ rule) \end{matrix}, \tag{4.9}$$

where $j = 1,2, \dots M$, $i = 1,2, \dots, N_K$; $\omega_{K,i,j}$ is the normalised accumulated sum of parameter values at the time instance K:

$$\omega_{K,i,j} = \frac{\rho_{K,i,j}}{\sum_{j=1}^{M}\rho_{K,i,j}} = \frac{\sum_{t=I_i}^{K}|a_{t,i,j}|}{\sum_{j=1}^{M}\sum_{t=I_i}^{K}|a_{t,i,j}|}, \tag{4.10}$$

where $\rho_{K,i,j} = \sum_{t=I_i}^{K}|a_{t,i,j}|$ is the accumulated sum of parameter values; ε is a constant, $\varepsilon \in [0.03, 0.05]$.

If the j^{th} set of the i^{th} fuzzy rule meets Condition 23, it is removed from the rule and the corresponding column and row of the covariance matrix $\boldsymbol{\Theta}_{K,i}$.

4.1.4. System Output Generation

Once the ALMMO system has updated its structure and parameters, it is ready for the next data sample. When the next data sample $\boldsymbol{x}_K$ $(K \leftarrow K + 1)$ comes, the system output is generated as:

$$y_K = \sum_{j=1}^{N_{K-1}}\lambda_{K,j}[1, \boldsymbol{x}_K^T]\boldsymbol{a}_{K-1,j}. \tag{4.11}$$

After the system performs the prediction, it will update its structure and parameters based on $\boldsymbol{x}_K$ and the prediction error.

The main procedure of the learning process of the ALMMO algorithm is summarised in the form of a flowchart presented in Figure 19.

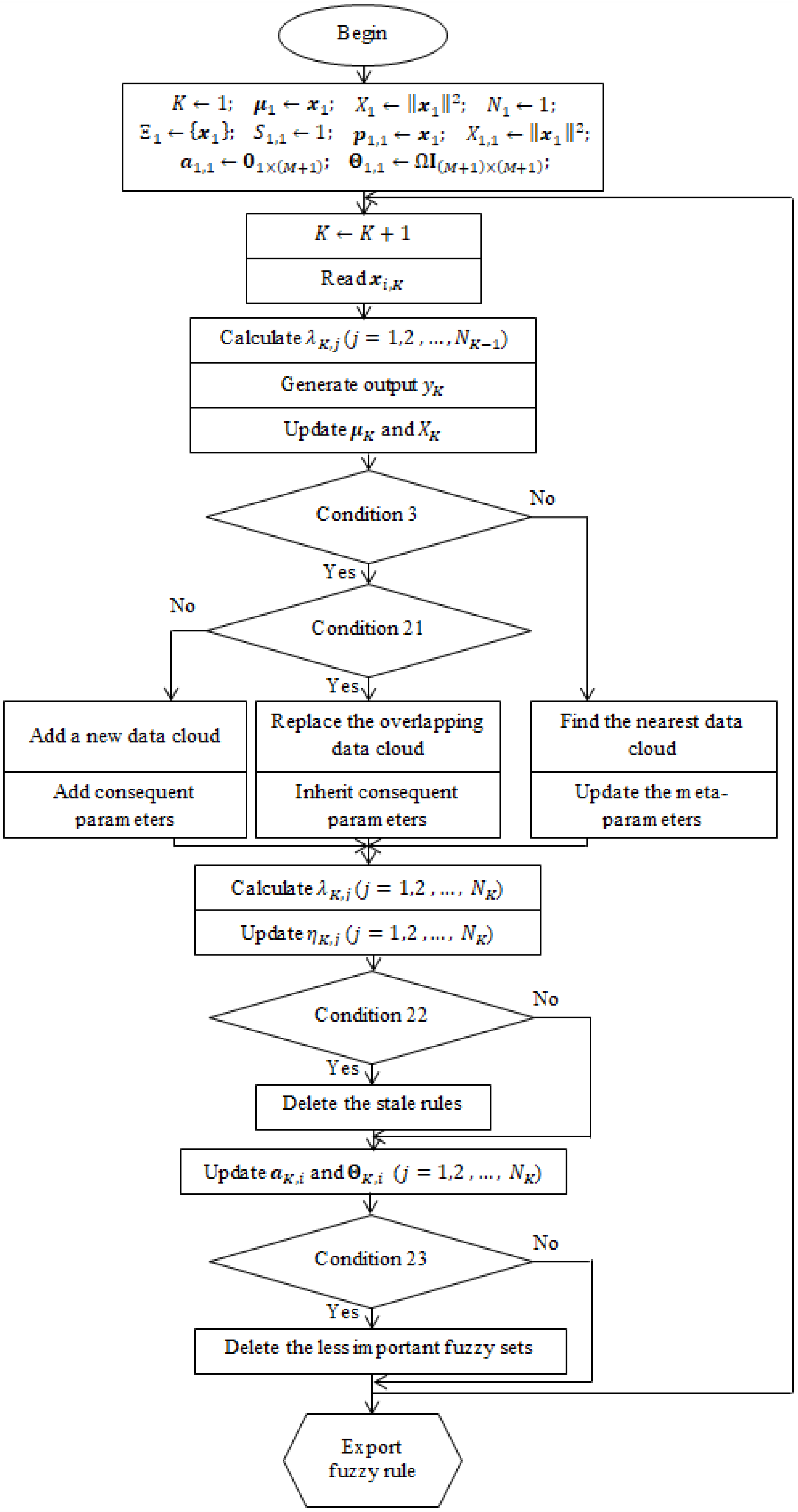

Begin
$K \leftarrow 1$; $\boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_1$; $X_1 \leftarrow \|\boldsymbol{x}_1\|^2$; $N_1 \leftarrow 1$;
$\Xi_1 \leftarrow \{\boldsymbol{x}_1\}$; $S_{1,1} \leftarrow 1$; $\boldsymbol{p}_{1,1} \leftarrow \boldsymbol{x}_1$; $X_{1,1} \leftarrow \|\boldsymbol{x}_1\|^2$;
$\boldsymbol{a}_{1,1} \leftarrow \mathbf{0}_{1\times(M+1)}$; $\boldsymbol{\Theta}_{1,1} \leftarrow \Omega\mathbf{I}_{(M+1)\times(M+1)}$;
$K \leftarrow K+1$
Read $\boldsymbol{x}_{i,K}$
Calculate $\lambda_{K,j}$ $(j = 1,2,\ldots,N_{K-1})$
Generate output y_K
Update $\boldsymbol{\mu}_K$ and X_K
Condition 3
No
Yes
Condition 21
No
Yes
Add a new data cloud
Add consequent parameters
Replace the overlapping data cloud
Inherit consequent parameters
Find the nearest data cloud
Update the meta-parameters
Calculate $\lambda_{K,j}$ $(j = 1,2,\ldots,N_K)$
Update $\eta_{K,j}$ $(j = 1,2,\ldots,N_K)$
Condition 22
No
Yes
Delete the stale rules
Update $\boldsymbol{a}_{K,i}$ and $\boldsymbol{\Theta}_{K,i}$ $(j = 1,2,\ldots,N_K)$
Condition 23
No
Yes
Delete the less important fuzzy sets
Export fuzzy rule

Figure 19. Main procedure of the learning process of the ALMMO system.

4.2. Zero Order Autonomous Learning Multi-Model Classifier

The zero order autonomous learning multi-model (ALMMO-0) classifier [35] is introduced on the basis of the 0-order AnYa type fuzzy rule-based (FRB) systems [55], [60] in a multiple-model architecture [161]. This classifier is nonparametric, non-iterative and fully autonomous. There is no need to train any parameters due to its feedforward structure. The proposed classifier automatically identifies the focal points from the empirically observed data and forms data clouds resembling Voronoi tessellation [64] per class. Then, sub-classifiers corresponding to different classes are built in a form of a set of AnYa type of fuzzy rules from the non-parametric data clouds. For a new data sample, each AnYa FRB sub-classifier generates a score of confidence objectively and the label is assigned to the new data sample based on the "winner takes all" rule. The proposed ALMMO-0 classifier learns from the data and conducts classification based on very fundamental principles, a variety of modifications and extensions can further be done, i.e. using the fuzzy rules with 1st order consequent part.

4.2.1. Multiple-Model Architecture

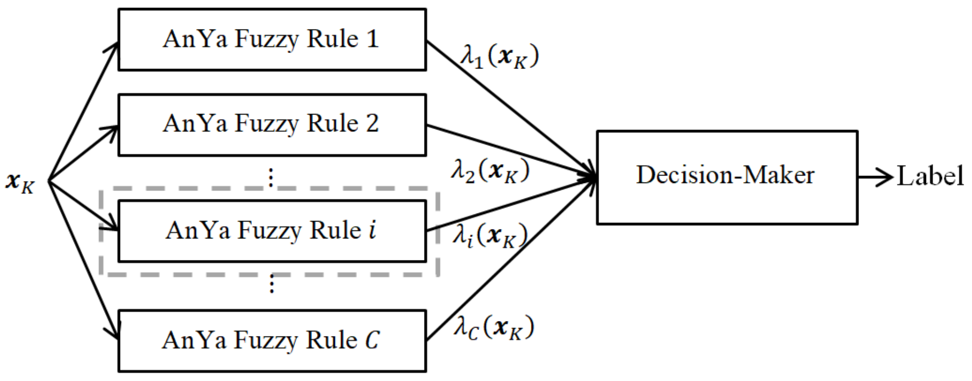
The multiple-model architecture is based on the 0-order AnYa type fuzzy rules [55], [60]. An illustrative diagram of the classifier with the multiple-model architecture is depicted in Figure 20. Figure 20(a) illustrates the multiple-model structure of the classifier, and Figure 20(b) is the zoom-in structure of a 0-order AnYa type fuzzy rule.

It is demonstrated in Figure 20 that, each time a new data sample $\boldsymbol{x}_K$ is coming, it is sent to C 0-order AnYa type fuzzy rules corresponding to C different classes in the dataset. Each fuzzy rule can be viewed as a combination of a large number of singleton fuzzy rules that are built upon prototypes identified from data samples of the corresponding class connected by the logic "OR" operator ($i = 1,2, \dots, C$):

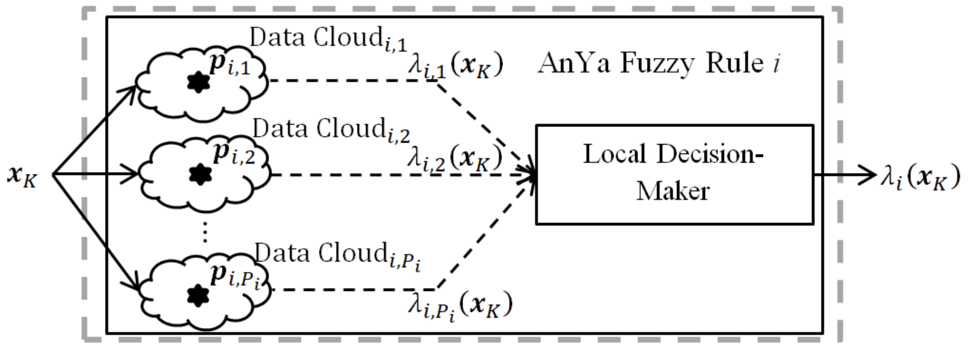
$$\begin{gathered} IF\ (\boldsymbol{x}\sim\boldsymbol{p}_{i,1})\ AND\ (\boldsymbol{x}\sim\boldsymbol{p}_{i,2})\ AND\ \dots\ AND\ (\boldsymbol{x}\sim\boldsymbol{p}_{i,P_i}) \\ THEN\ (Class\ i) \end{gathered}, \tag{4.12}$$

where $\boldsymbol{p}_{i,j}$ is the j^{th} prototype of the i^{th} fuzzy rule; P_i is the number of identified prototypes.

The "winner takes all" principle is firstly used to select out the most similar prototype with $\boldsymbol{x}_K$ in terms of the degree of confidence from each fuzzy rule. Then, the "winner takes all" principle is used again to assign $\boldsymbol{x}_K$ to the class that it is most likely to be associated with.


AnYa Fuzzy Rule 1
AnYa Fuzzy Rule 2
AnYa Fuzzy Rule i
AnYa Fuzzy Rule C
$\boldsymbol{x}_K$
$\lambda_1(\boldsymbol{x}_K)$
$\lambda_2(\boldsymbol{x}_K)$
$\lambda_i(\boldsymbol{x}_K)$
$\lambda_C(\boldsymbol{x}_K)$
Decision-Maker
Label

(a) The multiple-model architecture



(b) Zoom-in structure of the *i*th fuzzy rule

Figure 20. Multiple-model architecture of ALMMO-0.

4.2.2. Learning Process

In this subsection, the learning process of ALMMO-0 classifier is described. Due to the multiple-model architecture of the classifier, each AnYa fuzzy rule is trained in parallel with the data samples from the corresponding class (one rule per class). Assuming the *i*th fuzzy rule, the detailed learning process is as follows.

For each newly arrived data sample of the *i*th class, denoted by $\boldsymbol{x}_{i,K_i}$, it will be normalised by its norm, namely:

$$\boldsymbol{x}_{i,K_i} \leftarrow \boldsymbol{x}_{i,K_i} / \left\| \boldsymbol{x}_{i,K_i} \right\|. \tag{4.13}$$

This type of normalisation can convert the Euclidean distance between different data samples into cosine dissimilarity, which enhances the classifier's ability for high-dimensional data processing [33].

The AnYa fuzzy rule is initialised by the first data sample $\boldsymbol{x}_{i,1}$ with its global parameters set as: $P_i \leftarrow 1$; $\boldsymbol{\mu}_i \leftarrow \boldsymbol{x}_{i,1}$; $X_i \leftarrow 1$. And the local meta-parameters of the first data cloud are set as $\Xi_{i,1} \leftarrow \{\boldsymbol{x}_{i,1}\}$; $\boldsymbol{p}_{i,1} \leftarrow \boldsymbol{x}_{i,1}$; $S_{i,1} \leftarrow 1$; $r_{i,1} \leftarrow r_o$, where $r_{i,1}$ is the radius of the influence area; r_o is a small value to stabilize the initial status of the new-born data clouds, $r_o = \sqrt{2(1-\cos(30^o))}$ is used by default [33]. It has to be stressed that, r_o is not a problem-specific parameter and requires no prior knowledge to decide. It is for preventing the new-born data clouds from attracting data samples that are not close enough. It defines a degree of closeness that is interesting and distinguishable. The AnYa fuzzy rule is firstly initialised as:

$$IF\ (\boldsymbol{x} \sim \boldsymbol{p}_{i,1}) \quad THEN\ (Class\ i). \tag{4.14}$$

For each newly arrived data sample ($K_i \leftarrow K_i + 1$), firstly, the global mean $\boldsymbol{\mu}_i$ of the *i*th class is updated by $\boldsymbol{x}_{i,K_i}$ using equation (3.26). There is no need to update the average scalar product anymore because $X_i = \|\boldsymbol{x}_{i,K_i}\| = 1$ ($\boldsymbol{x}_{i,K_i}$ has been normalised at first). The unimodal densities of the data sample $\boldsymbol{x}_{i,K}$ and all the identified focal points of the *i*th class, denoted as $\boldsymbol{p}_{i,j}$ ($j = 1,2,\ldots,P_i$) are calculated using equations (2.18), (2.24) and (2.25).

Then, Condition 3 (equation (3.10)) is checked to see whether $\boldsymbol{x}_{i,K_i}$ will generate a new data cloud and becomes a new prototype added into the fuzzy rule [55]. If Condition 3 (equation (3.10)) is triggered, a new data cloud is being formed around $\boldsymbol{x}_{i,K_i}$ and its parameters are being updated as follows:

$$P_i \leftarrow P_i + 1; \quad \Xi_{i,P_i} \leftarrow \{\boldsymbol{x}_{i,K_i}\}; \quad \boldsymbol{p}_{i,P_i} \leftarrow \boldsymbol{x}_{i,K_i}; \quad S_{i,P_i} \leftarrow 1; \quad r_{i,P_i} \leftarrow r_o, \tag{4.15}$$

and a new prototype $\boldsymbol{p}_{i,P_i}$ is added to the fuzzy rule as initialised in equation (4.15).

If Condition 3 (equation (3.10)) is not satisfied, then the algorithm continues by finding the nearest data cloud $\Xi_{i,n}$ to $\boldsymbol{x}_{i,K_i}$, which is achieved with equation (3.12).

Before $\boldsymbol{x}_{i,K_i}$ is assigned to the nearest data cloud, Condition 24 is being checked to see whether $\boldsymbol{x}_{i,K_i}$ is close to the data cloud $\Xi_{i,n}$ or not:

$$Condition\ 24{:}\quad \begin{matrix} IF\ (\|\boldsymbol{x}_{i,K_i} - \boldsymbol{p}_{i,n}\| \le r_{i,n}) \\ THEN\ (\Xi_{i,n} \leftarrow \{\Xi_{i,n}, \boldsymbol{x}_{i,K_i}\}) \end{matrix}. \tag{4.16}$$

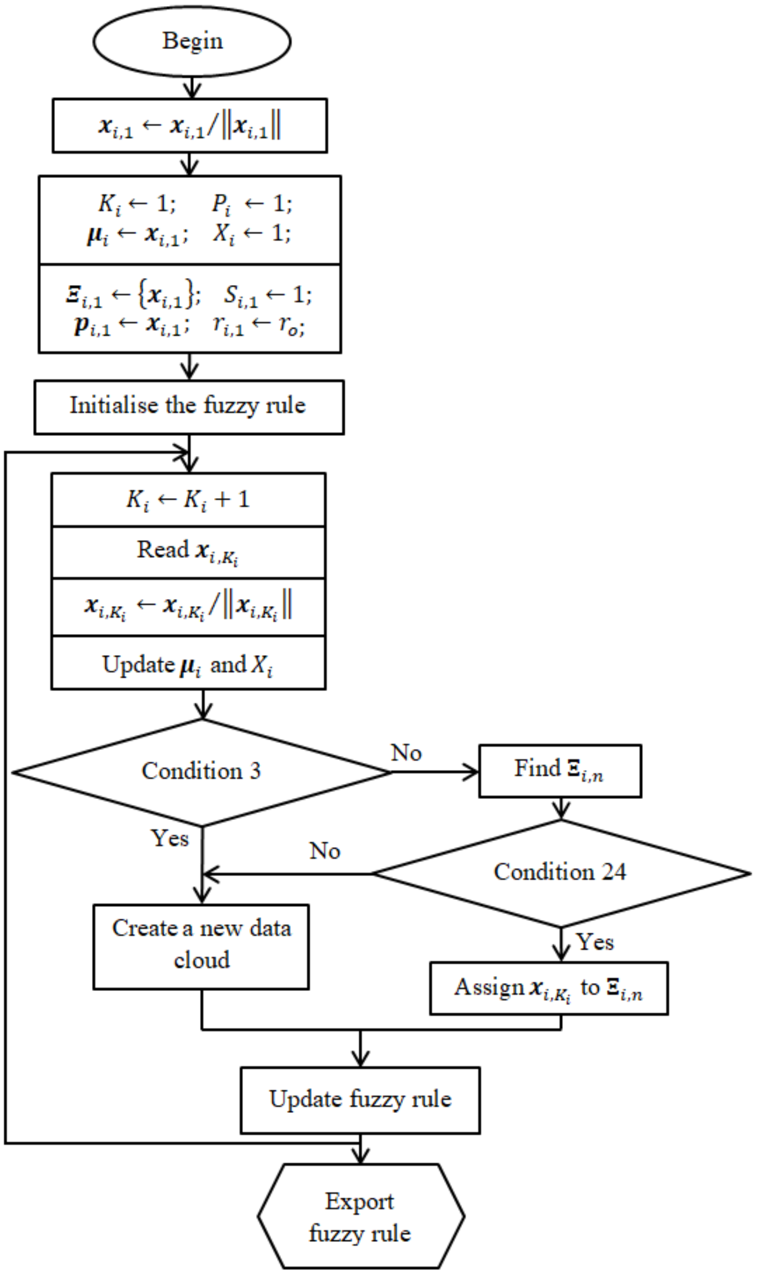
If Condition 24 is satisfied, the meta-parameters of the nearest data cloud $\Xi_{i,n}$ are updated as follows:

$$S_{i,n} \leftarrow S_{i,n} + 1; \quad \boldsymbol{p}_{i,n} \leftarrow \frac{S_{i,n}-1}{S_{i,n}}\boldsymbol{p}_{i,n} + \frac{1}{S_{i,n}}\boldsymbol{x}_{i,K_i}; \quad r_{i,n} \leftarrow \sqrt{0.5\left(r_{i,n}^2 + \left(1 - \|\boldsymbol{p}_{i,n}\|^2\right)\right)}, \tag{4.17}$$

and the fuzzy rule is updated accordingly. On the contrary, if Condition 24 is not met, a new data cloud is formed around $\boldsymbol{x}_{i,K_i}$ using equation (4.15) and a new prototype $\boldsymbol{p}_{i,P_i}$ is added to the fuzzy rule.

For the data clouds that do not receive new members, the parameters of the other data clouds stay the same for the next processing cycle.

The main procedure of the learning process of ALMMO-0 classifier is depicted in Figure 21 in the form of a flowchart.



Begin
$\boldsymbol{x}_{i,1} \leftarrow \boldsymbol{x}_{i,1}/\|\boldsymbol{x}_{i,1}\|$
$K_i \leftarrow 1$; $P_i \leftarrow 1$; $\boldsymbol{\mu}_i \leftarrow \boldsymbol{x}_{i,1}$; $X_i \leftarrow 1$;
$\Xi_{i,1} \leftarrow \{\boldsymbol{x}_{i,1}\}$; $S_{i,1} \leftarrow 1$; $\boldsymbol{p}_{i,1} \leftarrow \boldsymbol{x}_{i,1}$; $r_{i,1} \leftarrow r_o$;
Initialise the fuzzy rule
$K_i \leftarrow K_i + 1$
Read $\boldsymbol{x}_{i,K_i}$
$\boldsymbol{x}_{i,K_i} \leftarrow \boldsymbol{x}_{i,K_i}/\|\boldsymbol{x}_{i,K_i}\|$
Update $\boldsymbol{\mu}_i$ and X_i
Condition 3
No
Find $\Xi_{i,n}$
Yes
No
Condition 24
Create a new data cloud
Yes
Assign $\boldsymbol{x}_{i,K_i}$ to $\Xi_{i,n}$
Update fuzzy rule
Export fuzzy rule

Figure 21. Main procedure of the learning process of ALMMO-0 classifier.

4.2.3. Validation Process

The main procedure of the validation process of the ALMMO-0 classifier is as follows.

For each validation data sample, denoted by $\boldsymbol{x}$, it is sent to the C fuzzy rules corresponding to the C different classes, and each fuzzy rule will generate a score of confidence by equation (4.18) following the "winner takes all" principle:

$$\lambda_i(\boldsymbol{x}) = \max_{j=1,2,\dots,P_i}\left(\lambda_{i,j}(\boldsymbol{x})\right) = \max_{j=1,2,\dots,P_i}\left(e^{-\|\boldsymbol{x}-\boldsymbol{p}_{i,j}\|^2}\right). \tag{4.18}$$

The label of $\boldsymbol{x}$, denoted by $y(x)$, is decided by the "winner takes all" principle again:

$$y(\boldsymbol{x}) = \text{argmax}_{i=1,2,\dots,C}\left(\lambda_i(\boldsymbol{x})\right). \tag{4.19}$$

4.3. Self-Organising Fuzzy Logic Classifier

In this section, the self-organising fuzzy logic (SOFL) classifier is presented [36]. The SOFL approach is grounded at the Empirical Data Analytics (EDA) computational framework [25]–[27] and the autonomous data-driven clustering techniques [28], [29]. The SOFL classifier has two training stages, 1) offline and 2) online. During the offline stage, it learns from the static data to establish a stable 0-order AnYa type fuzzy rule-based (FRB) system [54], [60] and, during the online training stage, the FRB system identified during the offline training will be updated subsequently to follow the possible drifts and/or shifts in the data pattern [117]. The SOFL classifier only keeps the key meta-parameters in memory and is a one-pass type during its online training stage; therefore, it is very suitable for large-scale streaming data processing.

Most importantly, the SOFL classifier is nonparametric in the sense that no parameters or models are imposed for data generation [36]. Employing the EDA quantities as described in section 2.1.3, the SOFL classifier is able to objectively disclose the ensemble properties and mutual distributions of the streaming data based on the empirically observed data samples and all the meta-parameters of the classifier are directly derived from the data without prior knowledge [25]–[27].

The SOFL classifier keeps the advantage of objectiveness of the data-driven approaches, and, at the same time, puts the users "in the driving seat" by letting users to decide the level of granularity and the type of distance/dissimilarity measure for the classifier. However, it has to be stressed that there is no requirement for prior knowledge to decide the level of granularity and it can be given merely based on the preferences of users. Higher level of granularity leads to a classifier with fine details but with a risk of overfitting. A lower level of granularity, instead, gives users a classifier trained coarsely but with higher computational efficiency, generalisation and less memory requirement. The classifier is always guaranteed to be meaningful due to its data-driven nature. The choice of the type of distance/dissimilarity measure further allows more freedom for the users and also makes the SOFL approach highly adaptive to various applications, e.g. natural language processing. In addition, the SOFL classifier can also provide default level of granularity and distance measure options for the less experienced users.

In the following two subsections, the main procedures of the training process (both offline and online) and validation process of the SOFL classifier are presented separately.

4.3.1. Offline Training

The offline training process of the SOFL classifier is category-wise, the classifier will identify prototypes from each class separately and form a 0-order AnYa type fuzzy rule based on the identified prototypes per class (in the form of equation (4.12)). The training processes of the fuzzy rules of different classes will not influence each other. The diagram of the SOFL classifier for offline training is depicted in Figure 22 [36].

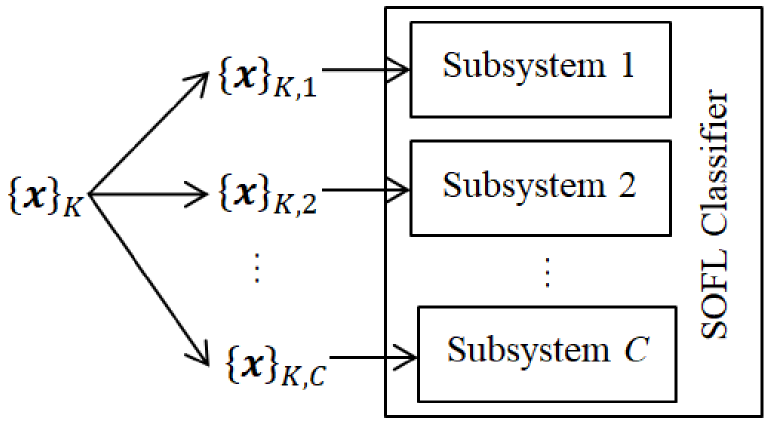


Figure 22. Diagram of the SOFL classifier for offline training.

4.3.1.1. Main Procedure

In the rest of this subsection, it is assumed that the training process is conducted on the data samples of the i^{th} class ($i = 1,2,\dots,C$), denoted as $\{\boldsymbol{x}\}_i = \{\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, \dots, \boldsymbol{x}_{i,K_i}\}$ ($\{\boldsymbol{x}\}_i \subset \{\boldsymbol{x}\}$), and the corresponding unique data sample set $\{\boldsymbol{u}\}_i = \{\boldsymbol{u}_{i,1}, \boldsymbol{u}_{i,2}, \dots, \boldsymbol{u}_{i,N_i}\}$ and the frequencies of occurrence $\{f\}_i = \{f_{i,1}, f_{i,2}, \dots, f_{i,N_i}\}$, where K_i is the number of data samples with $\{\boldsymbol{x}\}_i$, N_i is the number of unique data samples of the i^{th} class. Considering all the classes, we have $\sum_{i=1}^{C} K_i = K$ and $\sum_{i=1}^{C} N_i = N$.

In the SOFL approach, prototypes are identified based on the densities and the mutual distributions of the data samples. Firstly, multimodal densities $D_K^G(\boldsymbol{u}_{i,j})$ ($j = 1,2,\dots,N_i$) at all the unique data samples within $\{\boldsymbol{u}\}_i$ are calculated using equation (2.19). After this, the unique data samples are ranked in a list, denoted by $\{\boldsymbol{z}\}_i$, in terms of their mutual distances and values of multimodal density using the same approach as described in section 3.3.1.1. Then, porotypes of $\{\boldsymbol{u}\}_i$ are identified using the same approach as described in section 3.3.1.2, denoted by $\{\boldsymbol{u}^*\}_i$.

After $\{\boldsymbol{u}^*\}_i$ are identified, the filtering operation starts. The prototypes are firstly used to attract nearby data samples to form data clouds [60] resembling Voronoi tessellation [64] using equation (3.12).

With the data clouds formed around the existing prototypes $\{\boldsymbol{u}^*\}_i$ denoted by $\Xi_{i,j}$ ($j = 1,2,\dots,P_i$, P_i is the number of prototypes of the *i*th class), one can obtain the centres of the data clouds denoted by $\{\boldsymbol{\mu}\}_i$ and the multimodal densities at the centres are calculated using equation (2.19) as $D_K^G(\boldsymbol{\mu}_{i,j}) = S_{i,j}D_K(\boldsymbol{\mu}_{i,j})$, where $\boldsymbol{\mu}_{i,j} \in \{\boldsymbol{\mu}\}_i$; $S_{i,j}$ is the support (number of members) of $\Xi_{i,j}$.

Then, for each data cloud, assuming $\Xi_{i,j}$ ($j = 1,2,\dots,P_i$), the collection of the centres of its neighbouring data clouds, denoted by $\{\boldsymbol{\mu}\}_j^n$ are identified using the following principle:

$$Condition\ 25\text{:}\quad \begin{matrix} IF\ \left(d^2(\boldsymbol{\mu}_{i,j},\boldsymbol{\mu}_{i,k}) \leq \mathcal{L}_i^G\right) \\ THEN\ \left(\boldsymbol{\mu}_{i,k} \in \{\boldsymbol{\mu}\}_j^n\right) \end{matrix}, \tag{4.20}$$

where $\boldsymbol{\mu}_{i,j},\boldsymbol{\mu}_{i,k} \in \{\boldsymbol{\mu}\}_i$ and there is $\boldsymbol{\mu}_{i,j} \neq \boldsymbol{\mu}_{i,k}$; $\mathcal{L}_i^G$ is defined as the average radius of local influential area around each data sample, which is corresponding to the Gth ($G = 1,2,3,\dots$) level of granularity and is derived from the data of the *i*th class based on the users' choice:

Under the 1st level of granularity ($G = 1$), the average radius of local influential area, denoted by $\mathcal{L}_i^G$ around each prototype of the *i*th class is defined as follows:

$$\mathcal{L}_i^1 = \frac{\sum_{\boldsymbol{x},\boldsymbol{z}\in\{\boldsymbol{x}\}_i,\boldsymbol{x}\neq\boldsymbol{z},d^2(\boldsymbol{x},\boldsymbol{z})\leq\bar{d}_i^2} d^2(\boldsymbol{x},\boldsymbol{z})}{T_i^1}, \tag{4.21}$$

where T_i^1 is the number of the pairs of data samples between which the distance is smaller than the average distance, $\bar{d}_i$.

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

$$\mathcal{L}_i^G = \frac{\sum_{x,y \in \{x\}_i, x \neq y, d^2(x,y) \leq \mathcal{L}_i^{G-1}} d^2(\boldsymbol{x}, \boldsymbol{y})}{T_i^G}, \tag{4.22}$$

where $\mathcal{L}_i^{G-1}$is the average radius corresponding to $(G-1)^{th}$ level of granularity; T_i^G is the number of the pairs of data samples between which the distance is smaller than $\mathcal{L}_i^{G-1}$.

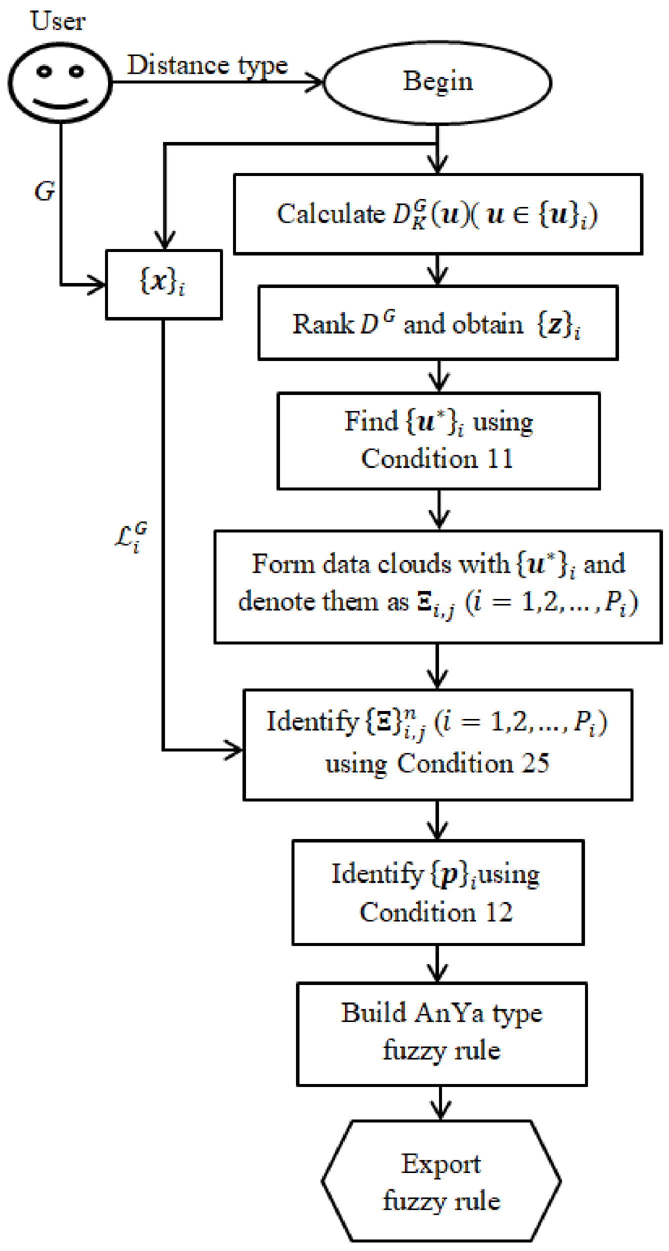
Compared with the traditional approaches, there are strong advantages in deriving the local information in this way. Firstly, $\mathcal{L}_i^G$ is guaranteed to be valid all the time. Defining the threshold or hard-coding mathematical principles in advance may suffer from various problems, i.e. prior knowledge is often unavailable, hard-coded principles are too sensitive to the nature of the data. The performance of the two approaches is often not guaranteed. In contrast, $\mathcal{L}_i^G$ is derived from the data directly and is always meaningful. There is no need for prior knowledge of data sets/streams, and the level of granularity used by the SOFL classifier can be decided merely based on the preferences of the users. Moreover, users are allowed to have freedom to make choices, but at the same time, are not overloaded. Finally, one can always adapt the classifier by changing the level of granularity based on the specific needs. Some problems rely heavily on fine details, while others may need generality only.

In general, the higher level of granularity is chosen, the more fine details (more prototypes) the SOFL classifier extracts from the data, and the classifier achieves a higher performance. At the same time, the SOFL classifier may consume more computational and memory resources, and overfitting may also appear. On the contrary, with low level of granularity, the SOFL classifier only learns the coarse information from training. Although the classifier will be more computationally efficient, its performance may be influenced due to the loss of fine information from the data.

Finally, the most representative prototypes of the i^{th} class, denoted by $\{\boldsymbol{p}\}_i$, are selected out from the centres of the existing data clouds satisfying Condition 12 (equation (3.47)) and one can build the AnYa type fuzzy rule in the same form as equation (4.12), where P_i is the number of prototypes in $\{\boldsymbol{p}\}_i$.

4.3.1.2. Algorithm Summary

The main procedure of the offline training process of the SOFL classifier is summarised in the form of a flowchart in Figure 23.



User
Distance type
Begin
G
Calculate $D_K^G(\boldsymbol{u})(\boldsymbol{u} \in \{\boldsymbol{u}\}_i)$
$\{\boldsymbol{x}\}_i$
Rank D^G and obtain $\{\boldsymbol{z}\}_i$
Find $\{\boldsymbol{u}^*\}_i$ using Condition 11
$\mathcal{L}_i^G$
Form data clouds with $\{\boldsymbol{u}^*\}_i$ and denote them as $\Xi_{i,j}$ $(i = 1,2,\dots,P_i)$
Identify $\{\Xi\}_{i,j}^n$ $(i = 1,2,\dots,P_i)$ using Condition 25
Identify $\{\boldsymbol{p}\}_i$using Condition 12
Build AnYa type fuzzy rule
Export fuzzy rule

Figure 23. Main procedure of the offline training process of SOFL classifier.

4.3.2. Online Self-Evolving Training

During the online training stage, the SOFL classifier continues to update its system parameters and structure with the streaming data on a sample-by-sample basis. Furthermore, because the EDA quantities [25]–[27] employed by the SOFL classifier can be updated recursively, it can be one-pass type, and its computation- and memory-efficiency is also guaranteed.

4.3.2.1. Main Procedure

In this subsection, we assume that the training process of the SOFL classifier with the static dataset $\{\boldsymbol{x}\}_K$ has been finished and new data samples start to arrive in a data stream form. Similar to the offline training stage, during the online training stage, the fuzzy rules of different classes are updated separately. During the online stage, recursive calculation expressions of the EDA quantities with Euclidean distance are used. Nonetheless, other types of distance/dissimilarity measures can be considered as well.

Assuming at the next time instance, a new data sample of the i^{th} class arrives ($K \leftarrow K + 1$, $K_i \leftarrow K_i + 1$) and the data sample is denoted as $\boldsymbol{x}_{i,K_i}$. The SOFL classifier firstly updates the global meta-parameters $\boldsymbol{\mu}_{i,K_i}$ and X_{i,K_i} using equations (2.26) and (2.27), where $\boldsymbol{\mu}_{i,K_i}$ and X_{i,K_i} are the mean and average scalar product of the data samples $\{\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, \ldots, \boldsymbol{x}_{i,K_i-1}, \boldsymbol{x}_{i,K_i}\}$.

The average radius of local areas of influence, $\mathcal{L}_i^G$ is updated afterwards in a recursive way based on the ratio between the average distances of the data samples at $(K_i - 1)^{th}$ and ${K_i}^{th}$ instances, respectively as:

$$\mathcal{L}_i^G \leftarrow \frac{\frac{1}{K_i^2}\sum_{l=1}^{K_i} q_K(\boldsymbol{x}_{i,l})}{\frac{1}{(K_i-1)^2}\sum_{l=1}^{K_i-1} q_{K-1}(\boldsymbol{x}_{i,l})} \cdot \mathcal{L}_i^G = \frac{X_{i,K_i} - \left\|\boldsymbol{\mu}_{i,K_i}\right\|^2}{X_{i,K_i-1} - \left\|\boldsymbol{\mu}_{i,K_i-1}\right\|^2} \cdot \mathcal{L}_i^G. \tag{4.23}$$

Instead of deriving $\mathcal{L}_i^G$ in an offline way as described in the previous subsection, equation (4.23) largely reduces the computational complexity and memory requirement, and further largely improves the efficiency of the SOFL classifier.

Then, $\boldsymbol{x}_{i,K_i}$ is checked by Condition 3 (equation (3.10)) to evaluate its potential to be a new prototype [29], [55], [160]. If $\boldsymbol{x}_{i,K_i}$ meets Condition 3, a new prototype is added to the fuzzy rule of the i^{th} class in the same form as equation (4.12), and the meta-parameters of the SOFL classifier are updated as $P_i \leftarrow P_i + 1$; $\Xi_{i,P_i} \leftarrow \{\boldsymbol{x}_{i,K_i}\}$; $\boldsymbol{p}_{i,P_i} \leftarrow \boldsymbol{x}_{i,K_i}$; $S_{i,P_i} \leftarrow 1$.

If Condition 3 (equation (3.10)) is unsatisfied, it is necessary to check whether $\boldsymbol{x}_{i,K_i}$ is very close to an existing prototype by using Condition 26.

$$Condition\ 26\text{:}\quad \begin{array}{c} IF\ \left(\min_{\boldsymbol{p}\in\{\boldsymbol{p}\}_i}\left(d^2(\boldsymbol{x}_{i,K_i}, \boldsymbol{p})\right) > \mathcal{L}_i^G\right) \\ THEN\ \left(\boldsymbol{x}_{i,K_i} \in \{\boldsymbol{p}\}_i\right) \end{array}. \tag{4.24}$$

If Conditions 3 and 26 are both unsatisfied, $\boldsymbol{x}_{i,K_i}$ is assigned to the nearest prototype $\boldsymbol{p}_{i,n} = \text{argmin}_{\boldsymbol{p}\in\{\boldsymbol{p}\}_i}\left(d\left(\boldsymbol{x}_{i,K_i},\boldsymbol{p}\right)\right)$ and the meta-parameters of the corresponding data cloud are updated as $\Xi_{i,n} \leftarrow \{\Xi_{i,n},\boldsymbol{x}_{i,K_i}\}$; $\boldsymbol{p}_{i,n} \leftarrow \frac{S_{i,n}}{S_{i,n}+1}\boldsymbol{p}_{i,n} + \frac{1}{S_{i,ne}+1}\boldsymbol{x}_{i,K_i}$; $S_{i,n} \leftarrow S_{i,n} + 1$ [55].

After the meta-parameters of the classifier are updated, the AnYa type fuzzy rule (equation (4.12)), will be updated accordingly and the SOFL classifier is ready for processing the next data sample or conducting classification.

4.3.2.2. Algorithm Summary

The main procedure of the online training process of the SOFL classifier is summarised in the form of a flowchart in Figure 24.

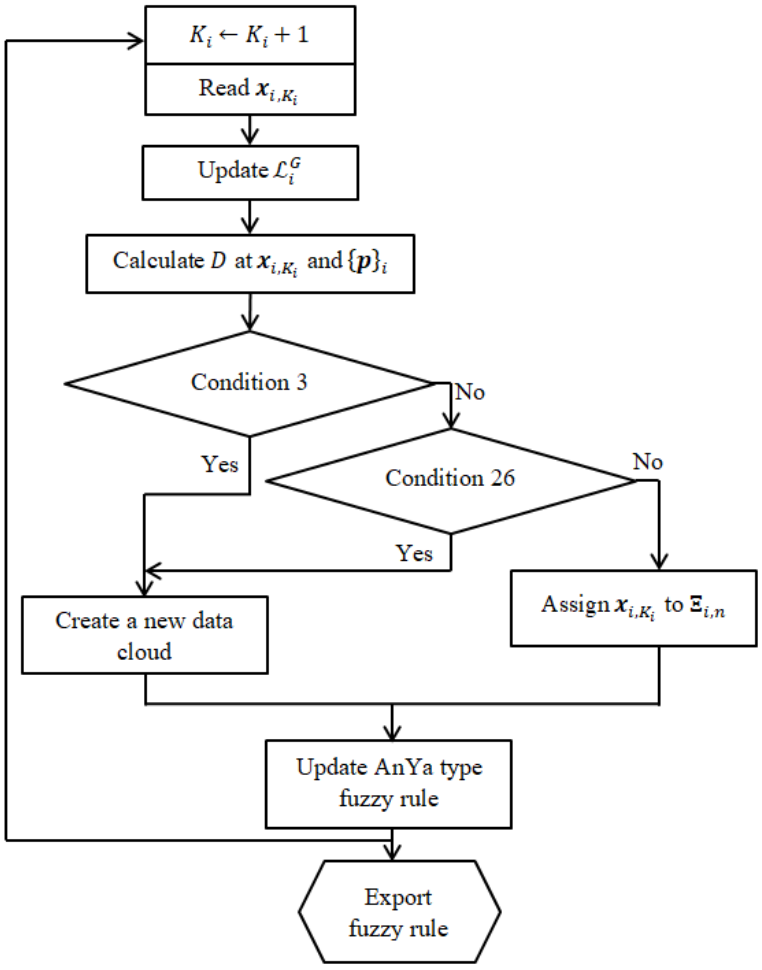

$K_i \leftarrow K_i + 1$
Read $\boldsymbol{x}_{i,K_i}$
Update $\mathcal{L}_i^G$
Calculate D at $\boldsymbol{x}_{i,K_i}$ and $\{\boldsymbol{p}\}_i$
Condition 3
No
Yes
Condition 26
No
Yes
Create a new data cloud
Assign $\boldsymbol{x}_{i,K_i}$ to $\Xi_{i,n}$
Update AnYa type fuzzy rule
Export fuzzy rule

Figure 24. Main procedure of the online training process of SOFL classifier.

4.3.3. Validation Process

Due to the fact that both the SOFL classifier presented in this section and the ALMMO-0 presented in section 4.2 use the same type of fuzzy rules, the procedure of the SOFL classifier for decision-making for the unlabelled samples is the same as the ALMMO-0, which is described in subsection 4.2.3.

4.4. Autonomous Anomaly Detection

In this section, a new fully autonomous anomaly detection (AAD) method is presented [37]. In this approach, the nonparametric EDA estimators, cumulative proximity, unimodal density and multimodal density [25]–[27] are employed to identify the potential anomalies from the empirically observed data at the first stage of the process. Then, these potential anomalies are used for forming shape-free data clouds using the autonomous data partitioning approach as described in section 3.2.1. Finally, the local anomalies are identified in regards to the data clouds.

The AAD approach can autonomously and objectively detect both individual and collective anomalies (remote, small clouds) and also global anomalies as well as anomalies that are centrally located. Its procedure consists of three stages as follows.

4.4.1. Identifying Potential Anomalies

In the first stage, the global mean and average scalar product, $\boldsymbol{\mu}_K$ and X_K of $\{\boldsymbol{x}\}_K$ are calculated. Then, the multimodal densities D^G at $\{\boldsymbol{u}\}_N$ are obtained using equation (2.19). By extending D^G to $\{\boldsymbol{x}\}_K$, the multimodal densities at each data sample $\boldsymbol{x}$ ($\boldsymbol{x} \in \{\boldsymbol{x}\}_K$) are obtained and denoted as $\{D_K^G(\boldsymbol{x})\}$.

Chebyshev inequality (equation (2.39)) [2]–[4] describes the probability data samples to be more than $n\sigma$ distance away from the mean value $\boldsymbol{\mu}$. As a corollary, if $n = 3$, the maximum probability of $\boldsymbol{x}$ to be at least 3σ away from $\boldsymbol{\mu}$ is no more than $1/9$. In other words, on average, out of 9 data samples, one may be anomalous, but no more than 1 (at most 1). Therefore, in the AAD approach, it is assumed that $1/n^2$ of the data samples are potentially abnormal, however, it does not mean that they have to be real anomalies.

By ranking $\{D_K^G(\boldsymbol{x})\}$ in an ascending order, the first half of $1/n^2$ of the data samples with the smallest D^G being the first half of the potential anomaly collection, denoted as $\{\boldsymbol{x}\}_{PA,1}$. Here, n is a small integer corresponding to the "n" in the Chebyshev inequality. In this thesis, $n = 3$ is adopted because the "3σ" rule has been widely adapted in various anomaly detection applications [49], [203], [204]. It has to be stressed that in traditional approaches, $n = 3$ does directly influence detecting each anomaly. In the AAD approach, this is simply the first stage of sub-selection of potential anomalies (an upper limit according to equation (2.39)).

As the multimodal density is less sensitive to the degree of sparsity of local data distribution, an additional criterion is necessary for detecting the isolated data samples. We consider the weighted local unimodal density as the second criterion for identifying potential anomalies.

The local unimodal density of each unique data sample is calculated using equation (3.1) and (3.2), denoted by $D_L(\boldsymbol{u}_i)$ $(\boldsymbol{u}_i \in \{\boldsymbol{u}\}_N)$. However, in the AAD approach, instead of calculating D_L locally for all the data samples located in the hypersphere with $\boldsymbol{u}_i$ as its centre and $\bar{d}$ as its radius, D_L is calculated within the hypersphere with $\boldsymbol{u}_i$ as its centre and $\bar{d}/2$ as its radius ($D_L(\boldsymbol{u}_i) = \frac{\sum_{d(\boldsymbol{x},\boldsymbol{u}_i)\leq(\bar{d}/2)} q_L(\boldsymbol{x})}{2N_i q_L(\boldsymbol{u}_i)}$), which allows the AAD approach more effectively in detecting data samples away from the majority.

By taking both, the sparsity of unique data samples around $\boldsymbol{u}_i$ and the data distribution of the local area into consideration, the local unimodal density at $\boldsymbol{u}_i$ is weighed by the amount of its unique neighbours as:

$$D_L^W(\boldsymbol{u}_i) = \frac{N_i}{N} \cdot D_L(\boldsymbol{u}_i), \tag{4.25}$$

where the coefficient N_i/N is for ensuring the value of $D_L^W(\boldsymbol{u}_i)$ to be linearly and inversely correlated to the degree of sparsity of the data distribution, N_i is the number of unique data samples around $\boldsymbol{u}_i$ within the range of $\bar{d}/2$; By expanding the weighted local unimodal densities, D_L^W, at $\{\boldsymbol{u}\}_N$ to the original dataset $\{\boldsymbol{x}\}_K$ accordingly, the set $\{D_L^W(\boldsymbol{x})\}$ is obtained. After re-ranking the $\{D_L^W(\boldsymbol{x})\}$ in the ascending order, the first half of $1/n^2$ of the data samples with smallest D_L^W are selected as the second half of the potential anomaly collection, denoted as $\{\boldsymbol{x}\}_{PA,2}$.

Finally, by combining $\{\boldsymbol{x}\}_{PA,1}$ and $\{\boldsymbol{x}\}_{PA,2}$ (together $1/n^2$ or less of the data), we obtain the whole set of potential anomalies, $\{\boldsymbol{x}\}_{PA}$, which forms the upper limit of possible anomalies according to Chebyshev inequality (equation (2.39)).

4.4.2. Forming Data Clouds with Anomalies

In this stage, all the identified potential anomalies are checked to see whether they are able to form data clouds using the ADP algorithm as described in section 3.3.1. After the data clouds are formed from $\{\boldsymbol{x}\}_{PA}$ based on the ADP algorithm, denoted by $\{\Xi\}_{PA}$, the AAD algorithm enters the last stage.

4.4.3. Identifying Local Anomalies from Identified Data Clouds

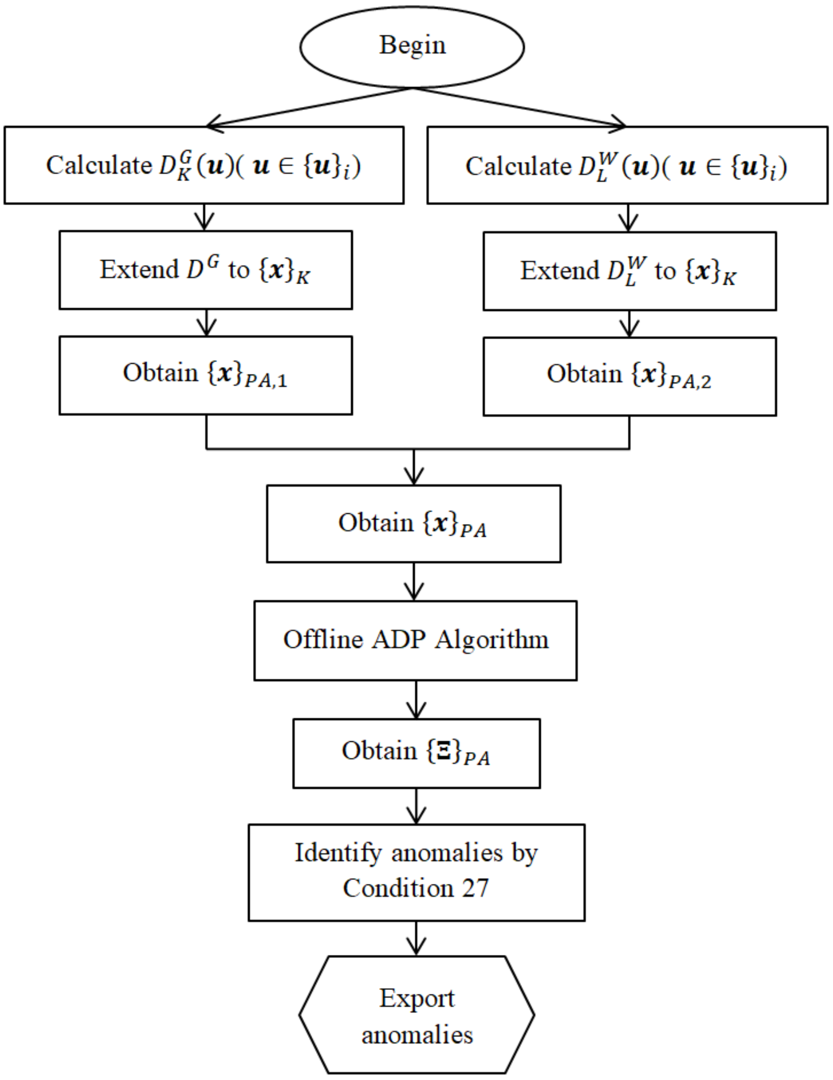
In the final stage, all the potential anomalies are checked to see if they are isolated or form minor data cloud(s) between themselves. All the data clouds formed from $\{\boldsymbol{x}\}_{PA}$ are checked using Condition 27:

$$Condition\ 27{:}\quad \begin{array}{c} IF\ \left(S_i < S_{average}\right) \\ THEN\ (\Xi_i\ is\ formed\ by\ anomalies) \end{array}, \tag{4.26}$$

where $\Xi_i \in \{\Xi\}_{PA}$ and S_i is the support of Ξ_i.

After all the data clouds meeting Condition 27 are identified, anomalies are identified and declared/confirmed.

The main procedure of the AAD algorithm is summarised in the form of a flowchart in Figure 25.



Begin
Calculate $D_K^G(\boldsymbol{u})(\ \boldsymbol{u} \in \{\boldsymbol{u}\}_i)$
Calculate $D_L^W(\boldsymbol{u})(\ \boldsymbol{u} \in \{\boldsymbol{u}\}_i)$
Extend D^G to $\{\boldsymbol{x}\}_K$
Extend D_L^W to $\{\boldsymbol{x}\}_K$
Obtain $\{\boldsymbol{x}\}_{PA,1}$
Obtain $\{\boldsymbol{x}\}_{PA,2}$
Obtain $\{\boldsymbol{x}\}_{PA}$
Offline ADP Algorithm
Obtain $\{\Xi\}_{PA}$
Identify anomalies by Condition 27
Export anomalies

Figure 25. Main procedure of AAD algorithm.

4.5. Conclusion

In this chapter, four different supervised machine learning algorithms for regression, classification and anomaly detection problems are presented. Compared with the traditional approaches, the algorithms presented in this chapter have the following distinctive properties:

1) They are nonparametric and free from unrealistic prior assumptions;

2) They are autonomous, self-organising;

3) They are based on the ensemble properties and mutual distribution of empirically observed data, and thus, are able to fully objective results.

5. Transparent Deep Learning Systems

Deep learning is closely associated with the artificial neural networks (ANNs) [66]. Nowadays, deep learning has gained a lot of popularity in both the academic circles and the general public due to the very quick advance in the computational resources (both hardware and software) [19], [66]. A number of publications have demonstrated that deep convolutional neural networks (DCNNs) can produce highly accurate results in various image processing problems including, but not limited to, handwritten digits recognition [22], [24], [72]–[74], object recognition [21], [23], [75], [76], human action recognition [77], [78], remote sensing image classification [79], etc. Some publications suggest that the DCNNs can match the human performance on the handwritten digits recognition problems [22], [24], [73], [74]. Indeed, DCNN is a powerful technique that provides high classification rates. Nonetheless, the celebrated success comes at a price, the DCNNs still have a number of deficiencies and shortcomings, i.e. the computational burden of training using huge amount of data, lack of transparency and interpretation, ad hoc decisions about the internal structure, no proven convergence for the adaptive versions that rely on reinforcement learning, limited parallelisation and offline training, etc. These deficiencies largely hinder the wider application of the DCNNs for real problems.

In this chapter, the newly introduced transparent, nonparametric, feedforward and human interpretable deep learning networks developed on the basis of the recently introduced AnYa type FRB systems and the EDA framework are presented. The fast feedforward nonparametric deep learning (FFNDL) network with automatic feature extraction is presented in section 5.1. The deep rule-based system with the prototype-based nature and transparent structure is presented in section 5.2. The semi-supervised active learning mechanism of the deep rule-based system is given in section 5.3. Several successful examples of deep rule-based ensemble classifiers are presented in section 5.4. This chapter is concluded by section 5.5.

5.1. Fast Feedforward Nonparametric Deep Learning Network

In this section, the fast feedforward nonparametric deep learning (FFNDL) network with automatic feature extraction is presented [38]. The FFNDL network is based on human-understandable local aggregations extracted directly from the images. There is no need for any feature selection and parameter tuning. It involves nonlinear transformation, segmentation operations to select the most distinctive features from the training images and builds RBF neurons based on them to perform classification with no weights to train. The design of the FFNDL network is very efficient (computation and time wise) and produces highly accurate classification results (see subsection 6.3.1 and [38]). Moreover, the training process is parallelisable, and the time consumption can be further reduced with more processors involved.

5.1.1. Architecture of FFNDL Network for Feature Extraction

The architecture of the FFNDL network up to the final class prediction stage is depicted in Figure 26. As it is shown in Figure 26, the FFNDL network has six layers plus the prediction layer. The first layer is for non-overlapping mean pooling with size 2×2 (obtained empirically). The second layer is for extracting local aggregations as features from the pooled images. The third layer is nonlinear mapping layer. The forth layer is the segmentation layer. The fifth layer is for filtering out the overlapping/similar features extracted from images of different classes. The sixth layer includes the RBF neurons built based on the extracted local aggregations.

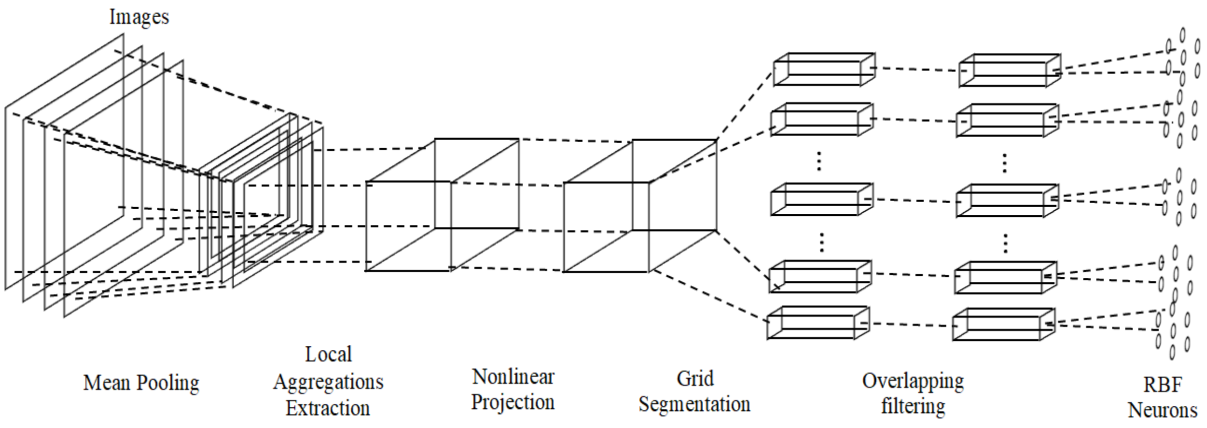


Figure 26. Architecture of the FFNDL network for feature extraction.

In the rest of this section, the novel characteristics of the FFNDL network will be described. For simplicity, only grayscale images with pixel values scaled into $[0,1]$ are considered. The size of the original images is denoted as $2d \times 2d$, and, thus, after the mean pooling, the size of images becomes $d \times d$.

5.1.1.1. Local aggregations extraction layer

In this layer, the local aggregations within images are extracted. These are based on the gradients between the grey level values of the surrounding/neighbouring pixels to a given pixel. In the FFNDL network, the local aggregations for the pixel $z_{i,j}$ (i, j are the coordinates indicating the position of this pixel within a particular image) is achieved by using a $n \times n$ (n

is a small odd number) sliding window with a stride of one pixel, and the pixel $z_{i,j}$ is in the centre of the sliding window. The local aggregation around $z_{i,j}$ is expressed as:

$$\begin{aligned}\mathbf{A}_{i,j} &= \begin{bmatrix} z_{i,j} - z_{i-\frac{n-1}{2},j-\frac{n-1}{2}} & \cdots & z_{i,j} - z_{i-\frac{n-1}{2},j+\frac{n-1}{2}} \\ \vdots & \ddots & \vdots \\ z_{i,j} - z_{i+\frac{n-1}{2},j-\frac{n-1}{2}} & \cdots & z_{i,j} - z_{i+\frac{n-1}{2},j+\frac{n-1}{2}} \end{bmatrix}, \\ &= \left[\boldsymbol{\alpha}_{i,j-\frac{n-1}{2}}, \dots, \boldsymbol{\alpha}_{i,j-1}, \boldsymbol{\alpha}_{i,j-1}, \boldsymbol{\alpha}_{i,j+1}, \dots, \boldsymbol{\alpha}_{i,j+\frac{n-1}{2}}\right]\end{aligned} \tag{5.1}$$

where $\boldsymbol{\alpha}_{i,j-l} = \left[z_{i,j} - z_{i-\frac{n-1}{2},j-l}, \dots, z_{i,j} - z_{i+\frac{n-1}{2},j-l}\right]^T$.

By using the gradients of the grey level values as local aggregations, the local features, i.e. edges, shapes, are preserved, while the influence of illumination is reduced. In order to get the most effective local aggregations, only the valid features $\mathbf{A}_{i,j}$that have more than half of its elements being non-zero are considered. The $\mathbf{A}_{i,j}$ that fail to meet this requirement are being discarded.

After the $n \times n$ local aggregations are extracted, the matrix is converted into a long vector by concatenating different rows from the local aggregation matrix. Because, the centre of each aggregation is always equal to zero, the centre in the vector can be omitted, and, thus, a $(n^2 - 1) \times 1$ local aggregation vector is obtain:

$$\mathbf{B}_{i,j} = \left[\boldsymbol{\alpha}^T_{i,j-\frac{n-1}{2}}, \dots, \boldsymbol{\alpha}^T_{i,j-1}, \overline{\boldsymbol{\alpha}}^T_{i,j}, \boldsymbol{\alpha}^T_{i,j+1}, \dots, \boldsymbol{\alpha}^T_{i,j+\frac{n-1}{2}}\right]^T, \tag{5.2}$$

where $\overline{\boldsymbol{\alpha}}_{i,j} = \left[z_{i,j} - z_{i-\frac{n-1}{2},j}, \dots, z_{i,j} - z_{i-1,j}, z_{i,j} - z_{i+1,j}, \dots, z_{i,j} - z_{i+\frac{n-1}{2},j}\right]^T$.

5.1.1.2. Nonlinear Projection Layer

After the extraction of the local aggregations, their values are limited to the range $[-1,1]$ because the pixel grey level values are normalised into the range $[0,1]$. This small value range makes it hard to linearly separate the local aggregations from different classes. Therefore, the following nonlinear one-to-one mapping function is employed to amplify the differences between various local aggregations and make them separable:

$$\kappa(x) = \mathrm{sgn}(1-x)\left[\exp\left(\left(1 + \mathrm{sgn}(1-x)(1-x)\right)^2\right) - \exp(1)\right], \tag{5.3}$$

where $\mathrm{sgn}(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0. \\ -1 & x < 0 \end{cases}$

By using the nonlinear mapping, the FFNDL network amplifies the differences between local aggregations, and, thus, improves the distinctiveness of the extracted local aggregations. After the nonlinear mapping, each local aggregation vector $\mathbf{B}_{i,j}$ is expressed as:

$$\boldsymbol{\ell}_{i,j} = \kappa\left(\mathbf{B}_{i,j}\right) = \left[\kappa\left(\boldsymbol{\alpha}_{i,j-\frac{n-1}{2}}^{T}\right), \ldots, \kappa\left(\boldsymbol{\alpha}_{i,j-1}^{T}\right), \kappa\left(\overline{\boldsymbol{\alpha}}_{i,j}^{T}\right), \kappa\left(\boldsymbol{\alpha}_{i,j+1}^{T}\right), \ldots, \kappa\left(\boldsymbol{\alpha}_{i,j+\frac{n-1}{2}}^{T}\right)\right]^{T}. \tag{5.4}$$

5.1.1.3. Grid Segmentation Layer

In the FFNDL network, the grid segmentation is achieved using a sliding window with size of $k \times k$ pixels and a stride of w pixel. The grid segmentation layer further divides the image space into $\left(\frac{d-k-n+1}{w} + 1\right)^2$ small blocks with the size of $k \times k \times (n^2 - 1)$ overlapping with each other. This operation is equal to the over-sampling. By assigning the local aggregations to the blocks they belong to, the original positions of the local aggregations are replaced by the positions of their corresponding blocks, which allow the local aggregations a small space for shifting. In addition, as the blocks are independent from each other, parallel computation can be achieved to process each block separately and, thus, improve the computation efficiency of the proposed network.

After the grid segmentation, each block can be viewed as a set of local aggregations from different images of different classes:

$$\mathbf{Q}_i = \left\{\{\boldsymbol{\ell}\}_{i,1}, \{\boldsymbol{\ell}\}_{i,2}, \ldots, \{\boldsymbol{\ell}\}_{i,C}\right\}, \tag{5.5}$$

where i is the index of the blocks, $i = 1,2, \ldots, \left(\frac{d-k-n+1}{w} + 1\right)^2$; $\{\boldsymbol{\ell}\}_{i,c}$ denotes the local aggregations extracted in the range covered by the i^{th} block from the images from the c^{th} class.

5.1.1.4. Overlapping Filtering Layer

The FFNDL network relies on the extracted local features to make the classification decision. However, in many cases, the same local features can appear in different classes. It is, therefore, important to select the most distinctive features only.

Considering the dimensionality of the extracted local aggregations in the FFNDL network, Euclidean distance is not the best choice due to its inherited deficiencies for high dimensionality problems [205], [206]. Instead, cosine dissimilarity of the local aggregations from two different classes within the same block is calculated:

$$d(\boldsymbol{\ell}_j, \boldsymbol{\ell}_t) = \sqrt{2 - 2cos\left(\theta_{\boldsymbol{\ell}_j,\boldsymbol{\ell}_t}\right)} = \left\| \frac{\boldsymbol{\ell}_j}{\|\boldsymbol{\ell}_j\|} - \frac{\boldsymbol{\ell}_t}{\|\boldsymbol{\ell}_t\|} \right\|, \tag{5.6}$$

where $\boldsymbol{\ell}_j \in \{\boldsymbol{\ell}\}_{i,j}$, $\boldsymbol{\ell}_t \in \{\boldsymbol{\ell}\}_{i,t}$; $\{\boldsymbol{\ell}\}_{i,j}, \{\boldsymbol{\ell}\}_{i,t} \subseteq \mathbf{Q}_i$ and $j \neq t$; $i = 1,2, \dots, \left(\frac{d-k-n+1}{w} + 1\right)^2$; $\theta_{\boldsymbol{\ell}_j,\boldsymbol{\ell}_t}$ is the angle between $\boldsymbol{\ell}_j$ and $\boldsymbol{\ell}_t$.

Then, Condition 28 is checked:

$$Condition\ 28: \quad \begin{array}{l} IF \left(\frac{d^2(\boldsymbol{\ell}_j,\boldsymbol{\ell}_t)}{2} \geq 1 - cos\left(\frac{\pi}{6}\right)\right) \\ THEN \left(\mathbf{Q}_{i,R} \leftarrow \{\mathbf{Q}_{i,R}, \boldsymbol{\ell}_j, \boldsymbol{\ell}_t\}\right) \end{array}, \tag{5.7}$$

where $\mathbf{Q}_{i,R}$ is the collection of similar local aggregations.

If Condition 28 is met, it means that, in the Euclidean data space, the angle between $\boldsymbol{\ell}_j$ and $\boldsymbol{\ell}_t$ is smaller than 30^o, which means that the two local aggregations are quite similar and keeping them in $\mathbf{Q}_i$ will lead to misleading results. By finding out all the $\boldsymbol{\ell}$ satisfying Condition 28, the collection of similar local aggregations, $\mathbf{Q}_{i,R}$, is obtained, and by excluding $\mathbf{Q}_{i,R}$ from $\mathbf{Q}_i$, the distinctive local aggregations are all selected.

5.1.1.5. Cosine Dissimilarity based RBF Neurons Layer

After the distinctive local aggregations are all selected, they are used to build the final layer of the proposed network. The final layer consists of a number of RBF neurons; each neuron is directly related to a distinctive local aggregation. The RBF neurons of each block are viewed as a group. It is important to stress that there is no dependence of different groups of RBF neurons between each other.

By including equation (5.6), the RBF function based on a particular distinctive local aggregation $\boldsymbol{\ell}$ is, finally, expressed as equation (5.8) [38]:

$$f(\boldsymbol{x}) = \exp\left(-\frac{1}{8}d^4(\boldsymbol{x}, \boldsymbol{\ell})\right) = \exp\left(-\frac{1}{8}\left\|\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} - \frac{\boldsymbol{\ell}}{\|\boldsymbol{\ell}\|}\right\|^4\right), \tag{5.8}$$

where $\boldsymbol{x}$ is the input vector and $\boldsymbol{\ell}$ is the distinctive local aggregation corresponding to the neuron.

After the RBF neurons are built based on the extracted distinctive local aggregations, the learning stage of the proposed network is finished, and it can be used for evaluation. As one can see from the above description, there is no parameter optimisation or iteration in the training of the FFNDL network. It is based on the local features extracted automatically from the training images to build RBF neurons and further classify new images. As a result, the FFNDL network is able to learn from a large number of images in a high speed.

5.1.2. Architecture of FFNDL Network for Classification

Once the FFNDL network has extracted the local aggregations from the training images in the learning stage, the network is prepared for classifying new images. The architecture of the FFNDL network for classification is depicted in Figure 27. In this section, only the components that have not been descriped prevously will be described in detail.

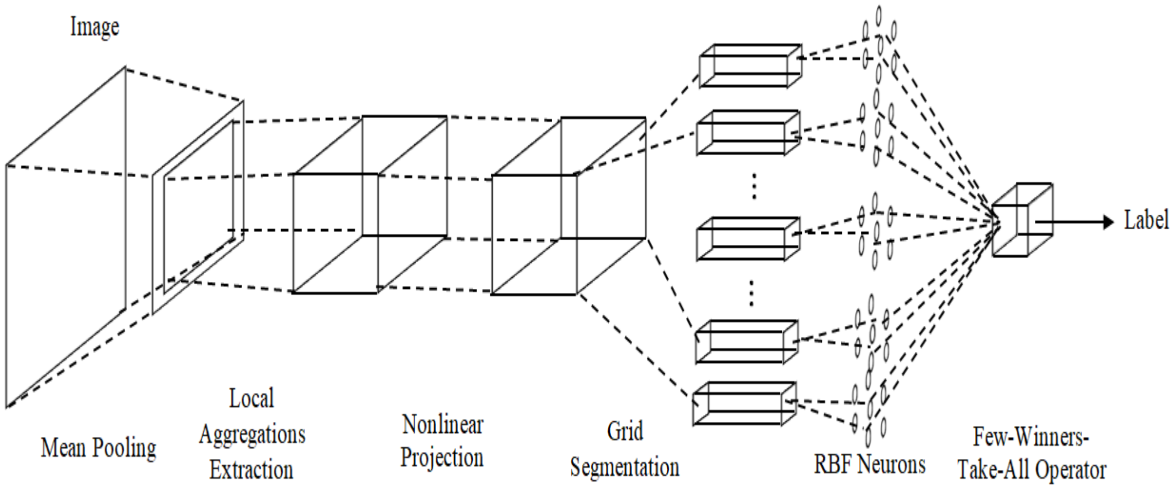


Figure 27. Architecture of the FFNDL network for classification.

5.1.2.1. RBF Neurons Layer

For each testing image, denoted by **I**, the process will sequentially go through the mean pooling layer, local aggregation, nonlinear mapping, and grid segmentation layers. After the segmentation operation, the image will be divided into blocks in the same way as described in section 5.1.1.3 and the local aggregations within each block will serve as the inputs of the RBF neuron group connected to that block.

When a local aggregation within the i^{th} block, denoted as $\boldsymbol{x}$, is sent to the connected neuron group, the likelihoods of x belonging to each class are calculated according to the following rule:

$$\omega_c(\boldsymbol{x}) = \text{argmax}_{j=1,2,\dots,P_C}\left(f_{c,j}(\boldsymbol{x})\right) = \text{argmax}_{j=1,2,\dots,P_C}\left(\exp\left(-\frac{1}{8}\left\|\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} - \frac{\boldsymbol{\ell}_{c,j}}{\|\boldsymbol{\ell}_{c,j}\|}\right\|^4\right)\right), \quad (5.9)$$

where $P_{c,i}$ is the number of RBF neurons belonging to the c^{th} class in the group; $\boldsymbol{\ell}_{c,j}$ is the j^{th} distinctive local aggregation of the c^{th} class within the group; $c = 1,2,\dots,C$.

Therefore, after all the local aggregations of the testing image, denoted as $\{\boldsymbol{x}\}$, have been segmented according to their positions in the image and gone through the corresponding RBF neuron groups, the outputs of the local classifiers in regards to different classes are obtained and denoted as: $\{\{\omega(\boldsymbol{x})\}_1, \{\omega(\boldsymbol{x})\}_2, \dots, \{\omega(\boldsymbol{x})\}_C\}$. Then, the outputs will be sent to the "few winners take all" module to decide the label of the testing image.

5.1.2.2. "Few Winners Take All" Operator

Due to the fact that the FFNDL network is operating based on the local features, one cannot expect that a particular testing image has all the local features at the same time. However, for any two images within the same class, there is a very large chance that they can hold some similar local features. Therefore, the "few winners take all" strategy is employed to decide the label. Considering the fact that the numbers of identified local features from the training images of different classes are different, only the average value of the top 15% outputs of the local classifiers of each collection is taken into account:

$$\lambda_c = \frac{1}{\lceil 0.15 \cdot L_c \rceil} \sum_{i=1}^{\lceil 0.15 \cdot L_c \rceil} \omega_c(\widehat{\boldsymbol{x}}), \tag{5.10}$$

where L_c is the number of local classifiers in the collection $\{\omega(\boldsymbol{x})\}_c$; $\{\omega(\widehat{\boldsymbol{x}})\}_c$ is the ranked$\{\omega(\boldsymbol{x})\}_c$ in a descending order.

Based on λ_c $(c = 1,2, \dots, C)$, the label of the image is decided as:

$$y(\mathbf{I}) = \text{argmax}_{j=1,2,\dots,P_c}(\lambda_c). \tag{5.11}$$

5.2. Deep Rule-Based Classifier

Traditional fuzzy rule-based classifiers were successfully used for classification [160], [201] offering transparent, interpretable structure, but could not reach the levels of performance achieved by deep learning classifiers. Their design also requires handcrafting membership functions, assumptions to be made and parameters to be selected.

In this section, a new type of deep rule-based (DRB) system with a multilayer architecture for image classification is presented. Combining the computer vision techniques, the DRB approach employs a massively parallel set of 0-order fuzzy rules [35], [160] as the learning engine, which self-organises a transparent and human understandable IF…THEN… fuzzy rule-based (FRB) system structure in a highly efficient way and offers extremely high classification accuracy at the same time [39], [43]. Its training process is fully online, non-iterative, non-parametric and can start "from scratch", more importantly, it can start classification from the very first image of each class in the same way as humans do, which makes the proposed classifier suitable for real-time applications (see subsection 6.3.2 and also see [39]).

The DRB classifier is also further extended with a semi-supervised learning strategy (presented in the next section) in a self-organising way and, thus, enhances its ability of handling unlabelled images. Thanks to the prototype-based nature of the DRB classifier, the semi-supervised learning process is fully transparent and human-interpretable. It not only can perform classification on out-of-sample images, but also supports recursive online training on a sample-by-sample basis or a batch-by-batch basis. Moreover, the semi-supervised DRB classifier is able to learn new classes actively without human experts' involvement.

5.2.1. General Architecture

The general architecture of the DRB classifier is depicted in Figure 28. One can see from the figure that the proposed DRB approach consists of the following layers:

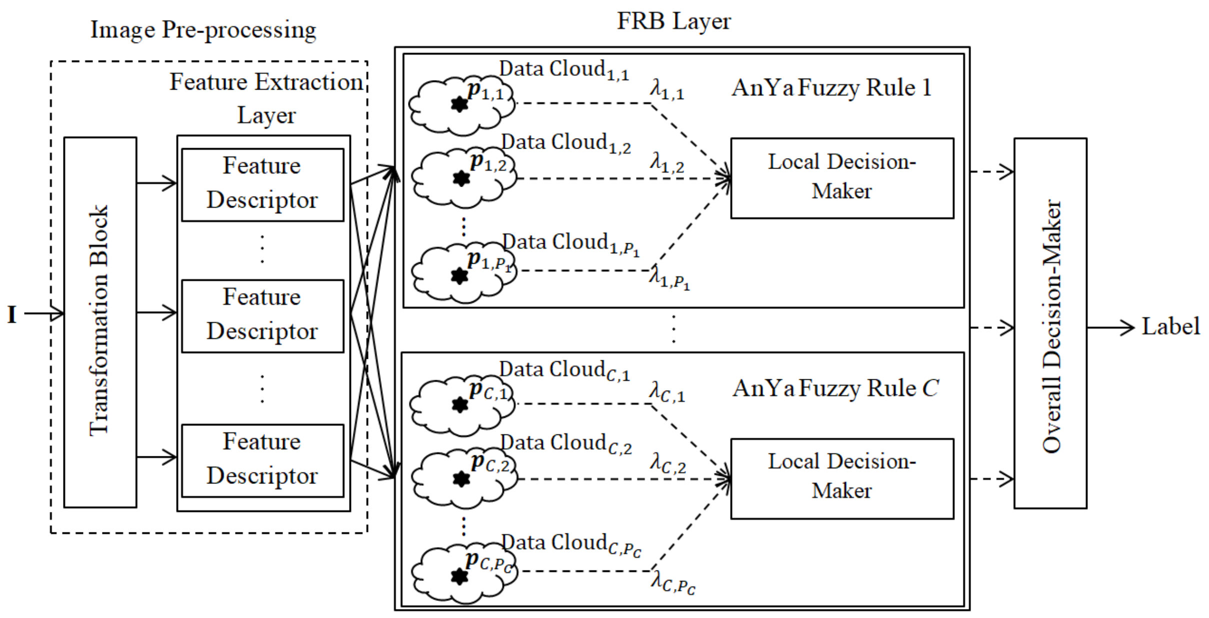


Figure 28. General architecture of DRB classifier.

① Transformation block;

② Feature extraction layer;

③ Massively parallel ensemble of IF…THEN… rules;

④ Decision-maker.

The transformation block of the proposed DRB classifier involves only the most fundamental image transformation techniques, namely: *i)* normalization, *ii)* scaling, *iii)* rotation and *iv)* image segmentation and, thus, it is, in fact, composed of a number of sublayers serving for various purposes. It is well known that normalization is the process of linear transformation of the original value range of $[0, 255]$ into the range $[0, 1]$ [207]. Scaling is the process of resampling and resizing of a digital image [208], [209]. Rotation is a technique usually applied to images rotated at a certain angle around a centre point [207]. Scaling and rotation techniques are two types of affine distortion, and they can significantly improve the generalization and decrease the overfitting [22], [24], [73]. Segmentation is the process of partitioning an image into smaller pieces to extract local information or discard the less informative part of the image [21]. The main purposes of the transformation block within the DRB classifier are *i)* improving the generalization ability of the classifier and *ii)* increasing the efficiency of the feature descriptors in harvesting information from the image. The sub-structures of the transformation block and the usages of the image transformation techniques are subjected to different problems and applications. A more detailed description of the pre-processing techniques we used can also be found in section 5.2.2 [41].

For feature extraction, namely layer ②, the proposed DRB classifier may employ various different kinds of feature descriptors that are used in the field of computer vision. Different feature descriptors have different advantages and deficiencies [210]. The details of feature extraction will be discussed in section 5.2.3.

The layer ③ of the proposed new DRB classifier is the massively parallel ensemble of IF…THEN… rules which will be described in more detail in section 5.2.4. This is the "engine" of the new DRB classifier and is based on autonomously self-developing fuzzy rule-based models of AnYa type [60] with singletons in the consequent part.

The structure of a particular AnYa type fuzzy rule can be found in Figure 28 as well. As one can see, each fuzzy rule used in the DRB classifier is itself a combination of a large number of data clouds associated with the fuzzy prototypes identified through the one-pass type training, which means that the IF…THEN… rule can be massively parallelised if consider each data cloud/prototype as a separate fuzzy rule. The local decision-maker is a "winner takes all" operator. Therefore, the fuzzy prototypes can be viewed as being connected by logical "OR" operators.

The final layer is the decision-maker that decides the winning class label based on the partial suggestion of the massively parallel IF…THEN… rule per class. This layer is only used during the validation stage and it applies the "winner takes all" principle as well. As a result, one can see that the proposed DRB classifier actually uses a two-layer decision-making structure. The validation process will be described in section 5.2.5.

5.2.2. Image Transformation Techniques

In this section, the image transformation techniques including normalisation, affine distortion and elastic distortion are presented.

The normalisation and affine distortion are the pre-processing transformations generally applicable to various image processing problems, i.e. remote sensing [211], object recognition [212], etc. In contrast, the elastic distortion is mostly only applicable to the handwritten digits and/or letters recognition problems, i.e. Modified National Institute of Standards and Technology (MNIST) database [213]. In the DRB classifier, only the normalisation and affine distortions techniques are employed.

5.2.2.1. Image Normalisation

Normalisation is a common process in image processing that changes the value range of the pixels within the image. The goal is to transform the image such that the values of pixels are mapped into a more familiar or normal range. This operation can be used to readjust the degree of illumination of the images as well.

In the DRB classifier, the most commonly used linear normalisation is used to fit the original pixel value range of $[0, 255]$ into the range of $[0,1]$.

5.2.2.2. Affine Distortions

Affine distortion can be done by applying affine displacement fields to images, computing for every pixel a new target location with respect to the original one. Affine distortions including rotations and scaling are very effective to improve the generalization and decrease the overfitting [22], [24], [73].

A. Image Scaling

Image scaling refers to the resampling and resizing of a digital image [208], [209]. There are two types of image scaling: 1) image contraction and 2) image expansion. Image scaling is achieved by using an interpolation function. There is a number of different interpolation methods for image resizing reported in the literature [208], [209], [214], [215], e.g. nearest neighbour interpolation, bilinear interpolation and bicubic interpolation methods. In this thesis, the most commonly used bicubic interpolation method [214], [215] is used, which considers the nearest 16 pixels (4×4) in the neighbourhood and calculates the output pixel value as their weighted average. Since the 16 neighbouring pixels are at various distances from the output pixel, closer pixels are given a higher weighting in the calculation.

B. Image Rotation

Image rotation is another common image pre-processing technique performed by rotating an image at certain angle around the centre point [207]. Usually, the nearest neighbour interpolation is used after the rotation and the values of pixels that are outside the rotated images are set to 0 (black).

C. Image Segmentation

Segmentation is the process of partitioning an image into smaller pieces to extract local information or discard the less informative part of the image [21]. The main purposes of the pre-processing layer within the DRB classifier are *i)* improving the generalization ability of the classifier and *ii)* increasing the efficiency of the feature descriptors in harvesting information from the image.

5.2.2.3. Elastic Distortion

Many approaches on the well-known MNIST database [213] have been proposed and reported with the best result published to the moment provides a recognition accuracy of 99.77% [22]. The elastic distortion is the key to the success.

Elastic distortion is a more advanced and effective technique to expand the dataset and improve the generalization [22], [24], [73]. The elastic distortion is done by, firstly, generating random displacement fields and then convolving the displacement fields with a Gaussian function of standard deviation σ (in pixels) [73]. This type of image deformation has been widely used in the state-of-the-art deep convolutional neural networks for handwriting recognition [22], [24] and largely improved the recognition accuracy.

However, elastic distortion is not only opaque (not clearly reported) but also is random in nature [4]. This, combined with the other random elements of the architecture, leads to the results being different each time training is performed on the same data.

This kind of distortion exhibits a significant randomness that puts in question the achieved results' repeatability and requires a cross-validation which further obstructs the online applications and the reliability of the results. In addition, it adds user-specific parameters that can be chosen differently. For a particular image, each time the elastic distortion is performed, an entirely new image is being generated.

In addition, there is no evidence or experiment supporting that the elastic distortion can be applied to other types of image recognition problems. In fact, the elastic distortion destroys the images.

5.2.3. Image Feature Extraction

Feature extraction is very important to solve computer vision problems such as object recognition, content-based image classification and image retrieval [216]. The extracted features have to be informative, non-redundant, and, most importantly, to be able to facilitate the subsequent learning and generalization.

The feature extraction, in fact, can be viewed as a projection from the original images into a feature space that makes the images from different classes separable. Current feature descriptors are divided into "low-level", "medium-level" and "high-level" three categories based on their descriptive abilities [210]. Different feature descriptors have different advantages. In general, the low-level feature descriptors work very well in the problems where low-level visual features, e.g., spectral, texture, and structure, play the dominant role. In contrast, high-level feature descriptors work better on classifying images with high-diversity and nonhomogeneous spatial distributions because they can learn more abstract and discriminative semantic features.

Within the DRB classifier, the low-level feature descriptors, 1) GIST [217], and 2) Histogram of Oriented Gradients (HOG) [218], are employed, and a combination of both is also used to improve their descriptive ability. However, as the low-level feature descriptors are not enough to handling complex, large-scale problems, one of the most widely used high-level feature descriptors, namely, the pre-trained VGG-VD-16 [23], is also introduced into the DRB classifier. It has to be stressed that the high-level feature descriptor is directly used within the DRB classifier without further tuning.

As there is no interdependence within the feature extraction of different images, the feature extraction process can be parallelised in a very large scale to further reduce the processing time. Once the global features (either low- or high- level) of the image are extracted and stored, there is no need to repeat the same process again. It has to be stressed that this thesis describes a general DRB approach and the feature descriptors do not need to be limited to GIST or HOG or the pre-trained VGG-VD-16 only. Alterative feature descriptors can also be used, and selecting the most suitable feature descriptor for a particular problem requires prior knowledge about this problem, also fine-tuning the high-level feature descriptor to the specific problem can also enhance the performance as well. One may also consider using data-driven feature selection techniques to select the optimal input feature vectors through an iterative searching process [219]–[222]. However, these are out of the scope of this thesis.

5.2.3.1. Employed Low-Level Feature Descriptors

A. GIST Descriptor

GIST feature descriptor gives an impoverished and coarse version of the principal contours and textures of the image [217]. In the proposed DRB approach, the same GIST descriptor is used as described in [217] without any modification. It extracts a 1×512 dimensional GIST feature vector denoted by $\mathbf{gist}(\mathbf{I}) = [\mathrm{gist}_1(\mathbf{I}), \mathrm{gist}_2(\mathbf{I}), \dots, \mathrm{gist}_{512}(\mathbf{I})]^T$, where **I** denotes the image.

B. HOG Descriptor

HOG descriptor [218], [223] has been proven to be very successful in various computer vision tasks such as object detection, texture analysis and image classification. In the proposed DRB approach, although the size of the images varies for different problems, w the default block size of 2×2 is used and the cell size is changed to fix the dimensionality of the HOG features to be 1×576, denoted by $\mathbf{hog}(\mathbf{I}) = [\mathrm{hog}_1(\mathbf{I}), \mathrm{hog}_2(\mathbf{I}), \dots, \mathrm{hog}_{576}(\mathbf{I})]^T$, which is experimentally found to be the most effective.

To improve the distinctiveness of the HOG feature and expand the range of the HOG features values, the nonlinear nonparametric function (equation (5.3)) is employed [41], [42] and the resulting nonlinearly mapped HOG features are denoted by $\kappa\big(\mathbf{hog}(\mathbf{I})\big)$.

C. Combined GIST-HOG Features

To improve the descriptive ability and effectiveness of the used features, the GIST and HOG is further combined to create new, more descriptive integrated feature set as follows:

$$\mathbf{cgh}(\mathbf{I}) = \left[\frac{\mathbf{gist}(\mathbf{I})^T}{\|\mathbf{gist}(\mathbf{I})\|}, \frac{\kappa\big(\mathbf{hog}(\mathbf{I})\big)^T}{\|\kappa\big(\mathbf{hog}(\mathbf{I})\big)\|}\right]^T, \tag{5.12}$$

where $\|\cdot\|$ denotes the norm.

5.2.3.2. Employed High-Level Feature Descriptor

The VGG-VD-16 [23] is one of the currently best performing pre-trained deep convolutional neural network (DCNN) models which are widely used in different works as the feature descriptor due to its simpler structure and better performance. The pre-trained VGG-VD-16 model is used without any tuning as the high-level feature descriptor of the DRB classifier to enhance its ability in handling complex, large-scale, high-density image classification problems. Following the common practice, the 1×4096 dimensional activations from the first fully connected layer are extracted as the feature vector of the image, denoted by $\mathbf{vgg}(\mathbf{I}) = [\mathrm{vgg}_1(\mathbf{I}), \mathrm{vgg}_2(\mathbf{I}), \ldots, \mathrm{vgg}_{4096}(\mathbf{I})]^T$.

However, as the pre-trained model requires the input image to be the size of 227×227 pixels [23], it is, in fact, not good in handling problems with simple and small size images.

5.2.4. Massively Parallel Fuzzy Rule Base

In the DRB classifier, a non-parametric rule base formed of 0-order AnYa type [60] fuzzy rules is employed. It makes the DRB classifier interpretable and transparent for human understanding (even to a non-expert) unlike the celebrated deep learning. Because of its prototype-based nature, the DRB classifier is free from prior assumptions about the type of the data distribution, their random or deterministic nature, the requirements to set the ad hoc model structure, membership functions, number of layers, etc. Meanwhile, its nature allows the DRB classifier a non-parametric, non-iterative, self-organising, self-evolving and highly parallel underlying structure. The training of the DRB classifier is driven by the ALMMO-0 approach as described in section 4.2, and thus, the training process is fully autonomous, significantly faster and can start "from scratch".

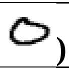
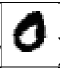
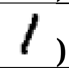
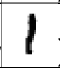
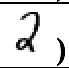
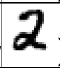
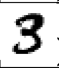
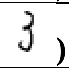
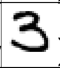
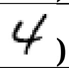
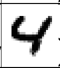
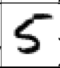
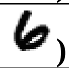
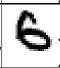
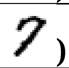
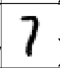
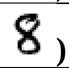
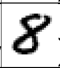
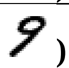
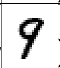
As described in more detail in section 4.2 as well as in [35], the system automatically identifies prototypes from the empirically observed data and forms data clouds resembling Voronoi tessellation [64] per class. Thus, for a training dataset, which consists of C classes, C independent 0-order fuzzy rule-based subsystems are trained (one per class) in parallel. After the training process is finished, each sub-classifier generalizes/learns one 0-order AnYa type fuzzy rule corresponding to its own class based on the identified prototypes:

$$IF\ (\mathbf{I} \sim \mathbf{p}_{c,1})\ OR\ (\mathbf{I} \sim \mathbf{p}_{c,2})\ OR \ldots OR\ (\mathbf{I} \sim \mathbf{p}_{c,P_c})\quad THEN\ (class\ c), \tag{5.13}$$

where $c = 1,2, \ldots, C$; $\mathbf{p}_{c,j}$ is the j^{th} visual prototype of the c^{th} class; $j = 1,2, \ldots, P_c$; P_c is the number of prototypes of the c^{th} class.

Examples of AnYa type fuzzy rules generalized from the popular handwritten digits recognition problem, MNIST dataset [213] for digits "0" ~ "9" are visualised in Table 2, where one can see that AnYa type fuzzy rules in the table provide a very intuitive representation of the mechanism. Moreover, each of the AnYa type fuzzy rules can be interpreted as a number of simpler fuzzy rules with single prototype connected by “OR” operator. As a result, a massive parallelisation is possible.

Table 2. Illustrative example of AnYa fuzzy rules with MNIST dataset

Fuzzy rule				
IF **(I~** **)** ***OR*** **(I~**	 **)** ***OR*** **(I~**	**)** ***OR*** **(I~**	 **)** ***OR … OR*** **(I~**	**)** ***THEN*** **(digit 0)**
IF **(I~** **)** ***OR*** **(I~**	 **)** ***OR*** **(I~**	**)** ***OR*** **(I~**	 **)** ***OR … OR*** **(I~**	**)** ***THEN*** **(digit 1)**
IF **(I~** **)** ***OR*** **(I~**	 **)** ***OR*** **(I~**	**)** ***OR*** **(I~**	 **)** ***OR … OR*** **(I~**	**)** ***THEN*** **(digit 2)**
IF **(I~**  **)** ***OR*** **(I~**	 **)** ***OR*** **(I~**	**)** ***OR*** **(I~**	 **)** ***OR … OR*** **(I~**	**)** ***THEN*** **(digit 3)**
IF **(I~** **)** ***OR*** **(I~**	 **)** ***OR*** **(I~**	**)** ***OR*** **(I~**	 **)** ***OR … OR*** **(I~**	**)** ***THEN*** **(digit 4)**
IF **(I~** **)** ***OR*** **(I~**	**)** ***OR*** **(I~**	**)** ***OR*** **(I~**	 **)** ***OR … OR*** **(I~**	**)** ***THEN*** **(digit 5)**
IF **(I~** **)** ***OR*** **(I~**	 **)** ***OR*** **(I~**	**)** ***OR*** **(I~**	 **)** ***OR … OR*** **(I~**	**)** ***THEN*** **(digit 6)**
IF **(I~** **)** ***OR*** **(I~**	 **)** ***OR*** **(I~**	**)** ***OR*** **(I~**	 **)** ***OR … OR*** **(I~**	**)** ***THEN*** **(digit 7)**
IF **(I~** **)** ***OR*** **(I~**	 **)** ***OR*** **(I~**	**)** ***OR*** **(I~**	 **)** ***OR … OR*** **(I~**	**)** ***THEN*** **(digit 8)**
IF **(I~** **)** ***OR*** **(I~**	 **)** ***OR*** **(I~**	**)** ***OR*** **(I~**	 **)** ***OR … OR*** **(I~**	**)** ***THEN*** **(digit 9)**

5.2.5. Decision-Making Mechanism

5.2.5.1. Local Decision-Making

After the system identification procedure (ALMMO-0 algorithm in section 4.2), the DRB system generates C fuzzy rules in regards to the C classes. For each testing image **I**, each one of the C fuzzy rules will generate a score of confidence $\lambda_c(\mathbf{I})$ ($c = 1,2,\dots,C$) by the local decision-maker within the fuzzy rule based on the global features of the image denoted by $\boldsymbol{x}$:

$$\lambda_c(\mathbf{I}) = \text{argmax}_{j=1,2,\dots,P_c}\left(\exp\left(-\|\boldsymbol{x}-\boldsymbol{p}_{c,j}\|^2\right)\right). \qquad (5.14)$$

As a result, one can get C scores of confidence $\boldsymbol{\lambda}(\mathbf{I}) = [\lambda_1(\mathbf{I}),\lambda_2(\mathbf{I}),\dots,\lambda_C(\mathbf{I})]$, which are the inputs of the decision-maker of the DRB classifier.

5.2.5.2. Overall Decision-Making

For a single DRB system, the label of the testing sample, denoted by $y(\mathbf{I})$, is given by the decision-maker, namely, the layer ④ in Figure 28, using the "winner takes all" principle:

$$y(\mathbf{I}) = \text{argmax}_{j=1,2,\dots,C}\left(\lambda_j(\mathbf{I})\right). \tag{5.15}$$

In some applications, i.e. face recognition, remote sensing, object recognitions, etc., where local information plays a more important role than the global information, one may consider to segment (both the training and testing) images to capture local information. In such cases, the 0-order FRB systems are trained with segments of training images instead of the full images. The overall label of a testing image is given as an integration of all the scores of confidence that the DRB subsystems give to its segments, denoted by $\mathbf{sg}_1, \mathbf{sg}_2, \dots, \mathbf{sg}_S$:

$$y(\mathbf{I}) = \text{argmax}_{j=1,2,\dots,C}\left(\frac{1}{S}\sum_{i=1}^{S}\lambda_j(\mathbf{sg}_i)\right). \tag{5.16}$$

If a DRB ensemble [201] is used, the label of the testing image is consider as the integration of all the scores of confidence that the DRB systems give to the image [41]:

$$y(\mathbf{I}) = \text{argmax}_{j=1,2,\dots,C}\left(\frac{1}{E}\sum_{i=1}^{E}\lambda_{j,i}(\mathbf{I}) + \max_{i=1,2,\dots,E}\lambda_{j,i}(\mathbf{I})\right), \tag{5.17}$$

where E is the number of DRB systems in the ensemble.

As one can see, the overall decision-making process of the DRB ensemble (equation (5.17)) takes both the overall confidence scores and the maximum confidence scores into consideration. Thus, it integrates the two types of most important information to make the judgement, which differs from the simple voting mechanism used in many other works [41].

5.3. Semi-Supervised DRB Classifier

In this section, the DRB classifiers [39], [41], [42] is extended with a semi-supervised learning strategy in a self-organising way and, thus, its ability of handling unlabelled images is enhanced [40]. Thanks to the prototype-based nature of the DRB classifier, the semi-supervised learning process is fully transparent and human-interpretable. It not only can perform classification on out-of-sample images, but also supports recursive online training on a sample-by-sample basis or a chunk-by-chunk basis. Moreover, unlike other semi-supervised approaches, the semi-supervised DRB (SSDRB) classifier is able to learn new classes actively without human experts' involvement, thus, to self-evolve [55].

Compared with the existing semi-supervised approaches [164]–[168], [170]–[175], [178], the SSDRB classifier has the following distinctive features because of its prototype-based nature [39]–[42]:

1) Its semi-supervised learning process is fully transparent and human-interpretable;

2) It can be trained online on a sample-by-sample or chunk-by-chunk basis;

3) It can classify out-of-sample images;

4) It is able to learn new classes (self-evolving).

The general architecture and principles of the DRB classifier have been introduced in the previous sections. In this section, in order to simplify the problem, the DRB classifier with the architecture depicted in Figure 29 is used. Nonetheless, it has to be stressed that the semi-supervised learning strategy is a general learning approach, and it is suitable for all types of DRB classifiers or ensembles [40].

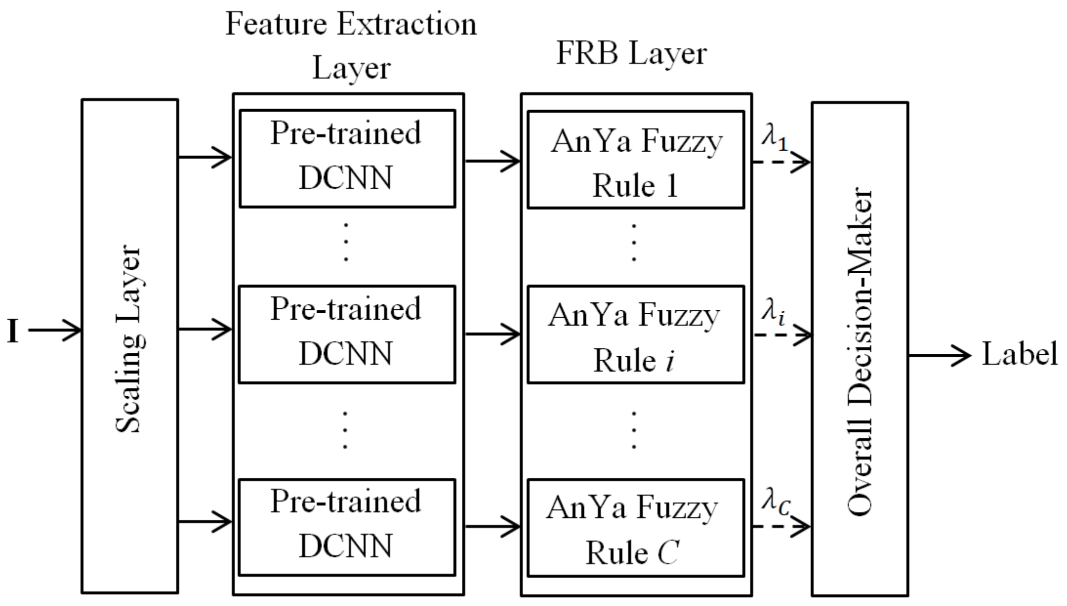


Figure 29. Architecture of the DRB classifier for semi-supervised learning.

One can see from Figure 29 that, the DRB classifier consists of the following four layers

1) Scaling layer;

This layer is for resizing the original size of the images to the desired size needed by the feature descriptors. In the DRB classifier used in this section, all the images are resized to the size of 227×227 pixels [23] because of the specific feature descriptor used.

2) Feature descriptor;

The pre-trained VGG-VD-16 DCNN [23] is used as the feature descriptor, and the 1×4096 dimensional activations from the first fully connected layer is used as the feature vector of the image [210]. This helps avoid handcrafting and automates the whole process. In the SSDRB classifier presented in this thesis, C pre-trained DCNNs are used (one per rule) to parallelise the feature extraction process. Nonetheless, one may also use only one pre-trained DCNN, but feature extraction would take more time.

3) Fuzzy rule base (FRB) layer [39], [41], [42];

4) Decision-maker.

The decision-maker makes the overall decision by equation (5.15).

In the following subsections, the semi-supervised learning strategies of the DRB classifier in both offline and online scenarios are described. A strategy for the DRB classifier to actively learn new classes from unlabelled training images is also presented.

5.3.1. Semi-supervised Learning Process from Static Datasets

5.3.1.1. Main Procedure of the Strategy

In an offline scenario, all the unlabelled training images are available and the DRB classifier starts to learn from these images after the training process with labelled images finishes.

First of all, the unlabelled training images are denoted as the set $\{\mathbf{U}\}$ and the number of unlabelled training images as L. The main steps of the semi-supervised learning strategy are described as follows [40]:

Step 1. Extract the score of confidence vector for each unlabelled training image, denoted by $\boldsymbol{\lambda}(\mathbf{U}_i)$ $(i = 1,2, \dots, L)$ using equation (5.14).

Step 2. Find out all the unlabelled training images satisfying Condition 29:

$$Condition\ 29{:}\quad IF\ \left(\lambda_{1^{st}max}(\mathbf{U}_i) > \varphi \cdot \lambda_{2^{nd}max}(\mathbf{U}_i)\right)\ \ THEN\ (\mathbf{U}_i \in \{\mathbf{V}\}_0), \tag{5.18}$$

where $\lambda_{1^{st}max}(\mathbf{U}_i)$ denotes the highest score of confidence $\mathbf{U}_i$ obtains, and $\lambda_{2^{nd}max}(\mathbf{U}_i)$ denotes the second highest score; φ ($\varphi > 1$) is a free parameter, in this thesis, $\varphi = 1.2$ is used; $\{\mathbf{V}\}_0$ denotes the collection of feature vectors of the unlabelled training images that meet Condition 29. After the elements of $\{\mathbf{V}\}_0$ are identified, they are removed from $\{\mathbf{U}\}$.

For the unlabelled training images that meet Condition 29, the DRB classifier is very confident about the class these images belong to and they can be used for updating the structure and meta-parameters of the DRB classifier. Otherwise, it means that the DRB classifier is not confident enough about its judgement and, thus, these images may not be used for updating the fuzzy rules.

Step 3. Rank the elements within $\{\mathbf{V}\}_0$ in a descending order in terms of the values of $\lambda_{1^{st}max}(\mathbf{V}) - \lambda_{2^{nd}max}(\mathbf{V})$ ($\mathbf{V} \in \{\mathbf{V}\}_0$), and denote the ranked set as $\{\mathbf{V}\}_1$.

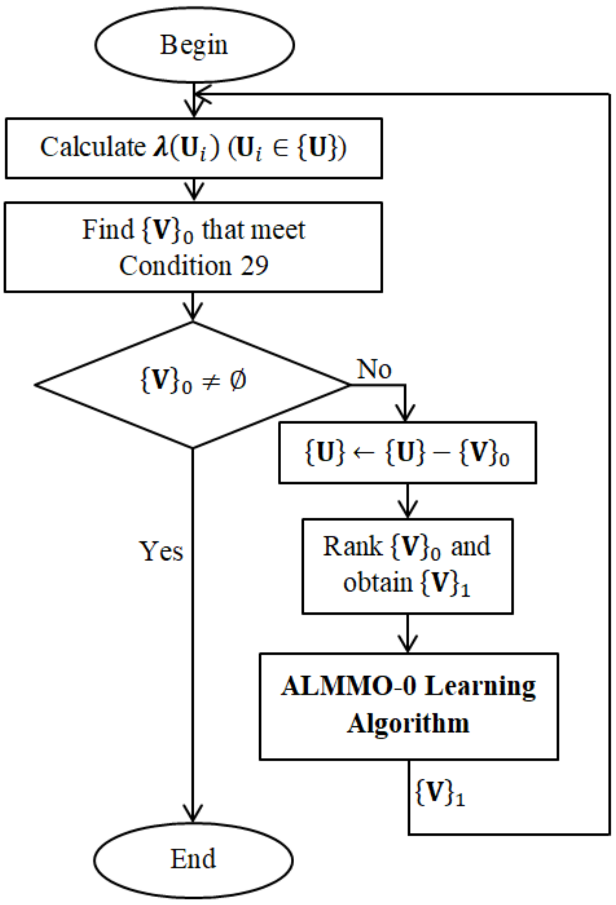
As one can see from the definition of the score of confidence equation (5.14) that, the higher $\lambda_{1^{st}max}(\mathbf{V})$ is, the more similar the image is to a particular prototype of the DRB classifier. Meanwhile, the higher $\lambda_{1^{st}max}(\mathbf{V}) - \lambda_{2^{nd}max}(\mathbf{V})$ is, the less ambiguous the decision made by the DRB classifier is. Since the DRB classifier learns sample-by-sample in the form of a data stream, by ranking $\{\mathbf{V}\}_0$ in advance, the classifier will firstly update itself with images that are more similar to the previously identified prototypes and have less ambiguity in the decisions of their labels, and later with the less familiar ones, which avoids overlapping and guarantees a more efficient learning.

Step 4. Update the DRB classifier using ALMMO-0 learning algorithm (section 4.2.2) with the set $\{\mathbf{V}\}_1$. Then the SSDRB classifier goes back to Step 1 and repeats the whole process, until there are no unlabelled training images that can meet Condition 29.

After the offline semi-supervised learning process is finished, if the DRB classifier is not designed to learn any new classes, the labels of all the unlabelled training images will be estimated using equation (5.15). Otherwise, the DRB classifier will, firstly, produce the labels of the images that can meet Condition 29 and then, learn new classes through the remaining unlabelled images (will self-evolve).

5.3.1.2. Algorithm Summary

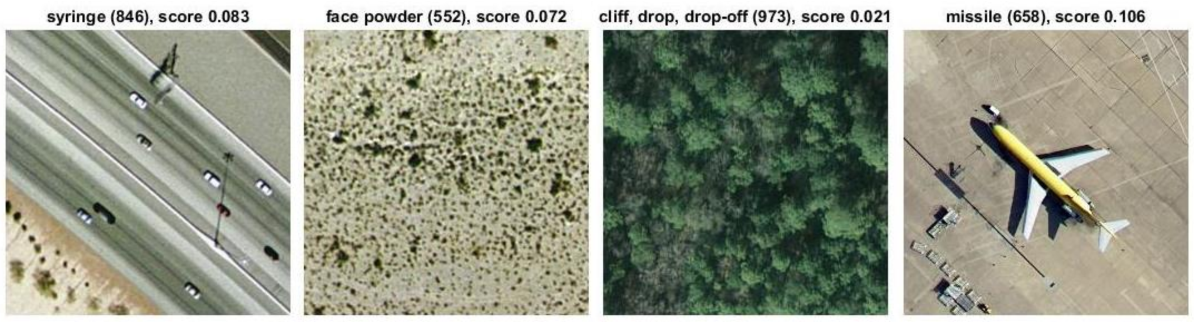
The main procedure of the offline semi-supervised learning process is summarised in a form of a flowchart in Figure 30.



Begin
Calculate $\boldsymbol{\lambda}(\mathbf{U}_i)$ ($\mathbf{U}_i \in \{\mathbf{U}\}$)
Find $\{\mathbf{V}\}_0$ that meet Condition 29
$\{\mathbf{V}\}_0 \neq \emptyset$
No
Yes
$\{\mathbf{U}\} \leftarrow \{\mathbf{U}\} - \{\mathbf{V}\}_0$
Rank $\{\mathbf{V}\}_0$ and obtain $\{\mathbf{V}\}_1$
ALMMO-0 Learning Algorithm
$\{\mathbf{V}\}_1$
End

Figure 30. Main procedure of the offline semi-supervised learning of SSDRB classifier.

5.3.2. Learning New Classes Actively



syringe (846), score 0.083
face powder (552), score 0.072
cliff, drop, drop-off (973), score 0.021
missile (658), score 0.106

Real label: **Freeway** Classified as: **Syringe** Score**:** **0.083**	Real label: **Chaparral** Classified as: **Face powder** Score: **0.072**	Real label: **Forest** Classified as: **Cliff, drop, drop-off** Score: **0.021**	Real label: **Airplane** Classified as: **Missile** Score: **0.106**

Figure 31. Misclassified images by VGG-VD-16 DCNN.

In real situations, the labelled training samples may fail to include all the classes due to various reasons, i.e. an insufficient prior knowledge or change of the data pattern in the perceived feature. For example, in Figure 31, despite the very low scores of confidence, the pre-trained VGG-VD-16 DCNN [23] recognises the three remote sensing images ("freeway", "chaparral" and "forest") as "syringe", "face powder" and "cliff, drop, drop-off", and for the image of an airplane which is taken from the top, the network recognises it as a "missile".

Therefore, it is of paramount importance for a classifier to be able to learn new classes actively, which not only guarantees the effectiveness of the learning process and reduces the requirement for prior knowledge, but also enables the human experts to monitor the changes of the data pattern. In this subsection, a strategy for the classifier to learn actively is introduced as follows [40].

5.3.2.1. Main Procedure of the Strategy

For an unlabelled training image (with its feature vector denoted by $\mathbf{U}_i$), if the Condition 30 is met, it means that the DRB classifier has not seen any similar images before, and therefore, a new class is being added.

$$Condition\ 30:\quad IF\ (\lambda_{1^{st}max}(\mathbf{U}_i) \leq \gamma)\quad THEN\ (\mathbf{U}_i \in the\ (C+1)^{th}\ class), \qquad (5.19)$$

where γ is a free parameter serving as the threshold. As a result, a new fuzzy rule is also added to the rule base with this training image as the first prototype. Generally, the lower γ is, the more conservative the DRB classifier will be when adds new rules to the rule base.

In the offline scenario, there may be a number of unlabelled images remaining in $\{\mathbf{U}\}$ after the offline semi-supervised learning process, re-denoted by $\{\mathbf{U}\}_1$. Some of these may satisfy Condition 30, denoted by $\{\mathbf{V}\}_2$, $\{\mathbf{V}\}_2 \subseteq \{\mathbf{U}\}_1$. Many of the images within $\{\mathbf{U}\}_1$ may actually belong to the a few unknown classes. To classify these images, the DRB classifier needs to add a few new fuzzy rules to the existing rule base in an active way.

The DRB classifier starts with the image that has the lowest $\lambda_{1^{st}max}$ (the corresponding feature vector is denoted by $\mathbf{V}_{min} \in \{\mathbf{V}\}_2$) and adds a new fuzzy rule with this image as the prototype. However, before adding another new fuzzy rule, the DRB classifier repeats the offline semi-supervised learning algorithm (the previous section) on the remaining unselected images within $\{\mathbf{U}\}_1$ to find other prototypes that are associated with the newly added fuzzy rule. This may solve the potential problem of adding too many excessive rules. After the newly formed fuzzy rule is fully updated, the DRB classifier will start to add the next new rule/class.

With this strategy, the SSDRB classifier is able to actively learn from the unlabelled training images, gain new knowledge, define new classes and add new rules/classes, correspondingly. Human experts can also examine the new fuzzy rules and give meaningful labels for the new classes by simply checking the prototypes afterwards, i.e. “new class 1” can be renamed as “agricultural” and “new class 2” can be renamed as “harbour”. This is less laborious than the usual approach as it only concerns the aggregated prototypical data, not the high volume raw data, and it is more important for the human users. However, it is also necessary to stress that identifying new classes and labelling them with human-understandable labels are not essential for the DRB classifier to work since in many applications, the classes of the images are predictable based on common knowledge. For example, for handwritten digits recognition problem, there will be images from 10 classes (from “0” to “9”), for Latin characters recognition problem, there will be images from 52 classes (from “a” to “z” and “A” to Z”), etc.

5.3.2.2. Algorithm Summary

The main procedure of actively learning new classes from unlabelled training data (self-evolving) of the SSDRB classifier is summarised in the following flowchart.

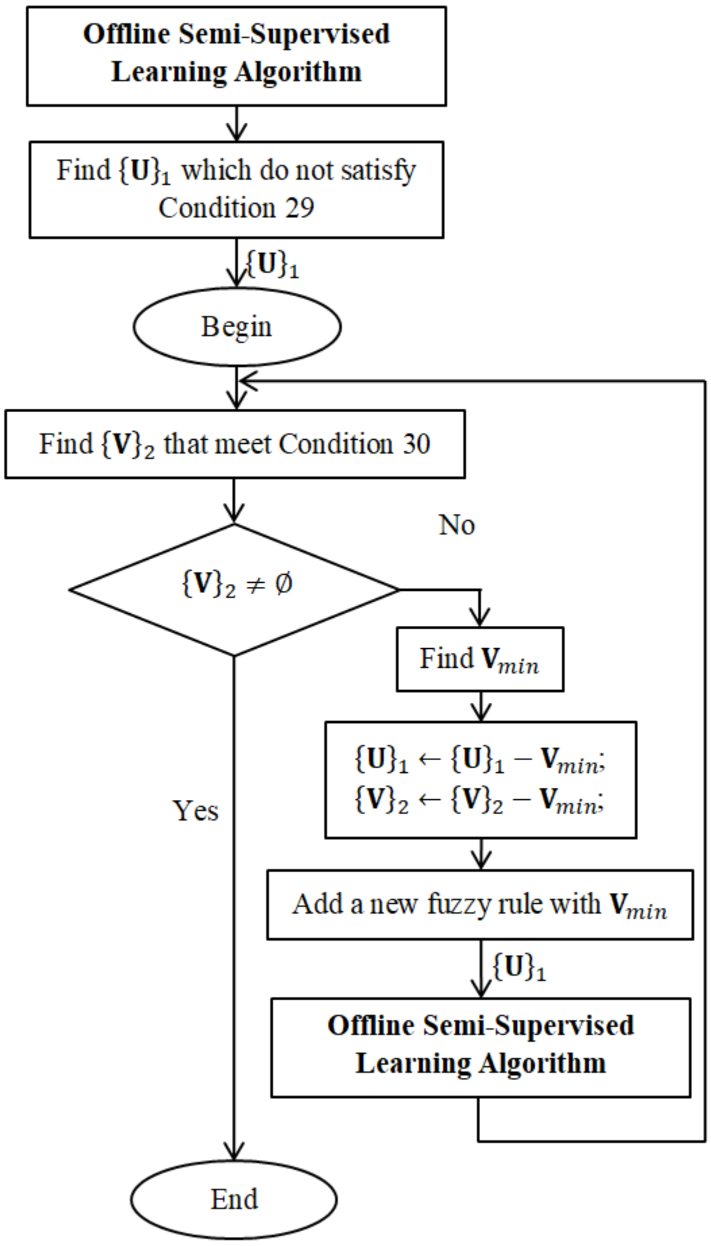

Offline Semi-Supervised Learning Algorithm
Find $\{\mathbf{U}\}_1$ which do not satisfy Condition 29
$\{\mathbf{U}\}_1$
Begin
Find $\{\mathbf{V}\}_2$ that meet Condition 30
$\{\mathbf{V}\}_2 \neq \emptyset$
No
Find $\mathbf{V}_{min}$
$\{\mathbf{U}\}_1 \leftarrow \{\mathbf{U}\}_1 - \mathbf{V}_{min}$;
$\{\mathbf{V}\}_2 \leftarrow \{\mathbf{V}\}_2 - \mathbf{V}_{min}$;
Yes
Add a new fuzzy rule with $\mathbf{V}_{min}$
$\{\mathbf{U}\}_1$
Offline Semi-Supervised Learning Algorithm
End

Figure 32. Main procedure of the active learning of SSDRB classifier.

5.3.3. Semi-supervised Learning from Data Streams

It is often the case that, after the algorithms have processed the available static data, new data continuously arrives in a form of data stream. Prior semi-supervised approaches [164]–[168], [170]–[175], [178] are limited to offline application due to their operating mechanism. Thanks to the prototype-based nature and the evolving mechanism [55], [160] of the DRB classifier [39], [41], [42], online semi-supervised learning can also be conducted .

The online semi-supervised learning of the DRB classifier can be conducted on a sample-by-sample basis or a chunk-by-chunk basis after the supervised training process with the labelled training images finishes. In this subsection, the main procedures of the semi-supervised learning processes of both types together with their corresponding flowcharts are presented.

The online semi-supervised learning strategy is the modification of the offline one as described in section 5.3.1. However, it has to be stressed that the performance of semi-supervised learning in an online scenario is influenced by the order of the images and is not as stable as the semi-supervised learning in an offline scenario.

5.3.3.1. Online Semi-Supervised Learning on a Sample-by-Sample Basis

The main steps of the online semi-supervised learning on a sample-by-sample basis are as follows [40]:

Step 1. Use Condition 29 and ALMMO-0 learning algorithm (section 4.2.2) to learn from the available unlabelled image denoted by $\mathbf{U}_{L+1}$;

Step 2 (optional). Check Condition 30 to see whether the DRB classifier needs to add a new rule (and class).

Step 3. DRB classifier goes back to Step 1 and processes the next image.

5.3.3.2. Online Semi-Supervised Learning on a Chunk-by-Chunk Basis

The main steps of the online semi-supervised learning on a chunk-by-chunk basis are as follows [40]:

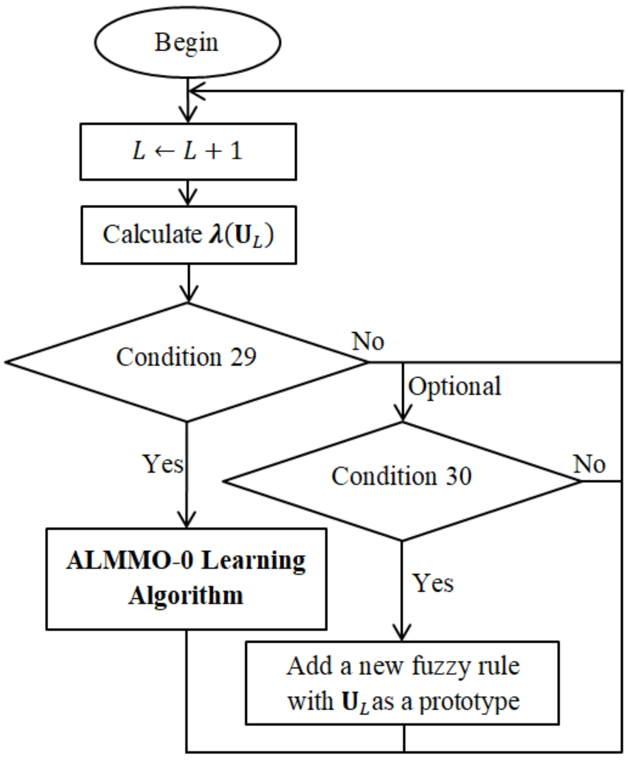
Step 1. Use offline semi-supervised learning algorithm (section 5.3.1) to learn from the available chunk of unlabelled images denoted by $\{\mathbf{U}\}_1$;

Step 2 (optional). Use active learning algorithm (section 5.3.2) to actively learn new classes from the remaining images denoted by $\{\mathbf{U}\}_2$;

Step 3. DRB classifier goes back to Step 1 and processes the next chunk.

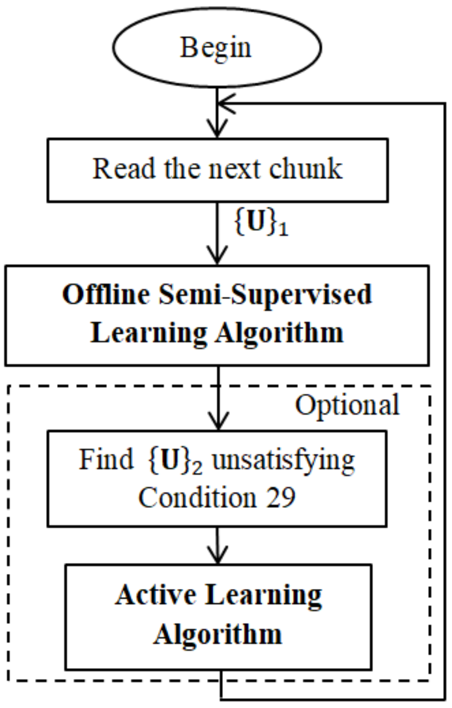
5.3.3.3. Algorithm Summary

The main procedures of online semi-supervised learning strategies (on a sample-by-sample basis and on a chunk-by-chunk basis) of the SSDRB classifier are summarised in the following flowcharts.



Begin
$L \leftarrow L+1$
Calculate $\boldsymbol{\lambda}(\mathbf{U}_L)$
Condition 29
No
Optional
Yes
Condition 30
No
ALMMO-0 Learning Algorithm
Yes
Add a new fuzzy rule with $\mathbf{U}_L$ as a prototype

(a) Semi-supervised learning on a sample-by-sample basis



Begin
Read the next chunk
$\{\mathbf{U}\}_1$
Offline Semi-Supervised Learning Algorithm
Optional
Find $\{\mathbf{U}\}_2$ unsatisfying Condition 29
Active Learning Algorithm

(b) Semi-supervised learning on a chunk-by-chunk basis

Figure 33. Main procedure of the online semi-supervised learning of SSDRB classifier.

5.4. Examples of DRB Ensembles

In some real applications, a single DRB classifier may not be sufficient. For example, a DRB classifier may take too much time to learn from a large scale image set, or multiple types of feature descriptors are necessary for achieving a good classification result. Thus, an ensemble of DRB classifiers is needed.

In this section, three successful examples of DRB ensembles based on the real problems (handwritten digits recognition and remote sensing scenes classification) are presented aiming to demonstrate the idea of how to create an ensemble with the DRB classifiers to improve the performance. Nonetheless, it has to be stressed that the DRB system is a general system for image classification, there is no fixed principle for creating ensembles, and DRB ensembles can be formed in different ways subjected to the requirements of the problems and the goals [39], [41], [42].

5.4.1. DRB Committee for Handwritten Digits Recognition

The DRB committee was designed for recognising the handwritten digit images of the well-known benchmark dataset MNIST [213], which consist of 60000 training images and 10000 testing images from 10 classes in regards to the digits “0”~“9”. The size of the images is 28×28 pixels.

5.4.1.1. Training Stage

The architecture of the DRB ensemble for handwritten digits recognition in the training stage is depicted in Figure 34.

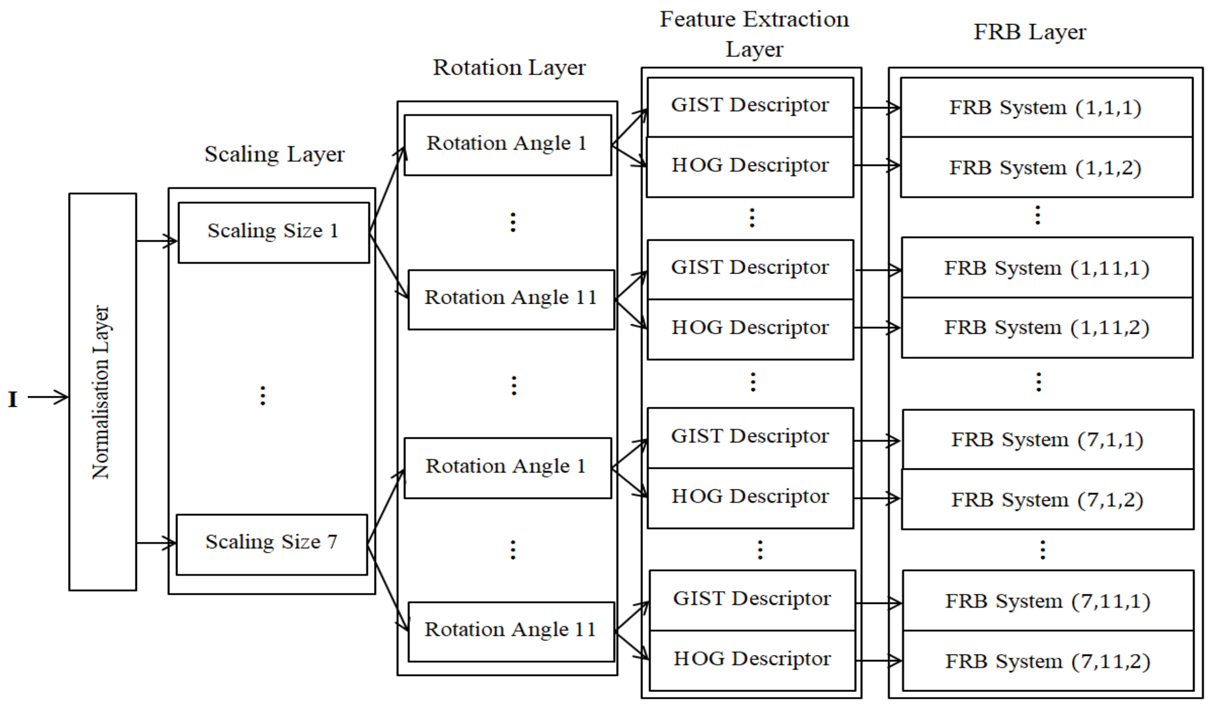


Figure 34. Architecture of DRB committee for training.

As one can see from Figure 34, the ensemble of DRB classifiers consists of the following components:

① Normalisation layer, which normalise the original value range of the pixels of the handwritten images from [0,255] to [0,1] linearly;

② Scaling layer, which resizes the images from the original image size of 28×28 pixels into seven different sizes: 1) 28×22 ; 2) 28×24; 3) 28×26; 4) 28×28; 5) 28×30; 6) 28×32 and 7) 28×34;

③ Rotation layer, which rotates each image (after the scaling operation) into 11 different angles from $-15°$ to $15°$ with the interval of $3°$;

Therefore, the scaling and rotation layer expands the original training image set into 77 new training sets with different scaling sizes and rotation angles.

④ Feature extraction layer. In this layer, a low-level feature descriptor, namely GIST [217] or HOG [218] (or both of them), is employed, which extract a 1×512 dimensional GIST feature vector or a 1×576 dimensional HOG feature vector or a 1×1088 dimensional combined GIST-HOG feature vector from each training image, respectively, as described in section 5.2.3.1;

⑤ FRB layer, which consists of 154 FRB systems, each of them is trained with one of the two types of feature vectors from one of the 77 expanded training sets. Each FRB system has 10 FRB subsystems (corresponding to 10 digits “0” to “9”) and each subsystem has one massively parallel 0-order fuzzy rule of AnYa type as described in section 5.2.4. As a result, the FRB layer, in total, has 1540 0-order fuzzy rules of AnYa type, and each one of them is trained separately. The training process of these fuzzy rules is described in section 4.2.2.

5.4.1.2. Classification Stage

The architecture of the DRB ensemble for handwritten digits recognition in the classification stage is depicted in Figure 35. As one can see from the figure, during the classification stage, the ensemble of DRB classifiers consists of the following components:

① Normalisation layer;

② Feature extraction layer;

③ FRB layer;

The normalisation layer and feature extraction layer are the same as used during the training stage. The FRB layer contains all the 1540 0-order fuzzy rules of AnYa type identified through the training.

④ Decision-making committee, which decides the label of the testing image based on the 1540 scores of confidences generated by the 1540 fuzzy rules (equation (5.14)) in the FRB layer following equation (5.17).

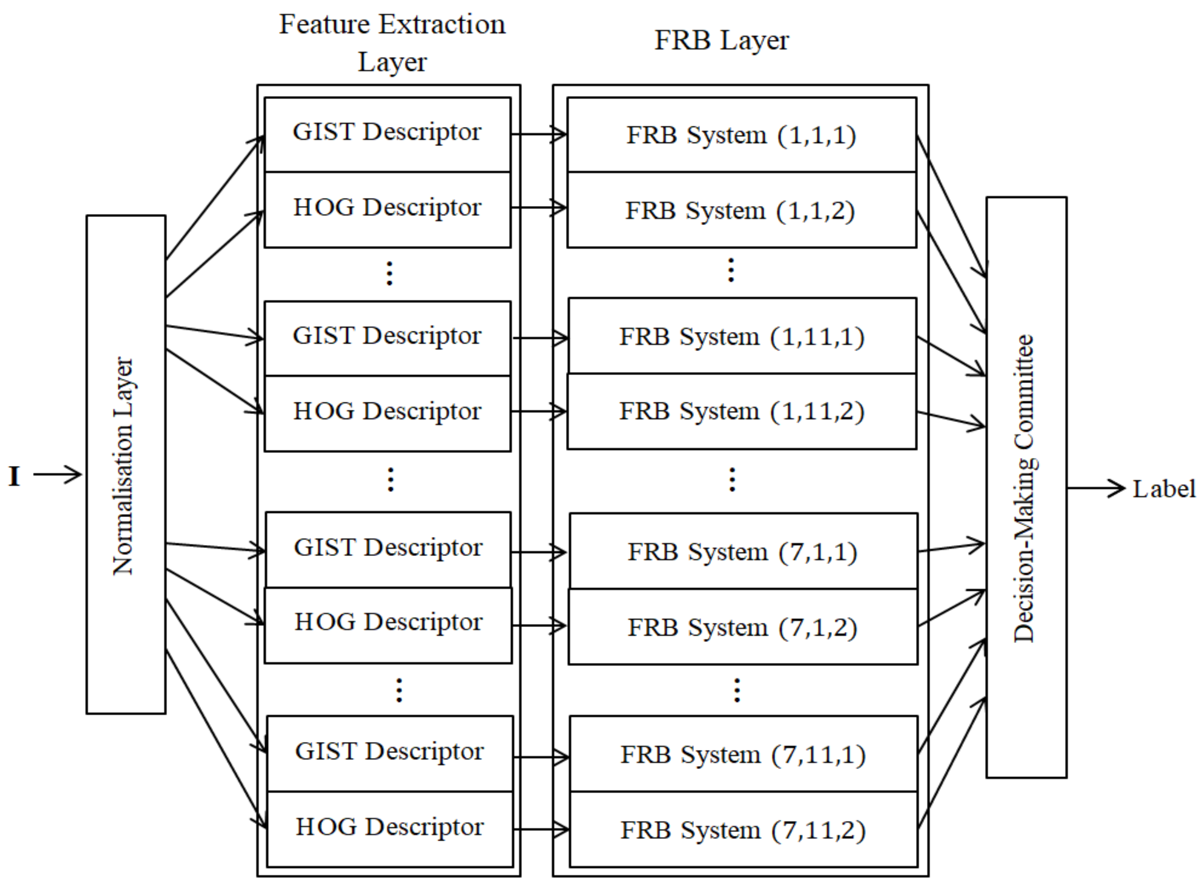


Figure 35. Architecture of DRB committee for classification.

5.4.2. A Cascade of DRB and SVM for Handwritten Digits Recognition

The cascade of the DRB and SVM was introduced in [42] for the MNIST dataset. The diagram of the cascade is given in Figure 36.

One can see from Figure 36 that the DRB ensemble is used as the main engine of the approach and a SVM based conflict resolution classifier is added in a cascade configuration to support the main engine when there is no clear winner but rather a conflict in the degree of confidence of the best two class suggestions.

The DRB committee [41] is able to perform highly accurate classification on the handwriting digits in the majority cases by following the "winner takes all" principle. However, it fails in the rare cases (less than 2% in the MNIST handwriting digits recognition problem [213]) in which there are two highly confident labels generated for a single image at the same time. In these rare cases, the "winner takes all" principle used in the DRB committee blindly assigns the class to the winner ignoring the fact that the second best is also likely.

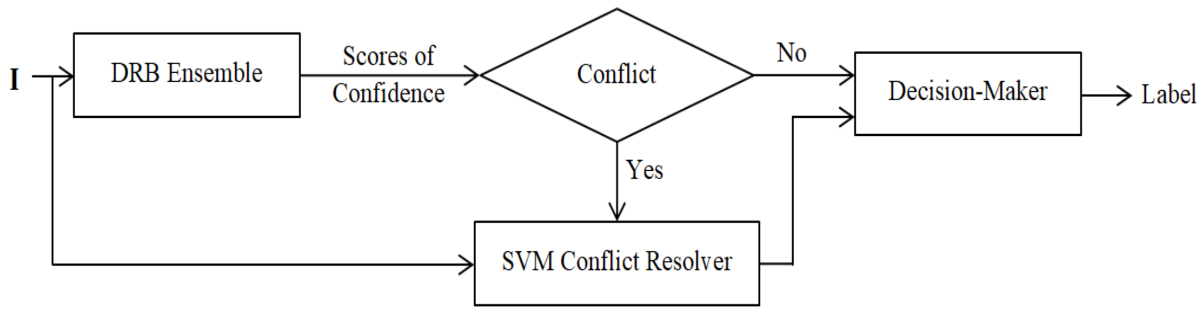


Figure 36. Diagram of the cascade of DRB ensemble and SVM.

Therefore, in this approach, a conflict resolution classifier is added as the auxiliary stage for the main classifier in making decisions when there are two highly confident labels produced for the same image. This conflict resolution classifier is using an SVM classifier [224] with polynomial kernel and it improves the overall performance of the DRB ensemble.

The learning process of the SVM is independent from the DRB ensemble, and, thus, can be trained in parallel and will not influence the evolving nature of the DRB ensemble.

5.4.2.1. The DRB Ensemble

The architecture of the DRB ensemble for the training is depicted in Figure 37, which has one extra layer compared with the DRB ensemble depicted in Figure 34. The DRB committee used in the cascade consists of the following components:

① Normalisation layer;

② Scaling layer;

③ Rotation layer;

④ Segmentation layer;

⑤ Feature extraction layer;

⑥ FRB layer.

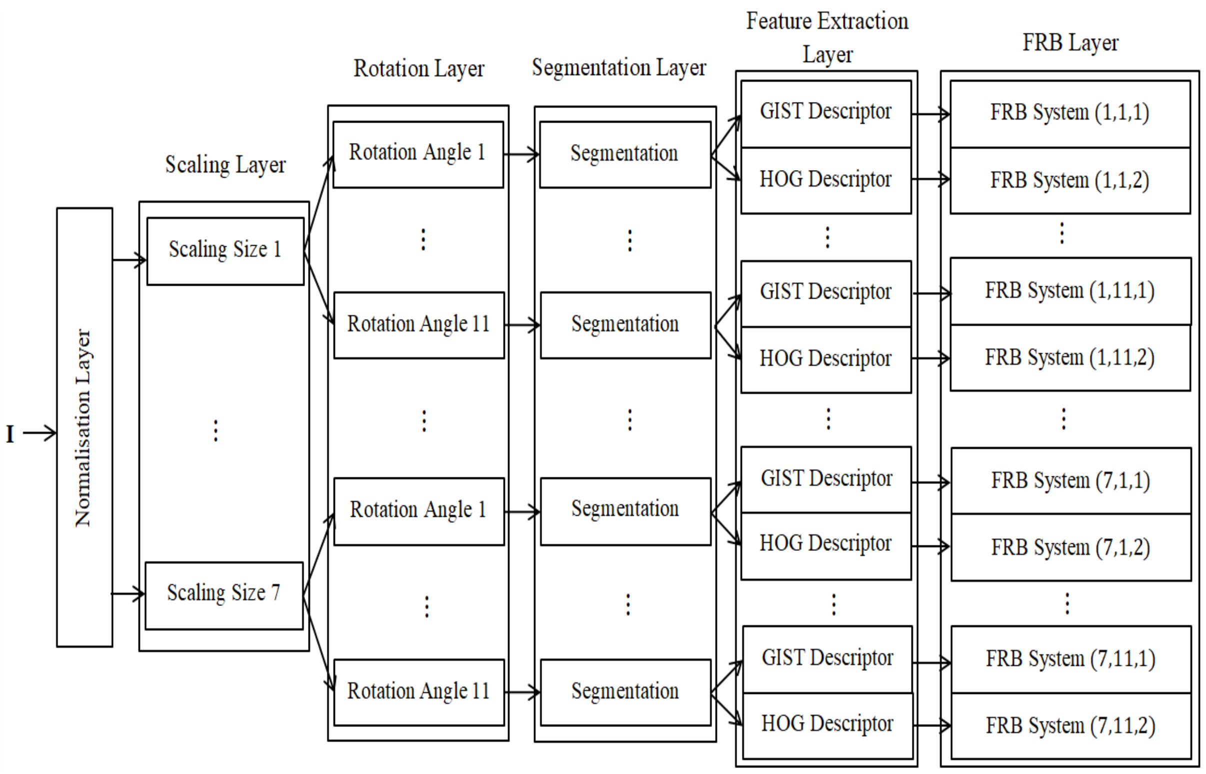


Figure 37. Architecture of the DRB ensemble for training.

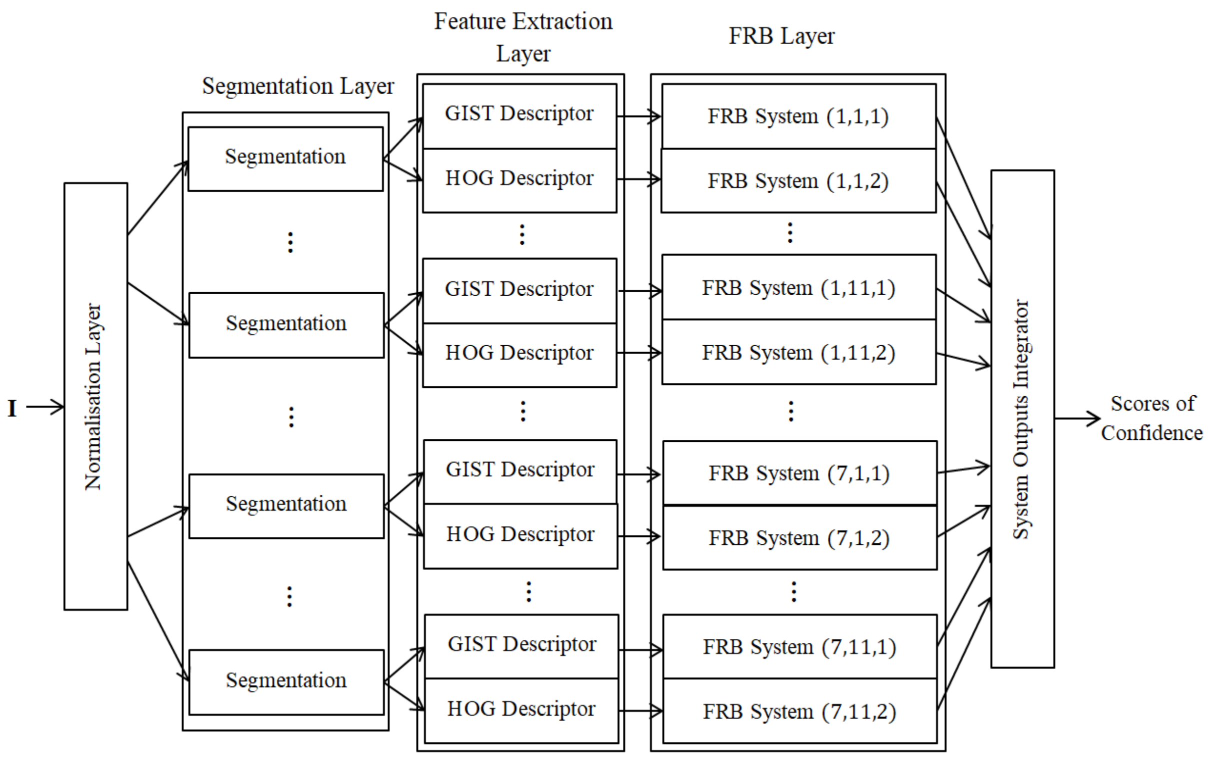


Figure 38. Architecture of the DRB ensemble for classification.

The ① - ③ and ⑤ - ⑥ layers are the same as used in section 5.4.1.1. The extra segmentation layer is for extracting the central area (22×22) from the training images. It discards the borders that mostly consist of white pixels with little or no information.

The architecture of the DRB ensemble for the classification is depicted in Figure 38, from which one can see that, the decision-making committee is replaced by a system output integrator that integrate the outputs of the 1540 fuzzy rules into 10 scores of confidences by the following equation:

$$\Lambda_c(\mathbf{I}) = \frac{1}{154}\sum_{i=1}^{154}\left(\lambda_{c,i}(\mathbf{I})\right) = \frac{1}{154}\sum_{i=1}^{154}\left(\max_{j=1,2,\dots,P_{c,i}}\left(\exp\left(-\left\|\boldsymbol{x} - \boldsymbol{p}_{c,i,j}\right\|^2\right)\right)\right), \quad (5.20)$$

where $c = 1,2,\dots,10$; $P_{c,i}$ is the number of prototypes within the i^{th} fuzzy rule of the c^{th} class; $\boldsymbol{p}_{c,i,j}$ is the corresponding j^{th} prototype.

5.4.2.2. Conflict Detector

The conflict detector will detect the rare cases in which the highest and the second highest overall scores of confidence given by the decision-making committee are very close. If such cases happen, it means that there is a conflict, and the decision-maker will involve the SVM conflict resolution classifier. The principle for detecting a conflict is as follows:

$$Condition\ 31: \quad \begin{matrix} IF\ \left(\Lambda_{1^{st}max}(\mathbf{I}) \leq \Lambda_{2^{nd}max}(\mathbf{I}) + \sigma_{\Lambda(\mathbf{I})}/4\right) \\ THEN\ (A\ conflict\ is\ detected) \end{matrix}, \quad (5.21)$$

where $\Lambda_{1^{st}max}(\mathbf{I})$ and $\Lambda_{2^{nd}max}(\mathbf{I})$ are the largest and the second largest integrated scores of confidence; σ_Λ is the standard deviation of the 10 integrated scores given by the DRB committee the testing image.

5.4.2.3. The SVM Conflict Resolution Classifier

In the cascade approach, the SVM conflict resolution classifier is added to assist the DRB ensemble when it produces two highly confident labels on one image. The architecture of the SVM conflict resolution classifier is given in Figure 39.

As one can see, the SVM conflict resolution classifier consists of the following layers:

① Normalisation layer;

② Segmentation layer;

③ Feature extraction layer;

④ SVM classifier.

The normalization and segmentation layers are the same as used in the DRB ensemble. The feature extraction layer extracts the combined GIST and HOG features of the images as described by equation (5.12), which can improve the classification accuracy of the SVM classifier as the combined feature is relatively more descriptive compared with the original ones.

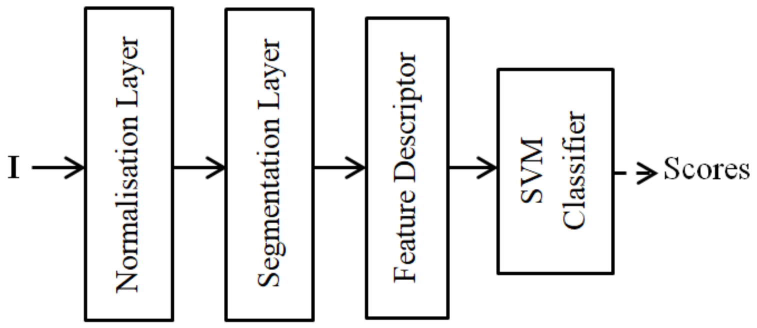


Figure 39. Architecture of the SVM conflict resolution classifier.

The SVM classifier during the classification stage conducts a binary classification on the image for the first and second most likely classes it belongs to. The output of the SVM conflict resolution classifier is not the label but the scores, which are denoted by $\Lambda_{s1}(\mathbf{I})$ and $\Lambda_{s2}(\mathbf{I})$, which correspond to $\Lambda_{1^{st}max}(\mathbf{I})$ and $\Lambda_{2^{nd}max}(\mathbf{I})$, respectively.

5.4.2.4. Decision Maker

During the classification stage, the SVM based conflict resolution classifier will not be functioning if Condition 31 is not satisfied. In such case, the decision-maker will make decision directly based on the maximum $\Lambda(\mathbf{I})$ obtained by the DRB ensemble.

If there is a conflict detected, the decision-maker will do a binary classification on the image between the first and second most likely classes with the assistance of the SVM based conflict resolver:

$$y(\mathbf{I}) = \text{argmax}\left(\left\{\Lambda_{1^{st}max}(\mathbf{I}) + \Lambda_{s1}(\mathbf{I}), \Lambda_{2^{nd}max}(\mathbf{I}) + \Lambda_{s2}(\mathbf{I})\right\}\right). \qquad (5.22)$$

5.4.3. DRB Ensemble for Remote Sensing Scenes

Land use classification is recognized widely as a challenging task because the land use sub-regions are recognised implicitly through their high-level semantic function, where multiple low-level features or land cover classes can appear in one land use category, and identical land cover classes can be shared among different land use categories. These high-level semantics need to be exploited sufficiently using robust and accurate approaches for feature representation.

In this section, by creating an ensemble of DRB classifiers [43] trained with segments of remote sensing images partitioned with different granularities, the DRB ensemble [43] is able to utilize spatial information at multiple scales and exhibit highly accurate classification performance.

5.4.3.1. General Architecture

The architecture of the DRB ensemble is given in Figure 40, which consists of four DRB classifiers trained with the segments of remote sensing images at four different levels of granularity (small, medium and large), which are achieved by using the sliding windows of three different sizes [43].

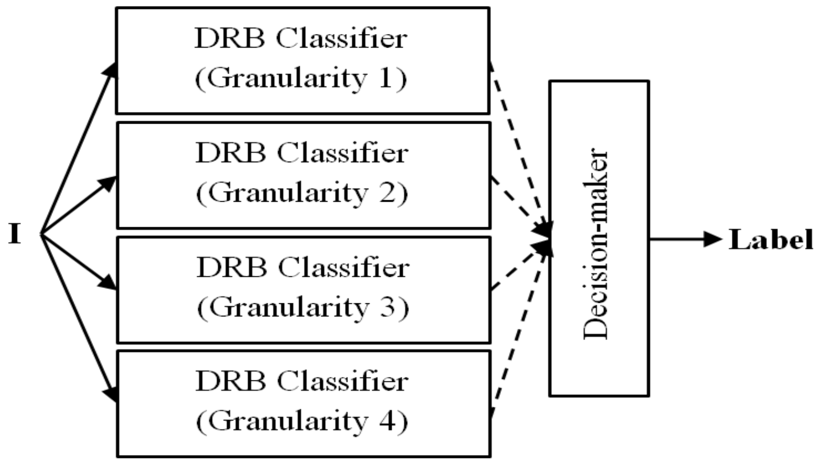

I
DRB Classifier (Granularity 1)
DRB Classifier (Granularity 2)
DRB Classifier (Granularity 3)
DRB Classifier (Granularity 4)
Decision-maker
Label

Figure 40. Architecture of the DRB ensemble for remote sensing scenes.

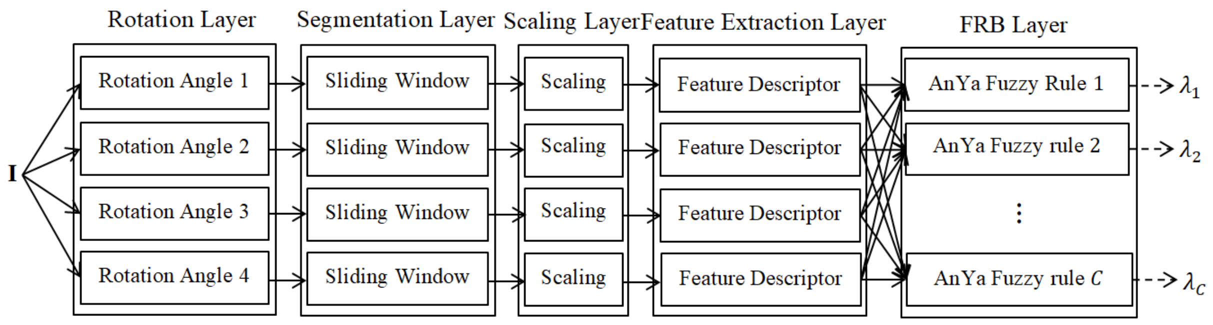


Figure 41. Structure of the DRB classifier for remote sensing scenes.

The architecture of a DRB classifier is presented in a modular/layered form in Figure 41. The decision-maker in the final layer of the ensemble decides the winning label of the validation images based on the suggestions of the individual (per class) IF… THEN… rules of the three DRB classifiers within the ensemble.

5.4.3.2. DRB Classifiers for Remote Sensing Scenes

The DRB classifier as depicted in Figure 42 has the following components [43]:

① Rotation layer, which rotates each remote sensing image at four different angles 1) 0°; 2) 90°; 3) 180° and 4) 270°.

② Segmentation layer, which uses a sliding window to partition the remote sensing images into smaller pieces for local information extraction. By changing the size of the sliding window, the level of granularity of the segmentation result can be changed accordingly. A larger sliding window size allows the DRB to capture coarse scale spatial information at the cost of losing fine scale detail and vice versa.

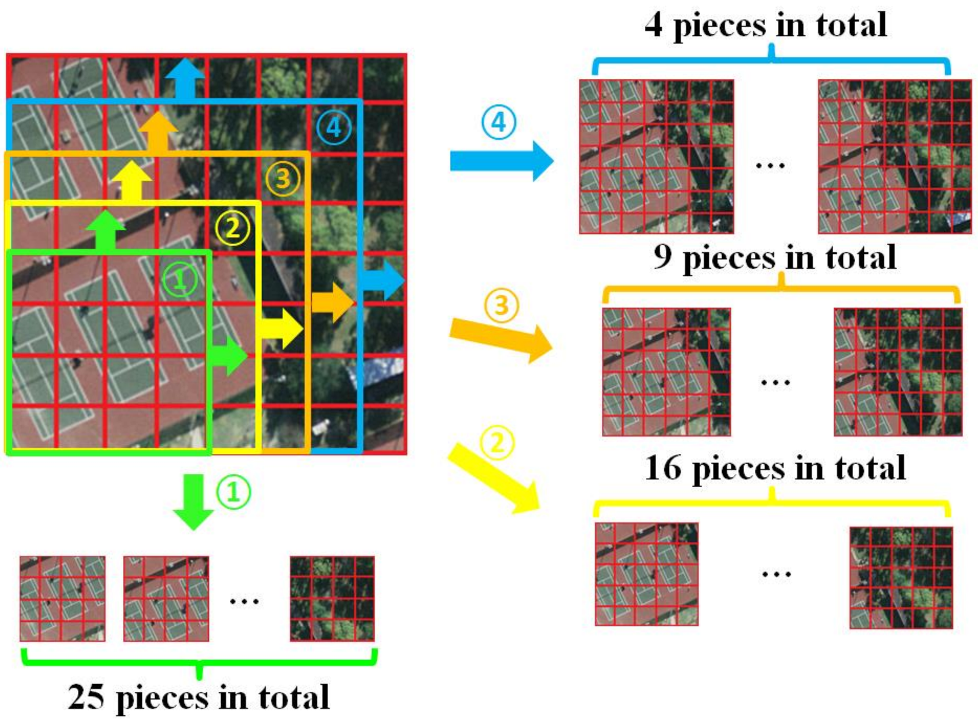


Figure 42. Image segmentation with different sliding windows.

In this example, three different sliding windows with sizes of 1) $(4\times4)/(8\times8)$ of image size (very small granularity); 2) $(5\times5)/(8\times8)$ of image size (small granularity); 3) $(6\times6)/(8\times8)$ of image size (medium granularity) and 4) $(7\times7)/(8\times8)$ of image size (large granularity) and the step size of $1/8$ width in the horizontal and $1/8$ length in the vertical direction. The segmentation process is illustrated in Figure 42.

③ Scaling layer, which is involved in the DRB classifier to rescale the segments into the uniform size of 227×227 pixels required by the VGG-VD-16 model [23].

④ Feature extraction layer, which is the high-level feature descriptor, namely the VGG-VD-16 model [23], as described in section 5.2.3.2.

⑤ FRB layer, which has been described in section 5.2.4.

5.4.3.3. Decision-Maker

During the validation stage, for each testing image, each DRB classifier can produce a label for the testing image in the way as described by equation (5.16). Since there are four DRB classifiers used, the simple voting mechanism is insufficient to utilise all the information. Therefore, the decision-maker uses a modified version of equation (5.16) to as follows [43]:

$$y(\mathbf{I}) = \text{argmax}_{j=1,2,\ldots,C}\left(\sum_{k=1}^{4}\frac{1}{S_k}\sum_{i=1}^{S_k}\lambda_{k,j}\left(\mathbf{sg}_{k,i}\right)\right), \tag{5.23}$$

where S_k is the number of segments obtained from the testing image under the k^{th} granularity; $\mathbf{sg}_{k,i}$ is the i^{th} segment of the corresponding granularity; $k = 1,2,3,4$.

5.5. Conclusion

In this chapter, the latest deep learning networks developed within the EDA framework for image classification are presented. Compared with the state-of-the-art deep learning based approaches, the presented work has the following distinctive features:

1) Highly efficient, transparent, human interpretable learning process;

2) Self-organising and self-evolving structure;

3) Free from the ad hoc decisions.

6. Implementation and Validation of the Developed Algorithms

In this chapter, the performance evaluation for the presented machine learning algorithms conducted on the benchmark datasets is presented. This chapter is organised as follows. The numerical experiments with the self-organising unsupervised machine learning algorithms described in chapter 3 are given in section 6.1. The results of the self-organising supervised machine learning algorithms are presented in section 6.2. The implementation and experiment results of the transparent deep learning systems are described in section 6.3. This chapter is finalised by section 6.4. This chapter is concluded by the final section.

The algorithms presented in this thesis are implemented on Matlab platform, and most of the numerical examples are conducted with Matlab2017a on a PC with WIN10 OS, dual core i7 processor with clock frequency 3.4 GHz each and 16GB RAM.

The source codes of the proposed algorithms are downloadable from:

https://uk.mathworks.com/matlabcentral/fileexchange/?term=authorid%3A1098949

6.1. Evaluation of the Unsupervised Learning Algorithms

The performance of the self-organising unsupervised machine learning algorithms presented in chapter 3, namely

1) Autonomous data-driven (ADD) clustering algorithm;

2) Hypercube-based data partitioning (HCDP) algorithm;

3) Autonomous data partitioning (ADP) algorithm;

4) Self-organising direction-aware (SODA) data partitioning algorithm;

are evaluated based on benchmark datasets. During the experiments, it is assumed that there is no any prior knowledge about the datasets. The following state-of-the-art algorithms are involved in the comparison:

1) Mean-shift clustering (MSC) algorithm [114];

2) Subtractive clustering (SUBC) algorithm [15];

3) Self-organizing map (SOM) algorithm [109];

4) Density peaks clustering (DPC) algorithm [225];

5) Density-based spatial clustering of applications with noise (DBSCAN) algorithm [6];

6) Affinity propagation clustering (APC) algorithm [105];

7) eClustering algorithm [11];

8) Evolving local means clustering (ELMC) algorithm [12];

9) Nonparametric mixture model based clustering (NMMBC) algorithm [122];

10) Nonparametric mode identification based clustering (NMIBC) algorithm [226];

11) Clustering of evolving data streams (CEDS) algorithm [118].

Due to insufficient prior knowledge, the recommended settings of the free parameters from the published literature are used throughout the numerical experiments. The experimental settings of the free parameters of the algorithms are presented in Table 3.

Table 3. Experimental settings of the comparative algorithms

Algorithms	Free Parameter(s)	Experimental setting
MSC	1) bandwidth, p 2) kernel function type	1) $p = 0.15$ [12] 2) Gaussian kernel
SUBC	initial cluster radius, r	$r = 0.3$ [15]
SOM	net size	12×12 [110]
DPC	1) minimum distance, r 2) local density value, δ	1) relatively high, r 2) high, δ [225]
DBSCAN	1) cluster radius, r 2) minimum number of data samples within the radius, k	1) the value of the knee point of the sorted k-dist graph 2) $k = 4$ [6]
APC	1) maximum number of iterative refinements 2) termination tolerance 3) dampening factor	predefined as in [105]
eClustering	1) initial radius, r 2) learning parameter, ρ	1) $r = 0.5$ 2) $\rho = 0.5$ [11]
ELMC	initial cluster radius, r	$r = 0.15$ [12]
NMMBC	1) prior scaling parameter 2) kappa coefficient	predefined as in [122]
NMIBC	grid size	predefined as in [226]
CEDS	1) microCluster radius, r 2) decay factor, ω 3) min microCluster threshold, φ	1) $r = 0.15$ 2) $\omega = 500$ 3) $\varphi = 1$ [118]

The clustering algorithms that require the number of clusters to be known in advance, i.e. k-means [5], online k-means [227], fuzzy c-means [124], random swap [228] algorithms, etc., or require the problem-specific thresholds, i.e. hierarchical clustering algorithm [99], are not included in the comparison if no specific declaration.

The quality of the clustering/partitioning results is evaluated based on the following five indicators:

1) Number of clusters/clouds (NC). Ideally, NC should be as close as possible to the number of actual classes (ground truth) in the dataset. However, this would mean one cluster/ data cloud per class and is only the best solution if each class has a very simple (circular) hyper-spherical pattern. However, this is not the case in the vast majority of the real problems. In most of the cases, data samples from different classes are mixed with each other. The best way to cluster/partition the dataset of this type is to divide the data into smaller parts (i.e. more than one cluster per class) to achieve a better separation. At the same time, having too many clusters per class is also reducing the generalization capability (leading to overfitting) and the interpretability. Therefore, in this thesis, the reasonable value range of NC is considered as $Number\ of\ Classes \leq \mathrm{NC} \leq 10\% \cdot Number\ of\ data\ samples$. If NC is smaller than the number of actual classes in the dataset or is more than 10% of all data samples, the clustering result is considered as an invalid one. The former case indicates that there are too many clusters generated by the clustering algorithm, which makes the information too trivial for users, and the latter case indicates that the clustering algorithm fails to separate the data samples from different classes.

2) Purity (PU) [20], which is calculated based on the result and the ground truth:

$$\mathrm{PU} = \sum_{i=1}^{\mathrm{NC}} S_{i,D}/K, \tag{6.1}$$

where $S_{i,D}$ is the number of data samples with the dominant class label in the i^{th} cluster. Purity directly indicates separation ability of the clustering algorithm. The higher purity a clustering result has, the stronger separation ability the clustering algorithm exhibits.

3) Calinski-Harabasz index (CH) [13], the higher the Calinski-Harabasz index is, the better the clustering result is;

4) Davies-Bouldin index (DB) [17], the lower Davies-Bouldin index is, the better the clustering result is.

5) Time (t_{exe}): the execution time (in seconds) should be as small as possible.

6.1.1. Autonomous Data-Driven Clustering Algorithm

6.1.1.1. Benchmark Problems for Evaluation

The ADD clustering algorithm is more effective in grouping data into clusters with regular shapes and clear boundaries. Therefore, in this section, the following datasets are used for evaluation:

1) **Iris** dataset [229];

2) **A1** dataset [230];

3) **A2** dataset [230];

4) **S1** dataset [231];

5) **S2** dataset [231].

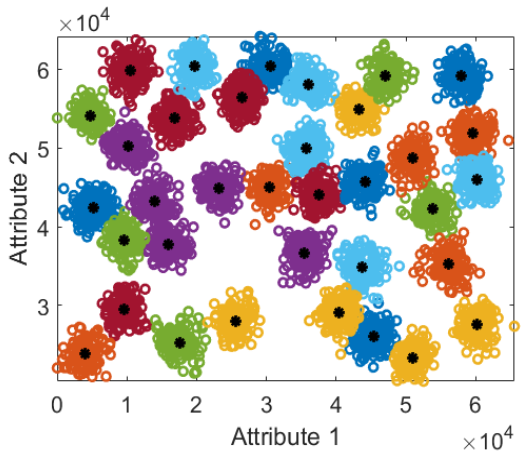
The details of the datasets are tabulated in Table 4.

Table 4. Details of benchmark datasets for evaluating ADD algorithm

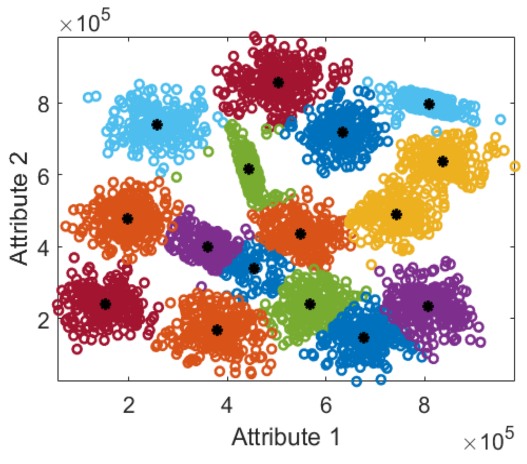
Dataset	Number of Classes	Number of Features	Number of Samples
Iris	3	4	150
A1	20	2	3000
A2	35	2	5250
S1	15	2	5000
S2	15	2	5000

6.1.1.2. Performance Evaluation and Discussion

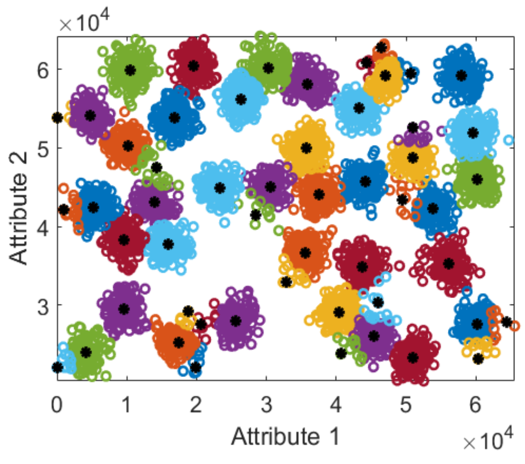
In this subsection, the performance of the ADD clustering algorithm is evaluated. For clarity, only the clustering results of the ADD clustering algorithm on A2 and S2 datasets are visualised in Figure 43, where the circles "o" in different colours denote data samples of different clusters, the black asterisks "*" denote the focal points/prototypes. In these experiments, the parallel computing ADD clustering algorithm uses five processors and each chunk has the size of 250 samples.



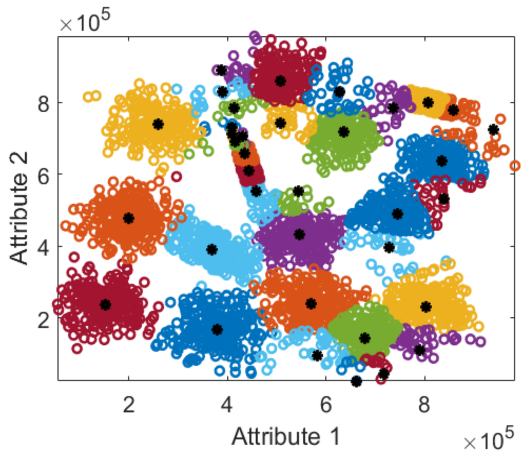
(a) A2-Offline ADD



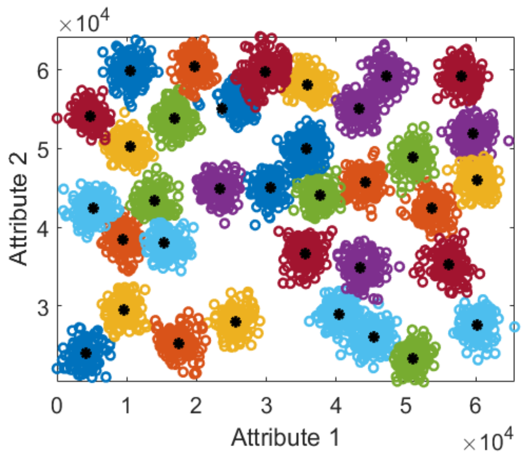
(b) S2-Offline ADD



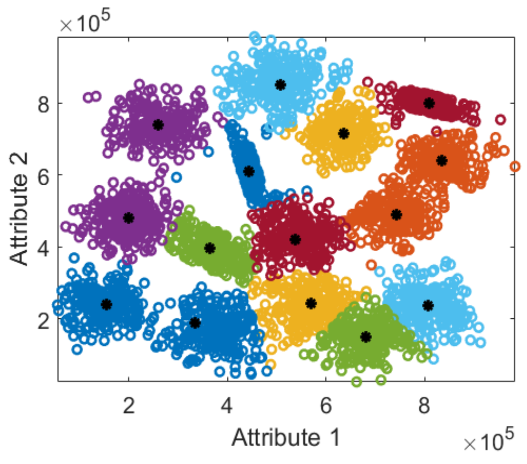
(c) A2-Evolving ADD



(d) S2-Evolving ADD



(e) A2-Parallel computing ADD



(f) S2-Parallel computing ADD

Figure 43. Clustering results of the ADD algorithm on A2 and S2 datasets.

One can see from Figure 43 that, the ADD clustering algorithm (all three versions) can effectively separate the data from different clusters. Due to the nature of the streaming data clustering on the sample-by-sample basis, the evolving ADD algorithm produces more clusters than the others.

For the parallel computing ADD algorithm, the influences of the number of processors and chunk size on the computational efficiency of the algorithm are studied based on the S1 dataset, where the number of processors varies from 2 to 10 and the chunk size varies from 100 to 400. The time consumption of the two stages of the clustering process with different experimental settings is tabulated in Table 5, where the amount of the time consumption during stage 1 as tabulated in this table is the average processing time per processor.

Table 5. Computational efficiency study under different experimental setting

Chunk Size	Stage	Number of Processors								
		2	3	4	5	6	7	8	9	10
100	1	0.51	0.17	0.11	0.09	0.07	0.06	0.05	0.04	0.03
	2	1.04	0.53	0.50	0.44	0.41	0.40	0.37	0.37	0.35
150	1	0.31	0.17	0.12	0.08	0.06	0.05	0.04	0.04	0.03
	2	0.61	0.53	0.47	0.42	0.39	0.38	0.35	0.35	0.34
200	1	0.30	0.16	0.11	0.08	0.06	0.05	0.04	0.04	0.03
	2	0.61	0.48	0.46	0.41	0.39	0.37	0.37	0.37	0.35
250	1	0.31	0.16	0.11	0.08	0.06	0.05	0.04	0.04	0.03
	2	0.66	0.51	0.44	0.39	0.39	0.36	0.35	0.34	0.33
300	1	0.27	0.15	0.10	0.07	0.06	0.05	0.04	0.04	0.03
	2	0.65	0.46	0.11	0.38	0.36	0.35	0.34	0.33	0.33
350	1	0.27	0.15	0.10	0.07	0.06	0.05	0.04	0.03	0.03
	2	0.60	0.48	0.39	0.36	0.35	0.34	0.34	0.34	0.33
400	1	0.25	0.14	0.09	0.07	0.06	0.05	0.04	0.03	0.03
	2	0.53	0.44	0.40	0.39	0.37	0.35	0.35	0.33	0.32

From Table 5 one can see that, in general, the more processors are used in the experiments, the more efficient the clustering process will be. Meanwhile, a larger chunk size can accelerate the clustering process.

The quality of the clustering results obtained by the ADD algorithm on the five benchmark datasets are evaluated in Table 6, Table 7 and Table 8, where for the parallel computing ADD algorithm, its performance is evaluated on the A1, A2, S1 and S2 datasets and it uses five processors with the chunk size of 250 samples. The performance comparison is also conducted in Table 6, Table 7 and Table 8 by using the 11 state-of-the-art approaches tabulated in Table 3.

In Table 6, Table 7 and Table 8, k-means [5], fuzzy c-means (FCM) [124] and random swap (RS) [228] algorithms are also involved in the comparison for a better evaluation. For these three algorithms, the number of clusters is set to be the same as the number of classes.

Table 6. Performance evaluation and comparison of the ADD algorithm

Dataset	Algorithm	NC	PU	CH	DB	t_{exe}	Validity
Iris	**OADD** [a]	**3**	**0.8933**	**560.3999**	**0.6623**	**0.33**	**Yes**
	EADD [b]	**9**	**0.8933**	**262.1338**	**1.0682**	**0.10**	**Yes**
	MSC	24	0.9733	205.448	0.8806	0.04	No
	SUBC	9	0.9400	195.4512	1.1244	0.11	Yes
	SOM	144	1.0000	260.9743	0.3180	2.37	No
	DPC	2	0.6667	501.9249	0.3836	1.91	No
	DBSCAN	2	0.6267	613.9620	0.3530	0.01	No
	APC	5	0.91333	440.6378	0.9267	0.11	Yes
	eClustering	4	0.6733	58.7119	1.4130	0.03	Yes
	ELMC	1	0.3333	NaN [c]	NaN	0.13	No
	NMMBC	1	0.3333	NaN	NaN	4.27	No
	NMIBC	3	0.8400	484.8990	**0.6338**	0.33	Yes
	CEDS	16	0.9533	168.2046	1.5277	0.66	No
	kmeans	3	0.8867	560.3660	0.6664	0.10	Yes
	FCM	3	0.8933	559.0000	0.6696	0.03	Yes
	RS	3	0.7200	42.0557	2.2776	0.45	Yes
A1	**OADD**	**20**	**0.9833**	**13829.3761**	**0.5267**	**30.94**	**Yes**
	EADD	**27**	**0.9827**	**10663.5118**	**0.7232**	**1.41**	**Yes**
	PCADD [d]	20	0.9833	13751.8751	0.5289	0.41	Yes
	MSC	9	0.4453	4009.6177	0.8612	0.04	No
	SUBC	9	0.4480	4307.8855	0.6763	1.15	No
	SOM	144	0.9703	9597.3680	0.8177	6.65	Yes
	DPC	9	0.4497	5361.9764	0.7468	1.83	No
	DBSCAN	25	0.8197	8627.927	0.5590	0.73	Yes
	APC	1401	0.9107	100.372	1.309	50.05	No
	eClustering	3	0.1500	1392.1482	0.7243	0.12	No
	ELMC	2	0.1000	2178.4792	0.8370	0.55	No
	NMMBC	4	0.1997	2844.1017	21.5226	198.03	No
	NMIBC	7	0.3500	4241.8797	0.8011	5.28	No
	CEDS	21	0.3846	251.8649	1.2713	8.03	Yes
	kmeans	20	0.8763	9070.4058	0.6962	0.12	Yes
	FCM	20	0.9837	13826.785	0.52664	0.21	Yes
	RS	20	0.4137	50.8635	28.2870	1.82	Yes

[a] Offline ADD; [b] Evolving ADD; [c] Not a number; [d] Parallel computing ADD.

Table 7. Performance evaluation and comparison of the ADD algorithm (continue - part 1)

Dataset	Algorithm	NC	PU	CH	DB	t_{exe}	Validity
A2	**OADD**	**35**	**0.9825**	**18205.8529**	**0.5241**	**123.81**	**Yes**
	EADD	**53**	**0.9796**	**12956.6360**	**0.7440**	**5.79**	**Yes**
	PCADD	35	0.9796	17966.9969	0.5278	**0.72**	Yes
	MSC	11	0.3139	5002.6452	0.8326	0.06	No
	SUBC	11	0.3143	6323.3781	0.7386	1.95	No
	SOM	144	0.9728	12749.7708	0.8833	11.83	Yes
	DPC	19	0.5423	7426.8078	0.6334	3.13	No
	DBSCAN	47	0.8187	11319.5382	0.6217	1.90	Yes
	APC	2844	0.8711	63.1582	1.0557	147.12	No
	eClustering	3	0.0857	2012.3325	0.8095	0.20	No
	ELMC	2	0.0571	2051.5427	1.1627	0.89	No
	NMMBC	7	0.1716	2193.3707	1.6057	67.89	No
	NMIBC	13	0.3714	6051.1654	0.7161	13.17	No
	CEDS	17	0.1897	756.4172	0.9958	11.19	No
	kmeans	35	0.8977	12963.9614	0.6631	0.13	Yes
	FCM	35	0.8697	10912.0552	0.7017	1.46	Yes
	RS	35	0.2916	42.5126	30.9873	5.96	Yes
S1	**OADD**	**16**	**0.9938**	**21624.5534**	**0.4199**	**94.73**	**Yes**
	EADD	**42**	**0.9856**	**11316.8467**	**0.7428**	**5.12**	**Yes**
	PCADD	15	0.9942	22670.4843	0.3661	0.47	Yes
	MSC	13	0.8132	10104.1419	0.5124	0.04	No
	SUBC	10	0.6732	8360.6375	0.5729	3.07	No
	SOM	144	0.9940	14901.7546	0.8055	9.53	Yes
	DPC	3	0.2100	2356.1843	0.8905	4.54	No
	DBSCAN	32	0.9146	1256.2090	1.2679	2.84	Yes
	APC	2297	0.9584	123.9501	0.8424	194.97	No
	eClustering	2	0.1400	1159.7451	1.8898	0.19	No
	ELMC	1	0.0700	NaN	NaN	1.11	No
	NMMBC	6	0.3952	2966.1273	0.6952	345.40	No
	NMIBC	6	0.4100	5922.5340	0.7305	10.38	No
	CEDS	21	0.6048	1285.7427	1.0851	11.85	Yes
	kmeans	15	0.9326	15172.8111	0.4822	0.12	Yes
	FCM	15	0.9938	22675.2540	0.3665	0.38	Yes
	RS	15	0.3448	85.7921	20.3050	2.26	Yes

Table 8. Performance evaluation and comparison of the ADD algorithm (continue - part 2)

Dataset	Algorithm	NC	PU	CH	DB	t_{exe}	Validity
S2	**OADD**	**16**	**0.9638**	**13031.0961**	**0.5157**	**99.38**	**Yes**
	EADD	**36**	**0.9522**	**6997.2126**	**0.8773**	**3.34**	**Yes**
	PCADD	15	0.9694	13411.7340	0.4694	0.64	Yes
	MSC	13	0.7164	6443.3562	0.7698	0.04	No
	SUBC	10	0.6642	6265.4817	0.6630	2.90	No
	SOM	144	0.9692	9452.3645	0.8283	9.62	Yes
	DPC	2	0.1400	1764.3864	1.4709	4.55	No
	DBSCAN	35	0.7788	686.9831	2.0510	2.76	Yes
	APC	1283	0.9168	229.2614	1.4833	205.14	No
	eClustering	2	0.1366	1330.5700	1.4359	0.20	No
	ELMC	1	0.0700	NaN	NaN	1.11	No
	NMMBC	10	0.4652	2207.0915	2.3866	407.10	No
	NMIBC	4	0.2786	3966.1442	0.8060	12.41	No
	CEDS	26	0.6580	1324.1078	0.9463	12.96	Yes
	kmeans	15	0.9028	9434.6526	0.5841	0.13	Yes
	FCM	15	0.9112	10447.5460	0.5881	0.52	Yes
	RS	15	0.3240	34.7955	33.0821	2.27	Yes

It is clearly shown in Table 6, Table 7 and Table 8 that the ADD clustering algorithm exhibits stronger performance on all the five benchmark datasets compared with the alternative algorithms. Specifically, the offline ADD clustering algorithm produces the best results, but its computational efficiency is not high in comparison with the other two versions. The parallel computing version is the most efficient one and it outperforms all other algorithms on S1 and S2 datasets. The evolving ADD algorithm is very efficient as well, but its performance is not as high as the other two versions.

6.1.2. Hypercube-based Data Partitioning Algorithm

6.1.2.1. Benchmark Problems for Evaluation

The HCDP algorithm is more effective in partitioning low-dimensional datasets with irregular shapes. Therefore, in this section, the following datasets are used for evaluation:

1) **Flame** dataset [127];

2) **Jain** dataset [232];

3) **Aggregation** dataset [233];

4) **Pathbased** dataset [234];

5) **Banknote** authentication dataset [235].

The details of the datasets are tabulated in Table 9.

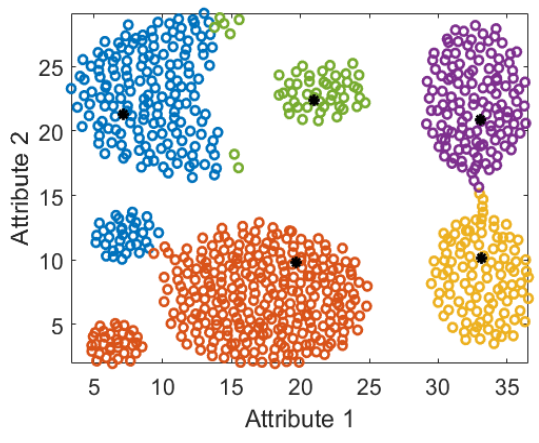
Table 9. Details of benchmark datasets for evaluating HCDP algorithm

Dataset	Number of Classes	Number of Features	Number of Samples
Flame	2	2	240
Jain	2	2	373
Aggregation	7	2	788
Pathbased	3	2	300
Banknote	2	4	1372

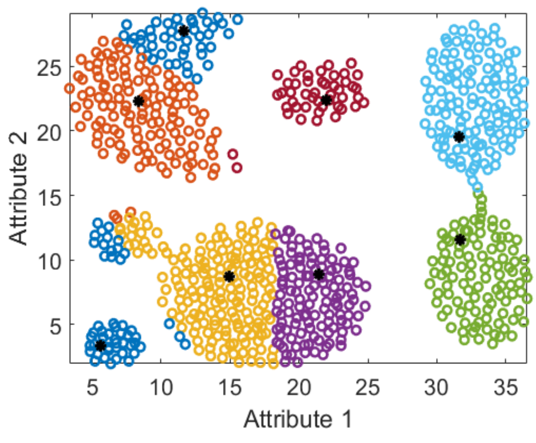
6.1.2.2. Performance Evaluation and Discussion

In this subsection, the performance of the HCDP algorithm is evaluated.

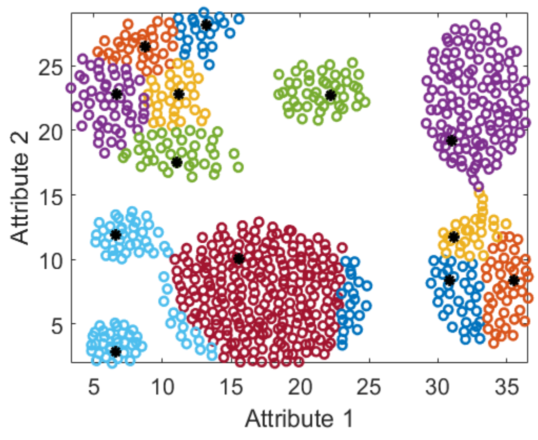
Firstly, the impact of the granularity, γ on the partitioning results is evaluated. For clarity, only the Aggregation dataset is used for this experiment. For offline HCDP algorithm, γ is set to be varied from 5 to 30, and the partitioning results are depicted in Figure 44. Similarly, for the evolving HCDP algorithm, γ is set to be varied from 10 to 35, and the partitioning results are depicted in Figure 45.



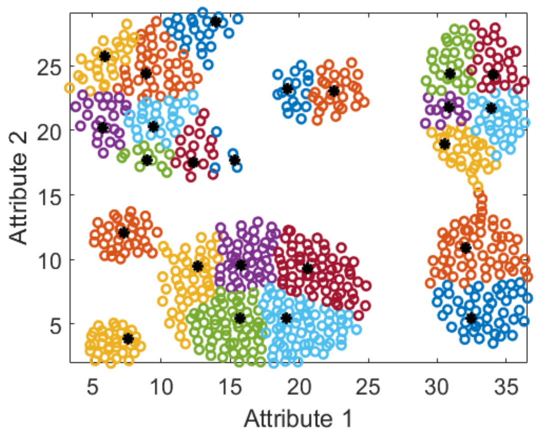
(a) $\gamma = 5$



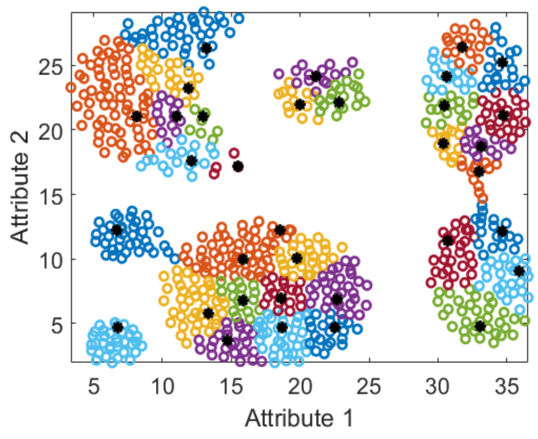
(b) $\gamma = 10$



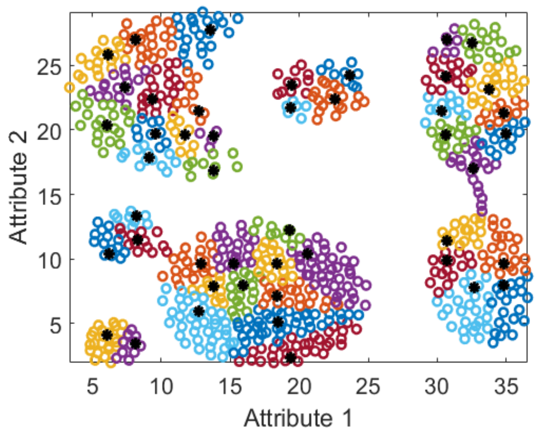
(c) $\gamma = 15$



(d) $\gamma = 20$

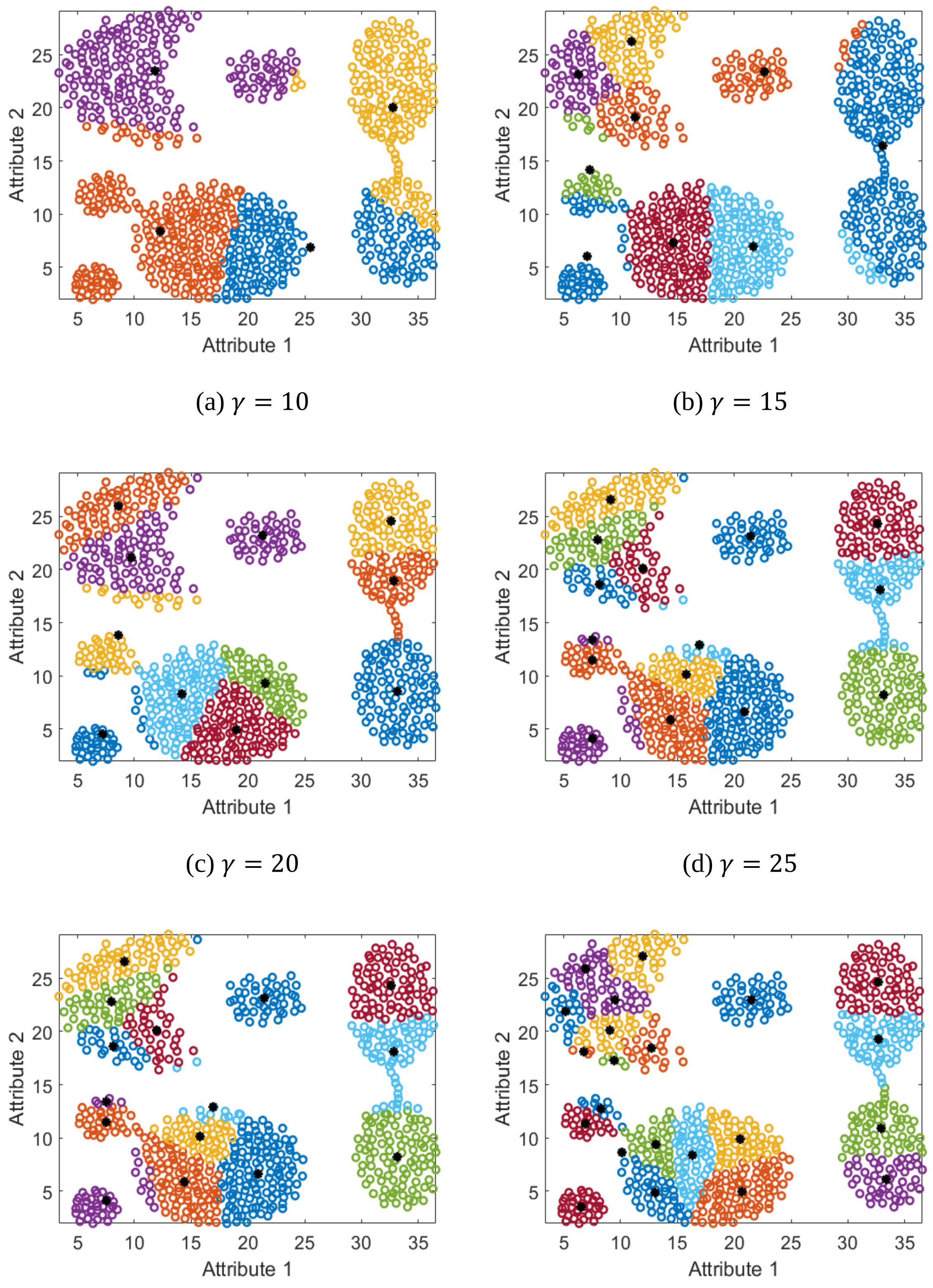


(e) $\gamma = 25$



(f) $\gamma = 30$

Figure 44. Partitioning results of the offline HCDP algorithm with different granularity.



(a) $\gamma = 10$ (b) $\gamma = 15$

(c) $\gamma = 20$ (d) $\gamma = 25$

(e) $\gamma = 30$ (f) $\gamma = 35$

Figure 45. Partitioning results of the evolving HCDP algorithm with different granularity.

One can see from Figure 44 and Figure 45 that, the higher granularity is chosen, the more data clouds are obtained from the dataset, which results in a more detailed partitioning results and vice versa. Therefore, by changing the granularity, one can partition the data with the HCDP algorithm in a more flexible, straightforward way to meet different purposes.

Table 10. Performance evaluation and comparison of the HCDP algorithm

Dataset	Algorithm	NC	PU	CH	DB	t_{exe}	Validity
Flame	**OHCDP** [a]	**20**	**0.9833**	**183.4588**	**0.8912**	**0.21**	**Yes**
	EHCDP [b]	**7**	**0.9792**	**221.5211**	**0.8467**	**0.13**	**Yes**
	MSC	9	0.9667	144.2422	0.8593	0.02	Yes
	SUBC	10	0.9708	215.6852	0.8728	0.13	Yes
	DPC	2	0.7875	133.6151	1.1338	1.31	Yes
	DBSCAN	1	0.6375	NaN	NaN	0.03	No
	APC	11	0.9875	244.4211	0.8714	0.08	Yes
	eClustering	4	0.7458	64.3348	1.1252	0.04	Yes
	ELMC	1	0.6375	NaN	NaN	0.10	No
	NMMBC	3	0.9583	161.6478	0.9199	11.56	Yes
	NMIBC	11	0.9917	126.7075	0.9187	0.85	Yes
	CEDS	73	0.9875	153.002	0.8228	1.34	Yes
Jain	**OHCDP**	**22**	**0.9893**	**713.0926**	**0.7170**	**0.26**	**Yes**
	EHCDP	**6**	**0.8928**	**387.0134**	**0.9215**	**0.12**	**Yes**
	MSC	9	0.9491	503.9343	0.7229	0.03	Yes
	SUBC	6	0.9625	595.4852	0.6817	0.12	Yes
	DPC	2	0.8606	468.0620	0.8001	1.99	Yes
	DBSCAN	4	0.7775	219.4335	0.6104	0.02	Yes
	APC	12	1.0000	927.1470	0.7039	0.29	Yes
	eClustering	3	0.7802	27.9497	1.2612	0.04	Yes
	ELMC	1	0.7400	NaN	NaN	0.10	No
	NMMBC	4	0.9652	331.2766	0.7400	15.67	Yes
	NMIBC	8	1.0000	636.1075	0.6751	1.18	Yes
	CEDS	63	1.0000	880.0854	0.8231	1.69	Yes
Aggregation	**OHCDP**	**24**	**0.9911**	**1353.5003**	**0.8873**	**0.52**	**Yes**
	EHCDP	**15**	**0.9683**	**1112.8078**	**0.9829**	**0.20**	**Yes**
	MSC	10	0.9886	1297.9986	0.7401	0.03	Yes
	SUBC	8	0.9315	1203.3351	0.6932	0.39	Yes
	DPC	4	0.7703	753.8644	0.6519	1.57	No
	DBSCAN	6	0.8591	727.9153	0.5780	0.05	No
	APC	24	0.9949	1618.8342	0.8158	2.09	Yes
	eClustering	4	0.5393	117.1214	0.9280	0.05	No
	ELMC	2	0.3465	223.8349	1.1456	0.19	No
	NMMBC	4	0.7234	434.3226	0.6965	33.93	No
	NMIBC	8	0.9975	1251.0015	0.6767	4.83	Yes
	CEDS	81	0.9327	244.2537	0.8979	3.05	Yes

[a] Offline HCDP; [b] Evolving HCDP.

Table 11. Performance evaluation and comparison of the HCDP algorithm (continue)

Dataset	Algorithm	NC	PU	CH	DB	t_{exe}	Validity
Pathbased	**OHCDP**	**15**	**0.9533**	**320.9985**	**0.7688**	**0.20**	**Yes**
	EHCDP	**14**	**0.9067**	**266.0269**	**0.8399**	**0.14**	**Yes**
	MSC	9	0.8300	191.2394	0.5955	0.03	Yes
	SUBC	8	0.9167	311.6385	0.7755	0.11	Yes
	DPC	3	0.7333	355.5116	0.6312	1.22	Yes
	DBSCAN	2	0.6767	66.4760	1.3332	0.01	No
	APC	16	0.9567	394.1452	0.7246	0.13	Yes
	eClustering	3	0.5433	88.0246	1.4032	0.03	Yes
	ELMC	1	0.3667	NaN	NaN	0.09	No
	NMMBC	4	0.6933	187.0646	1.6354	15.49	Yes
	NMIBC	10	0.8200	150.3195	0.5302	1.04	Yes
	CEDS	79	0.9967	529.0332	0.6764	1.68	No
Banknote	**OHCDP**	**79**	**0.9920**	**811.5949**	**0.9144**	**3.10**	**Yes**
	EHCDP	**107**	**0.9927**	**929.2889**	**0.9868**	**4.16**	**Yes**
	MSC	24	0.9927	717.8110	0.7831	0.05	Yes
	SUBC	14	0.9657	743.6576	1.0000	0.67	Yes
	DPC	3	0.7413	1039.7004	0.9687	1.20	Yes
	DBSCAN	48	0.9402	352.5907	0.7607	0.17	Yes
	APC	30	0.9883	1114.1723	0.9229	4.54	Yes
	eClustering	1	0.5554	NaN	NaN	0.06	No
	ELMC	4	0.6778	408.0301	0.8317	0.33	Yes
	NMMBC	4	0.8462	690.5961	1.2327	62.77	Yes
	NMIBC	20	0.9913	690.5714	0.7206	5.24	Yes
	CEDS	120	0.8848	163.6131	1.4338	5.46	Yes

The quality of the partitioning results obtained by the HCDP algorithm on the five benchmark datasets are evaluated in Table 10 and Table 11, where $\gamma = 20$ for the offline version and $\gamma = 25$ for the evolving version. The performance comparison is also conducted in Table 10 and Table 11 by using the 10 state-of-the-art approaches (SOM algorithm is not used here) tabulated in Table 3.

From Table 10 and Table 11 one can see that the HCDP algorithm exhibits very good performance on all the five benchmark datasets comparable with the best performed state-of-the-art algorithms. Moreover, compared with other approaches, the operating mechanism of the HCDP algorithm is simpler and more straightforward.

6.1.3. Autonomous Data Partitioning Algorithm

6.1.3.1. Benchmark Problems for Evaluation

The ADP algorithm is very effective in partitioning large-scale, high-dimensional, complex datasets. Therefore, in this section, the following datasets are used for evaluation:

1) **Cardio**tocography dataset [236];

2) **Pen-based** handwritten digits recognition dataset [237];

3) **Occupancy** detection dataset [238];

4) **MAGIC** gamma telescope dataset [239];

5) **Letter** recognition dataset [240].

The details of the datasets are tabulated in Table 12.

Table 12. Details of benchmark datasets for evaluating ADP algorithm

Dataset	Number of Classes	Number of Features	Number of Samples
Cardio	3	22	2126
Pen-Based	10	16	10992
Occupancy [a]	2	5	8143 (training set)
			2665 (testing set 1)
			9752 (testing set 2)
MAGIC	2	10	19020
Letter	26	16	20000

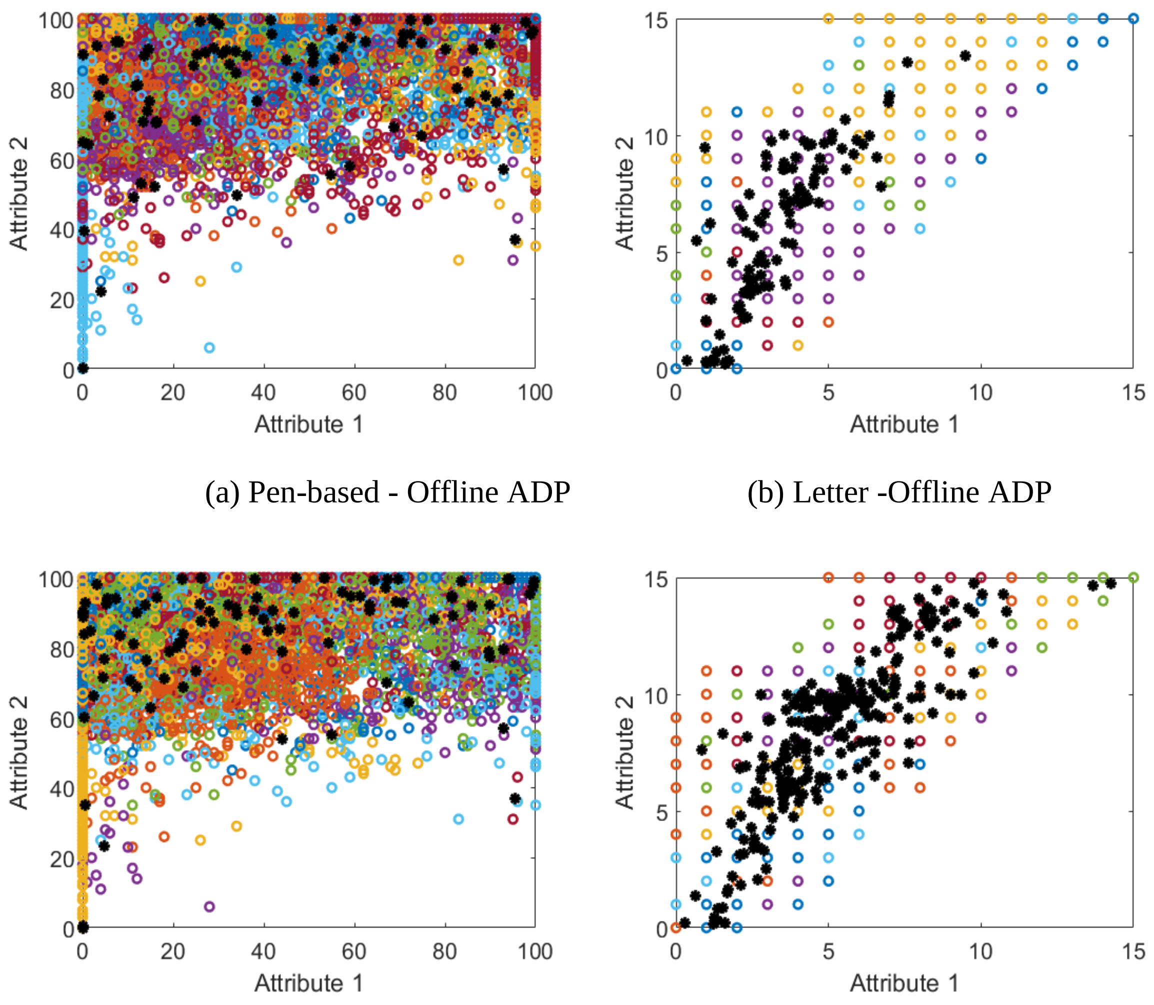
[a] The time stamps in the original dataset have been removed.

6.1.3.2. Performance Evaluation and Discussion

In this subsection, the performance of the ADP algorithm is evaluated. For clarity, only the partitioning results of the ADP algorithm on Pen-based handwritten digits recognition dataset and Letter recognition dataset are visualised in Figure 46.

One can see from Figure 46 that the ADP algorithm identified a number of prototypes from the observed data samples and partitioned the datasets into shape-free data clouds with the prototypes naturally by attracting data samples around them resembling Voronoi tessellation [64]. However, as there is no clear separation between data samples of different classes in the high-dimensional, large-scale datasets, it is very hard to directly evaluate the quality of the partitioning results.

Therefore, the quality indexes of the partitioning results obtained by the ADP algorithm as tabulated in Table 13 and Table 14 are used for further evaluation. The performance comparison is also conducted by using the 11 state-of-the-art approaches tabulated in Table 3.



(a) Pen-based - Offline ADP

(b) Letter -Offline ADP

(c) Pen-based - Evolving ADP

(d) Letter - Evolving ADP

Figure 46. Partitioning results of the ADP algorithm on Pen-based handwritten digits recognition dataset and Letter recognition dataset.

Table 13 and Table 14 clearly show that the ADP algorithm outperforms all other comparative algorithms in both the partitioning quality and computational efficiency. Moreover, the ADP algorithm is nonparametric and free from prior assumptions and user- and problem-specific parameters.

Table 13. Performance evaluation and comparison of the ADP algorithm

Dataset	Algorithm	NC	PU	CH	DB	t_{exe}	Validity
Cardio	**OADP [a]**	**81**	**0.8580**	**262.8859**	**1.0962**	**0.35**	**Yes**
	EADP [b]	**50**	**0.8561**	**315.6587**	**1.2816**	**0.36**	**Yes**
	MSC	1597	0.9958	189.5452	0.4150	2.61	No
	SUBC	254	0.9147	140.7584	1.3239	0.65	No
	SOM	144	0.8932	225.8973	1.1998	12.63	Yes
	DPC	3	0.7813	63.5735	0.5081	2.71	Yes
	DBSCAN	13	0.8053	35.8486	1.5204	0.43	Yes
	APC	43	0.8627	371.6572	1.3036	6.76	Yes
	eClustering	8	0.7949	269.3139	2.3566	0.24	Yes
	ELMC	2	0.7992	213.0737	1.4233	1.10	No
	NMMBC	4	0.7794	204.2315	1.0717	111.18	Yes
	NMIBC	328	0.9008	63.5207	0.6740	31.28	Yes
	CEDS	14	0.8015	152.9364	3.3800	23.61	Yes
Pen-Based	**OADP**	**79**	**0.9326**	**1057.9771**	**1.3264**	**4.45**	**Yes**
	EADP	**92**	**0.9342**	**967.5478**	**1.4241**	**1.53**	**Yes**
	MSC	8501	0.9999	154.0923	0.3652	169.14	No
	SUBC	187	0.8454	382.6055	1.9995	100.09	Yes
	SOM	144	0.9725	868.7807	1.4174	42.12	Yes
	DPC	7	0.5993	2559.6071	1.3044	17.32	No
	DBSCAN	38	0.6209	312.9177	1.4997	16.11	Yes
	APC	System Crashed					No
	eClustering	7	0.4394	1850.2452	2.094	1.49	No
	ELMC	9	0.3092	634.1555	2.1794	20.67	No
	NMMBC	41	0.9325	1010.81	2.2504	3727.38	Yes
	NMIBC	4316	0.9968	46.6194	0.4969	2187.06	No
	CEDS	1	0.1041	NaN	NaN	2466.47	No
Occupancy	**OADP**	**15**	**0.9783**	**34653.4935**	**0.6027**	**12.79**	**Yes**
	EADP	**131**	**0.9869**	**21530.3617**	**0.8165**	**2.65**	**Yes**
	MSC	37	0.9772	5710.9905	2.3532	0.39	Yes
	SUBC	9	0.9498	19878.6811	1.1872	30.81	Yes
	SOM	144	0.9895	52050.4029	0.7450	41.58	Yes
	DPC	2	0.7690	5495.9202	0.5548	30.49	Yes
	DBSCAN	208	0.8514	134.4039	1.4789	190.85	Yes
	APC	System Crashed					No
	eClustering	32	0.9178	3830.0232	1.0221	2.07	Yes
	ELMC	1	0.7689	NaN	NaN	3.18	No
	NMMBC	3	0.7691	4420.7364	0.5037	1062.11	Yes
	NMIBC	15	0.9761	10922.5114	0.3310	372.08	Yes
	CEDS	13	0.8484	1,555.1093	3.3988	42.22	Yes

[a] Offline ADP; [b] Evolving ADP.

Table 14. Performance evaluation and comparison of the ADP algorithm (continue)

Dataset	Algorithm	NC	PU	CH	DB	t_{exe}	Validity
MAGIC	**OADP**	**47**	**0.7289**	**1430.4657**	**1.3074**	**13.71**	**Yes**
	EADP	**380**	**0.7899**	**643.6832**	**1.3068**	**2.65**	**Yes**
	MSC	1469	0.7871	14.0702	0.7969	46.61	Yes
	SUBC	8	0.7145	5097.7399	1.3833	48.89	Yes
	SOM	144	0.7804	1238.2763	1.4087	59.21	Yes
	DPC	1	0.6483	NaN	NaN	36.98	No
	DBSCAN	15	0.6247	17.8876	0.8998	33.44	Yes
	APC	System Crashed					No
	eClustering	6	0.6484	2316.0290	2.2985	2.61	Yes
	ELMC	25	0.7381	548.5010	1.5284	25.06	Yes
	NMMBC	3	0.7345	1990.4386	1.8764	1560.68	Yes
	NMIBC	1578	0.7459	19.4133	0.4046	6833.96	Yes
	CEDS	54	0.7902	406.1227	4.4493	5999.13	Yes
Letter	**OADP**	**235**	**0.6000**	**433.4874**	**1.4023**	**15.19**	**Yes**
	EADP	**242**	**0.5825**	**414.5848**	**1.4839**	**2.92**	**Yes**
	MSC	7619	0.9760	61.9156	0.5979	256.07	No
	SUBC	153	0.5820	470.8426	1.5301	175.37	Yes
	SOM	144	0.5839	614.3846	1.5347	77.12	Yes
	DPC	3	0.0573	515.3193	1.1643	45.69	No
	DBSCAN	51	0.1584	94.7283	1.1522	33.85	Yes
	APC	System Crashed					No
	eClustering	5	0.1135	1590.4090	2.3557	4.23	No
	ELMC	9	0.1091	585.5138	1.6504	15.38	No
	NMMBC	46	0.4304	453.1501	2.5435	6316.60	Yes
	NMIBC	14526	0.9975	72.2380	0.3169	6279.86	No
	CEDS	43	0.2580	569.9774	2.0097	14865.00	Yes

6.1.4. Self-Organising Direction-Aware Data Partitioning Algorithm

6.1.4.1. Benchmark Problems for Evaluation

The SODA algorithm is very effective in partitioning very high-dimensional datasets. Therefore, in this section, the following datasets are used for evaluation:

1) **Wine** dataset [241]

2) **Steel** plates faults dataset [242];

3) **Dim1024** dataset [243];

4) **Dim15** dataset [243];

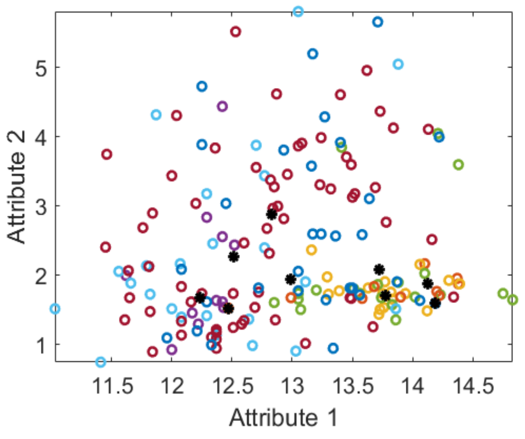
5) **Multiple** features dataset [244].

The details of the datasets are tabulated in Table 15.

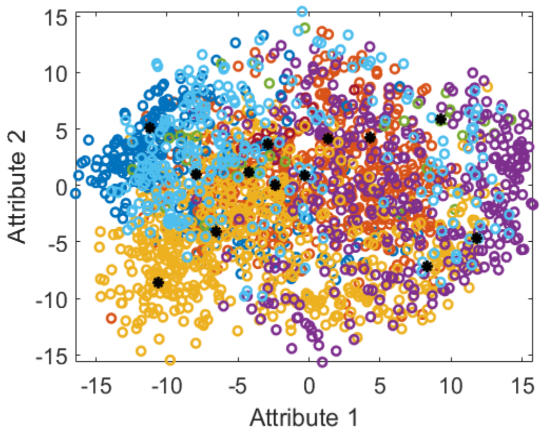
Table 15. Details of benchmark datasets for evaluating SODA algorithm

Dataset	Number of Classes	Number of Features	Number of Samples
Wine	3	13	178
Steel	7	27	1941
Dim1024	16	1024	1024
Dim15	9	15	10125
Multiple	10	649	2000

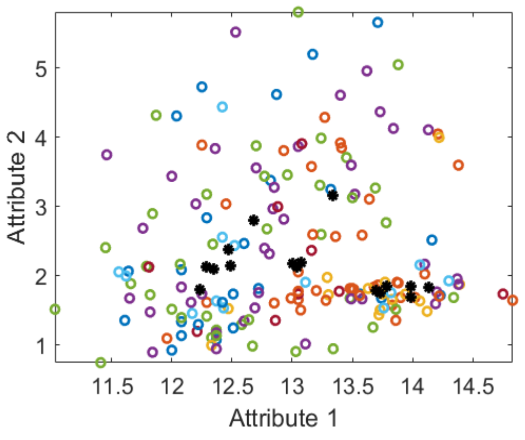
6.1.4.2. Performance Evaluation and Discussion



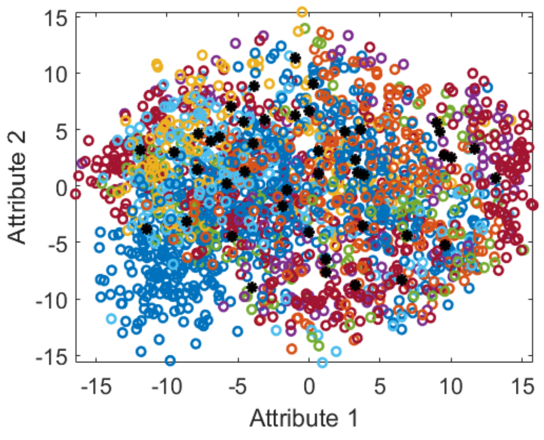
(a) Wine - Offline SODA



(b) Multiple features -Offline SODA



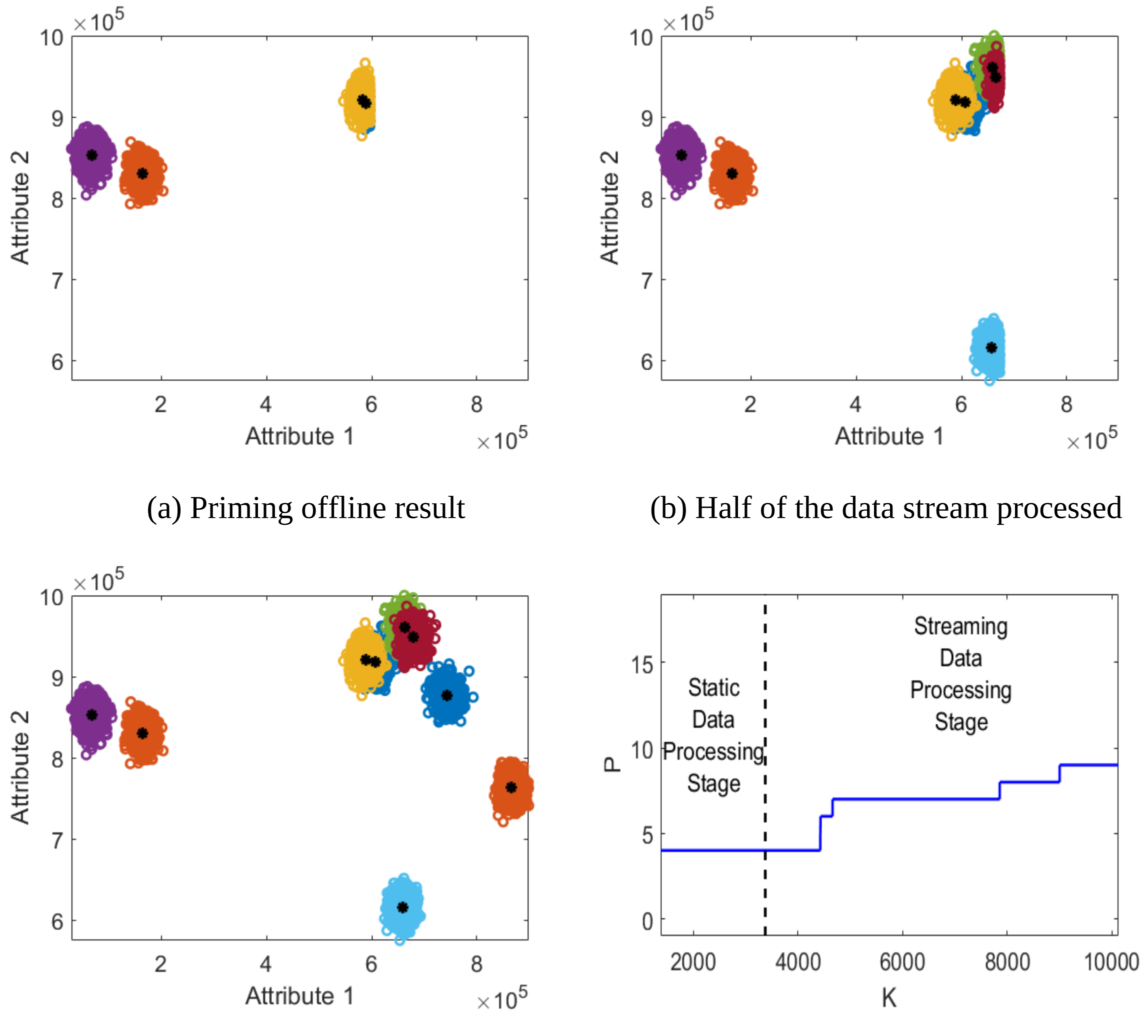
(c) Wine - Evolving SODA



(d) Multiple features - Evolving SODA

Figure 47. Partitioning results of the SODA algorithm on Wine dataset and Multiple features dataset.

In this subsection, the performance of the SODA algorithm is evaluated. The data partitioning results of the offline and evolving versions of the SODA algorithm based on the Wine and Multiple features datasets are depicted in Figure 47, where one can see that, the offline algorithm successfully identified a number of prototypes from the observed data samples and partitioned the datasets into shape-free data clouds.



(a) Priming offline result (b) Half of the data stream processed

(c) Final result (d) The change of the number of the planes

Figure 48. The evaluation of the extension of the offline SODA algorithm for streaming data.

The Dim15 dataset is further used to demonstrate the performance of the streaming data processing extension of the offline SODA algorithm. In the following example, one third of the total data samples of the dataset are used as a static priming dataset for the offline SODA algorithm to generate the initial partitioning results. The rest of the data samples are transformed into data streams for the algorithm to continue to build upon the priming result. The overall result is presented in Figure 48, and the change of the number of direction-aware planes is also given.

Table 16. Performance evaluation and comparison of the SODA algorithm

Dataset	Algorithm	NC	PU	CH	DB	t_{exe}	Validity
Wine	**OSODA** [a]	**9**	**0.6966**	**400.2223**	**1.2734**	**0.83**	**Yes**
	ESODA [b]	**16**	**0.7135**	**525.4880**	**1.9610**	**0.26**	**Yes**
	MSC	178	1.0000	NaN	0.0000	0.07	No
	SUBC	178	1.0000	NaN	0.0000	1.76	No
	SOM	144	0.9382	3058.57	0.3756	3.46	No
	DPC	3	0.6461	321.3938	0.4782	2.49	Yes
	DBSCAN	4	0.6685	139.8891	1.9340	0.03	Yes
	APC	51	0.8090	45.9785	0.5056	0.56	No
	eClustering	15	0.9157	31.8471	4.6352	0.06	Yes
	ELMC	54	0.9719	7.6683	0.6812	0.70	No
	NMMBC	1	0.3988	NaN	NaN	9.50	No
	NMIBC	4	0.6517	228.9969	0.3712	0.61	Yes
	CEDS	178	1.0000	NaN	0.0000	0.08	No
Steel	**OSODA**	23	**0.5095**	**2219.4197**	**0.9323**	**1.21**	**Yes**
	ESODA	**23**	**0.5064**	**2784.0320**	**1.8149**	**1.62**	**Yes**
	MSC	1555	0.9948	24.7451	9.8535	2.92	No
	SUBC	4	0.3988	494.1967	0.9100	4.37	No
	SOM	144	0.5538	2016.5369	0.6329	9.83	No
	DPC	3	0.3478	1224.2338	0.4226	2.40	No
	DBSCAN	18	0.4858	57.8279	1.7112	0.51	Yes
	APC	1477	0.8563	6.9878	0.4486	33.37	No
	eClustering	16	0.4153	184.5048	1.9151	0.24	Yes
	ELMC	7	0.3730	84.1426	1.2951	2.88	Yes
	NMMBC	2	0.3472	21.9988	0.1474	96.48	No
	NMIBC	9	0.3653	690.3357	0.3034	69.05	Yes
	CEDS	2	0.3467	2.0546	18.6821	17.73	No
Dim1024	**OSODA**	**16**	**1.0000**	**718469.7967**	**0.0132**	**1.15**	**Yes**
	ESODA	**16**	**1.0000**	**718469.7967**	**0.0132**	**3.66**	**Yes**
	MSC	120	1.0000	126798.4888	0.4496	0.88	No
	SUBC	16	1.0000	718469.7967	0.0132	16.32	Yes
	SOM	144	1.0000	144252.3793	0.9370	159.77	No
	DPC	14	0.8750	529.5497	0.6965	3.26	No
	DBSCAN	16	0.8721	381.3919	0.9975	0.56	Yes
	APC	1024	1.0000	NaN	0.0000	1.34	No
	eClustering	8	0.3389	54.6740	2.7683	1.67	No
	ELMC	1	0.0625	NaN	NaN	0.49	No
	NMMBC	3	0.1875	69.6915	3.1523	11827.25	No
	NMIBC	1024	1.0000	NaN	0.0000	2080.58	No
	CEDS	8	0.5000	139.4129	1.4281	52.41	No

[a] Offline SODA; [b] Evolving SODA; [c] Infinity.

Table 17. Performance evaluation and comparison of the SODA algorithm (continue)

Dataset	Algorithm	NC	PU	CH	DB	t_{exe}	Validity
Dim15	**OSODA**	9	**1.0000**	**302436.3684**	**0.1177**	**2.99**	**Yes**
	ESODA	9	**1.0000**	**302436.3684**	**0.1177**	**13.18**	**Yes**
	MSC	9	1.0000	302436.3684	0.1177	0.04	Yes
	SUBC	9	1.0000	302436.3684	0.1177	25.15	Yes
	SOM	144	1.0000	26172.4720	2.2328	39.30	Yes
	DPC	4	0.4444	4533.2627	0.6696	13.65	No
	DBSCAN	9	0.9586	20602.057	1.2317	15.92	Yes
	APC	System Crashed					No
	eClustering	16	0.5680	1528.6342	2.3851	1.26	Yes
	ELMC	2	0.2222	3319.7039	0.6205	2.58	No
	NMMBC	4	0.4444	2412.1759	1.4420	649.05	No
	NMIBC	3	0.3333	4327.2420	0.5837	141.34	No
	CEDS	76	0.6126	289.8403	2.2719	874.35	Yes
Multiple	**OSODA**	**13**	**0.5860**	**1593.494**	**1.3216**	**6.89**	Yes
	ESODA	**44**	**0.7095**	**1103.4226**	**1.4703**	**7.06**	Yes
	MSC	1994	1.0000	Inf [c]	0.0000	16.72	No
	SUBC	1994	1.0000	Inf	0.0000	77.53	No
	SOM	144	0.9230	695.3067	1.4205	197.11	No
	DPC	6	0.5830	2307.1654	1.1992	3.85	No
	DBSCAN	4	0.1915	15.5707	2.2674	0.89	No
	APC	22	0.8025	2098.7458	1.4701	12.63	Yes
	eClustering	19	0.5195	200.5162	3.8372	4.25	Yes
	ELMC	1988	1.0000	7.0694	0.2836	99.50	No
	NMMBC	1	0.1000	NaN	NaN	5059.90	No
	NMIBC	2000	1.0000	NaN	0.0000	2446.46	No
	CEDS	2	0.1735	65.3322	4.0605	381.28	No

The quality indexes of the partitioning results obtained by the SODA algorithm (both offline and evolving versions) are tabulated in Table 16 and Table 17 for further evaluation. The performance comparison is also conducted by using the 11 state-of-the-art approaches tabulated in Table 3.

It is clearly shown in Table 16 and Table 17 that the SODA algorithm outperforms all other comparative algorithms in terms of partitioning quality on the five benchmark datasets. Moreover, its computational efficiency is also very high, and does not decrease with the increase of dimensionality.

6.2. Evaluation of the Supervised Learning Algorithms

The performance of the self-organising supervised machine learning algorithms presented in chapter 4, namely

1) Autonomous learning multi-model (ALMMO) system;

2) Zero order autonomous learning multi-model (ALMMO-0) classifier;

3) Self-organising fuzzy logic (SOFL) classifier;

4) Autonomous anomaly detection (AAD) algorithm;

are evaluated based on benchmark datasets. Similarly, during the experiments, it is assumed that there is no any prior knowledge about the datasets.

6.2.1. Autonomous Learning Multi-Model System

In this section, the performance of the ALMMO system is evaluated based on two types of problems, namely 1) regression and 2) classification.

6.2.1.1. Benchmark Problems for Evaluation

A. Regression

The first regression problem for the evaluation is the QuantQuote Second Resolution Market (QQSRM) database [245], which contains tick-by-tick data on all NASDAQ, NYSE, and AMEX securities from 1998 to the present moment in time. The frequency of tick data varies from one second to few minutes. This dataset contains 19144 data samples. In this thesis, the following five attributes, namely

1) Time, K;

2) Open price, $x_{K,1}$;

3) High price, $x_{K,2}$;

4) Low price, $x_{K,3}$;

5) Close price, $x_{K,4}$;

are used for the prediction of the future values of high price 8, 12, 16, 20 and 24 steps ahead, namely, $y_K = f(x_{K,1}, x_{K,2}, x_{K,3}, x_{K,4})$ and $y_K = x_{K+8,2}$, $y_K = x_{K+12,2}$, $y_K = x_{K+16,2}$, $y_K = x_{K+20,2}$ and $y_K = x_{K+24,2}$, respectively. The data samples are standardized online before prediction.

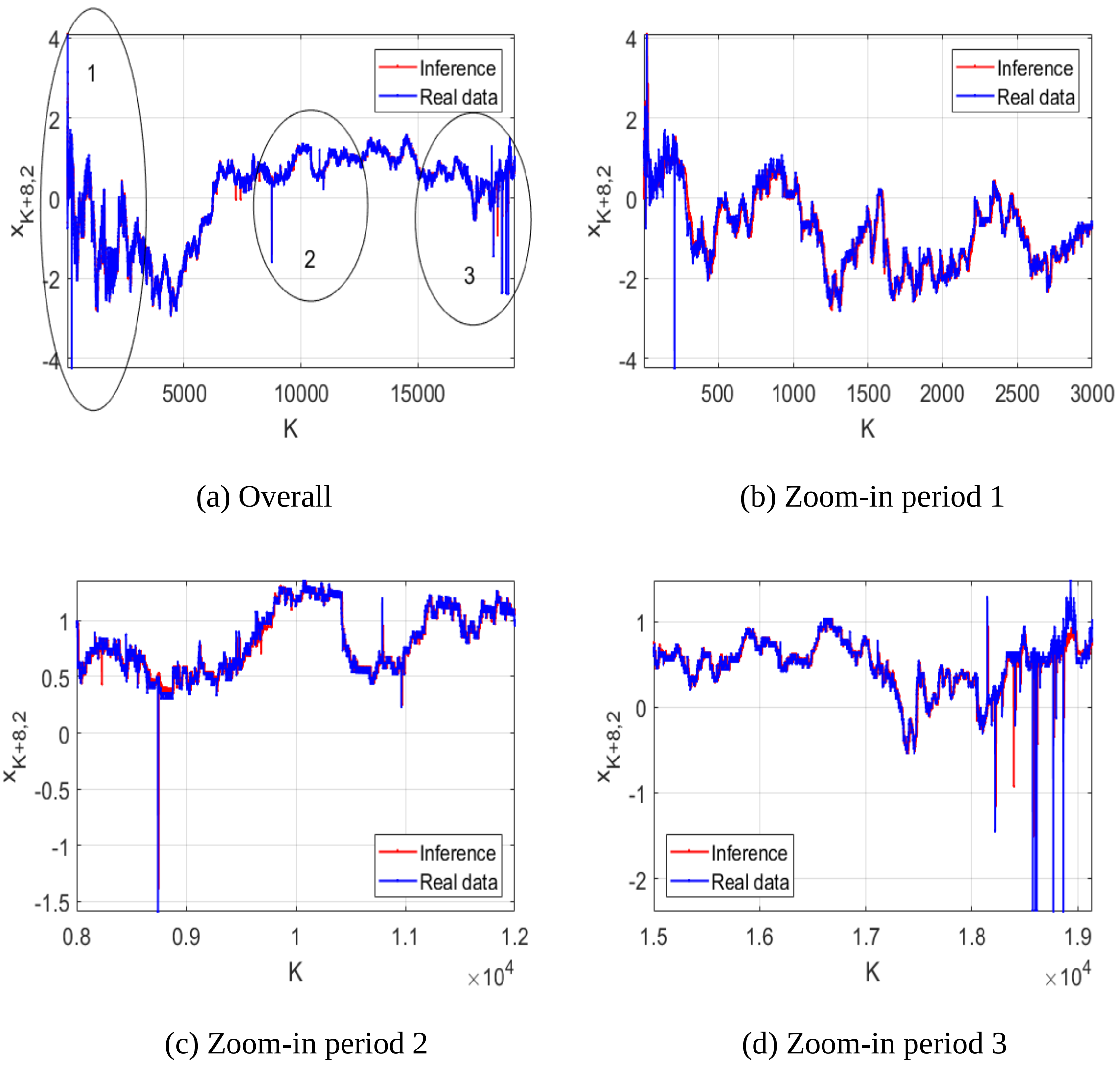
The second regression problem is based on a more frequently used real dataset, the Standard and Poor (S&P) index data [246]. This dataset contains 14893 data samples acquired from January 3, 1950 to March 12, 2009. Other prediction algorithms frequently use this dataset as a benchmark for performance because of the nonlinear, erratic and time-variant behaviour of the data. The input and output relationship of the system is governed by the following equation: $y_K = f(x_{K-4}, x_{K-3}, x_{K-2}, x_{K-1}, x_K)$ and $y_K = x_{K+1}$.

B. Classification

Two popular benchmark problems for binary classification, namely, PIMA [247] and occupancy detection [238] datasets, are used for evaluating the performance of the ALMMO system. PIMA dataset consists of 768 data samples, each of which has eight attributes and one label. The details of the occupancy detection dataset have been given in Table 11. The occupancy detection dataset contains one training set (8143 data samples) and two testing sets (2665 and 9752 data samples in each) [238].

6.2.1.2. Performance Evaluation and Discussion

A. Regression



(a) Overall (b) Zoom-in period 1

(c) Zoom-in period 2 (d) Zoom-in period 3

Figure 49. Prediction result for the QQSRM problem.

Firstly, the QQSRM dataset is considered. The current data sample, $\boldsymbol{x}_K = [x_{K,1}, x_{K,2}, x_{K,3}, x_{K,4}]^T$ is used to predict the High price 8 steps ahead $x_{K+8,2}$, namely, $x_{K+8,2} = f(x_{K,1}, x_{K,2}, x_{K,3}, x_{K,4})$. The overall prediction result is presented in Figure 49(a) and three zoom-in periods (circulated areas in Figure 49(a)) are depicted in Figure 49 (b)-(d). The evolution of number of data clouds/fuzzy rules is depicted in Figure 50. As one can see from Figure 49, there are many abnormal data samples and random fluctuations in the data stream. At the beginning and the end of this data stream, large fluctuations and abnormal data frequently appear, while in the middle, the data pattern changes relatively smoothly with only a small number of abnormal data. The corresponding changes of the system structure can also be seen in Figure 50. Thus, one can see that, the ALMMO system is capable to successfully follow the non-stationary data pattern and exhibits very accurate prediction results and demonstrates a strong evolving ability.

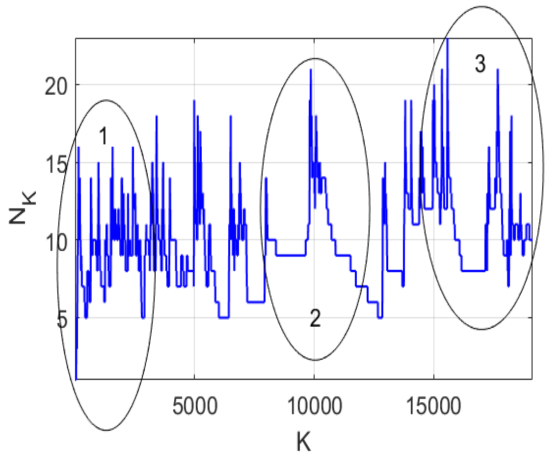


Figure 50. The evolution of number of data clouds/fuzzy rules.

The AnYa type fuzzy rules of the ALMMO system in the final time instance are presented in Table 18 as the illustrative examples.

Table 18. Example of fuzzy rules identified from the learning progress

Rule#	Detailed Expression
1	$IF \begin{bmatrix} Open\ Price, x_{K,1} \sim 1409050 \\ High\ Price, x_{K,2} \sim 1409073 \\ Low\ Price, x_{K,3} \sim 1409027 \\ Close\ Price, x_{K,4} \sim 1409053 \end{bmatrix} THEN \begin{pmatrix} y_K = 0.0187 \\ +0.0916x_{K,1} + 0.8096x_{K,2} \\ +0.2933x_{K,3} + 0.7554x_{K,4} \end{pmatrix}$
2	$IF \begin{bmatrix} Open\ Price, x_{K,1} \sim 1409639 \\ High\ Price, x_{K,2} \sim 1409659 \\ Low\ Price, x_{K,3} \sim 1409617 \\ Close\ Price, x_{K,4} \sim 1409648 \end{bmatrix} THEN \begin{pmatrix} y_K = 0.0505 \\ +0.6342x_{K,1} + 0.6156x_{K,2} \\ +0.4011x_{K,3} + 0.1668x_{K,4} \end{pmatrix}$
3	$IF \begin{bmatrix} Open\ Price, x_{K,1} \sim 1408586 \\ High\ Price, x_{K,2} \sim 1408595 \\ Low\ Price, x_{K,3} \sim 1408575 \\ Close\ Price, x_{K,4} \sim 1408582 \end{bmatrix} THEN \begin{pmatrix} y_K = 0.0426 \\ +0.6934x_{K,1} + 0.4999x_{K,2} \\ +0.4721x_{K,3} + 0.1119x_{K,4} \end{pmatrix}$
4	$IF \begin{bmatrix} Open\ Price, x_{K,1} \sim 1404596 \\ High\ Price, x_{K,2} \sim 1404596 \\ Low\ Price, x_{K,3} \sim 1404596 \\ Close\ Price, x_{K,4} \sim 1404596 \end{bmatrix} THEN \begin{pmatrix} y_K = 0.3734 \\ +0.1978x_{K,1} + 0.4681x_{K,2} \\ +0.0169x_{K,3} + 0.1065x_{K,4} \end{pmatrix}$
5	$IF \begin{bmatrix} Open\ Price, x_{K,1} \sim 1407657 \\ High\ Price, x_{K,2} \sim 1407656 \\ Low\ Price, x_{K,3} \sim 1407657 \\ Close\ Price, x_{K,4} \sim 1407656 \end{bmatrix} THEN \begin{pmatrix} y_K = 0.3960 \\ +0.1474x_{K,1} + 0.4800x_{K,2} \\ -0.0060x_{K,3} + 0.1204x_{K,4} \end{pmatrix}$
6	$IF \begin{bmatrix} Open\ Price, x_{K,1} \sim 1407442 \\ High\ Price, x_{K,2} \sim 1407442 \\ Low\ Price, x_{K,3} \sim 1407442 \\ Close\ Price, x_{K,4} \sim 1407442 \end{bmatrix} THEN \begin{pmatrix} y_K = 0.4063 \\ +0.1198x_{K,1} + 0.4500x_{K,2} \\ -0.0018x_{K,3} + 0.1031x_{K,4} \end{pmatrix}$
7	$IF \begin{bmatrix} Open\ Price, x_{K,1} \sim 1408443 \\ High\ Price, x_{K,2} \sim 1408443 \\ Low\ Price, x_{K,3} \sim 1402795 \\ Close\ Price, x_{K,4} \sim 1402795 \end{bmatrix} THEN \begin{pmatrix} y_K = 0.4692 \\ +0.1045x_{K,1} + 0.4195x_{K,2} \\ -0.0190x_{K,3} + 0.0837x_{K,4} \end{pmatrix}$
8	$IF \begin{bmatrix} Open\ Price, x_{K,1} \sim 1402769 \\ High\ Price, x_{K,2} \sim 1402768 \\ Low\ Price, x_{K,3} \sim 1402769 \\ Close\ Price, x_{K,4} \sim 1402769 \end{bmatrix} THEN \begin{pmatrix} y_K = 0.5284 \\ +0.1856x_{K,1} + 0.4638x_{K,2} \\ -0.1411x_{K,3} - 0.0306x_{K,4} \end{pmatrix}$
9	$IF \begin{bmatrix} Open\ Price, x_{K,1} \sim 1402744 \\ High\ Price, x_{K,2} \sim 1402744 \\ Low\ Price, x_{K,3} \sim 1402744 \\ Close\ Price, x_{K,4} \sim 1402744 \end{bmatrix} THEN \begin{pmatrix} y_K = 0.5753 \\ +0.0985x_{K,1} + 0.4011x_{K,2} \\ -0.0503x_{K,3} + 0.0624x_{K,4} \end{pmatrix}$
10	$IF \begin{bmatrix} Open\ Price, x_{K,1} \sim 1402732 \\ High\ Price, x_{K,2} \sim 1402732 \\ Low\ Price, x_{K,3} \sim 1402732 \\ Close\ Price, x_{K,4} \sim 1402732 \end{bmatrix} THEN \begin{pmatrix} y_K = 0.5302 \\ +0.1419x_{K,1} + 0.4569x_{K,2} \\ -0.0128x_{K,3} + 0.1007x_{K,4} \end{pmatrix}$

To study the performance of the ALMMO system for regression, more experiments have been done and tabulated in Table 19. Here, the following state-of-the-art algorithms are used for comparison:

1) AnYa FRB system [60];

2) Fuzzily connected multi-model systems (FCMMS) [248];

3) Least square linear regression (LSLR) algorithm [249], which is widely used in the fields of finance and economy [183];

4) Sliding window least square linear regression (SWLSLR) algorithm [250], which is also widely used in the fields of finance and economy [183];

5) Evolving Takagi-Sugeno (ETS) algorithm [11];

6) Dynamic evolving neural-fuzzy inference system (DENFIS) [188];

7) Sequential adaptive fuzzy inference system (SAFIS) [251].

The width of the sliding window for LSLR algorithm is 200. The following three measures: the non-dimensional error index (NDEI) [252], the number of rules (NR) and execution time ($\mathrm{t}_{\mathrm{exe}}$, in seconds) are considered to evaluate the performance. In this numerical example, the data samples are standardized online. The detailed expression of NDEI is given in equation (6.2):

$$\mathrm{NDEI} = \sqrt{\frac{\sum_{i=1}^{K}(t_i - y_i)^2}{K\sigma_t^2}}, \tag{6.2}$$

where y_i is estimated value as the output of the system; t_i is the true value and σ_t is the standard deviation of the true value.

It is clear from Table 19 that the ALMMO system always exhibits a better performance than its competitors. In addition, the ALMMO system is also faster than the ETS, OLSLR, DENFIS and SAFIS predictors and it can also work on a sample by sample (does not need sliding window) basis like the ETS, AnYa and FCMMS.

For a further evaluation, the S&P dataset is considered. The following algorithms:

1) Evolving fuzzy neural networks (EFUNN) [253],

2) SeroFAM [254];

3) Simpl_eTS [255];

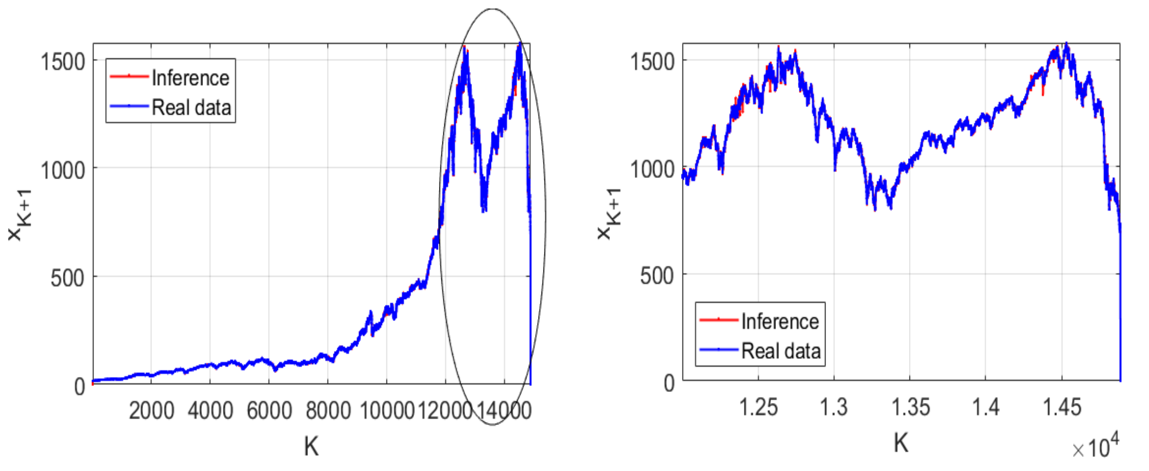
are additionally used for comparison. The comparative results are tabulated in Table 20. The prediction result of the S&P index dataset using ALMMO system is presented in Figure 51.

Table 19. Performance demonstration and comparison on QQSRM problem

Input and output	Performance			
	Algorithm	NDEI	NR	$\mathrm{t}_{\mathrm{exe}}$
Input: $\boldsymbol{x}_K$ Output: $x_{K+8,2}$	**ALMMO**	**0.135**	**10**	**4.63**
	FCMMS	0.143	4	7.77
	AnYa	0.164	3	3.69
	OLSLR	0.169	/	13.32
	SWLSLR	0.146	/	1.14
	ETS	0.183	6	36.52
	DENFIS	1.598	12	19.4
	SAFIS	0.554	20	23.16
Input: $\boldsymbol{x}_K$ Output: $x_{K+12,2}$	**ALMMO**	**0.152**	**10**	**4.40**
	FCMMS	0.162	4	7.75
	AnYa	0.197	3	3.66
	OLSLR	0.192	/	12.86
	SWLSLR	0.164	/	1.10
	ETS	0.234	8	45.71
	DENFIS	1.606	12	19.40
	SAFIS	1.007	17	22.59
Input: $\boldsymbol{x}_K$ Output: $x_{K+16,2}$	**ALMMO**	**0.168**	**10**	**4.42**
	FCMMS	0.175	4	7.60
	AnYa	0.185	3	3.69
	OLSLR	0.204	/	12.71
	SWLSLR	0.180	/	1.11
	ETS	0.191	8	48.85
	DENFIS	1.597	12	19.7
	SAFIS	0.964	18	22.54
Input: $\boldsymbol{x}_K$ Output: $x_{K+20,2}$	**ALMMO**	**0.178**	**10**	**4.43**
	FCMMS	0.189	4	7.84
	AnYa	0.195	3	3.67
	OLSLR	0.219	/	12.69
	SWLSLR	0.199	/	1.09
	ETS	0.200	8	48.60
	DENFIS	1.562	12	19.9
	SAFIS	1.042	11	16.73
Input: $\boldsymbol{x}_K$ Output: $x_{K+24,2}$	**ALMMO**	**0.192**	**10**	**4.68**
	FCMMS	0.204	4	7.78
	AnYa	0.231	3	3.66
	OLSLR	0.242	/	12.80
	SWLSLR	0.218	/	1.10
	ETS	0.271	7	45.71
	DENFIS	1.582	12	20.20
	SAFIS	0.779	14	22.55

Table 20. Performance demonstration and comparison on S&P problem

Algorithm	NDEI	NR
ALMMo	**0.013**	**8**
FCMMS	0.014	5
AnYa	0.018	11
OLSLR	0.020	
SWLSLR	0.018	
ETS	0.015	14
EFUNN	0.154	114.3
DENFIS	0.020	6
SAFIS	0.209	6
SeroFAM	0.027	29
FCMMS	0.045	7



(a) Prediction results (b) Zoom-in area (circle in (a))

Figure 51. Prediction result for the S&P problem.

As one can see from Table 20, for the S&P index data, the accuracy of the ALMMO system is 0.013, which ranks the first place from the 11 algorithms studied. It is also worth to notice that the S&P index dataset is, in fact, more smooth if compared with the QQSRM database [245]. Thus, one can conclude that the ALMMO system outperforms other prediction algorithms, especially in a more complicated situation. In addition, it is autonomously self-developing and does not require any user- or problem- specific parameters or prior assumptions to be made.

B. Classification

For the binary classification problems, the class of an unlabelled data sample, $\boldsymbol{x}$ can be determined by the output of the ALMMO system as:

$$\hat{y}(\boldsymbol{x}) = \mathrm{Round}\big(y(\boldsymbol{x})\big), \tag{6.3}$$

where $\mathrm{Round}\big(y(\boldsymbol{x})\big)$ denotes the operation of rolling $y(\boldsymbol{x})$ to the nearest integer.

The performance of the ALMMO system is tested on the PIMA [247] and occupancy detection [238] datasets as mentioned in the previous subsection. The most popular online and offline classification approaches are involved for further comparison. The ALMMO system is compared with the following well-known approaches:

1) Self-organizing map (SOM) with "winner takes all" principle [110] with the net size 9×9;

2) Learning vector quantization (LVQ) [109] with a hidden layer of size 32;

3) Back-propagation neural network (BPNN) with three hidden layers of size 16;

4) Naïve Bayes classifier;

5) SVM with Gaussian kernel function (SVM-G) [142];

6) SVM with linear kernel function (SVM-L) [142];

7) FLEXFIS-Class [256];

8) Dynamic evolving neural-fuzzy inference system (DENFIS) [188];

9) Peephole long short-term memory (LSTM) [67], [257] with a hidden layer of size 32;

10) AnYa FRB classifier [60];

11) eClass0 [160];

12) Simpl_eClass0 [162];

13) Fuzzily connected multi-model systems (FCMMS) [248].

Note that among the comparative algorithms listed above, FLEXFIS-Class, DENFIS, AnYa classifier, eClass0, Simpl_eClass0 and FCMMS are the multi-model approaches. The ALMMO system as well as AnYa classifier, eClass0, Simpl_eClass0 and FCMS are evolving approaches which can start classifying "from scratch" from the very first data sample and self-evolve with the data stream, while the other classifiers require pre-training. In contrast with the original AnYa FRB classifier, the ALMMO system uses an advanced, nonparametric mechanism for data cloud/fuzzy rule identification as well as the unimodal density-based membership functions.

For the PIMA dataset, 90% of the data samples are selected randomly for training and the rest are used for validation. For a fair comparison, all classifiers are pre-trained and the involved evolving approaches will stop learning after the training process. 30 Monto Carlo experiments are conducted; the means and standard deviations of the accuracy rates of the classification results are tabulated in Table 21.

Table 21. Overall classification performance-offline scenario

Algorithm	Multi-model (Yes/no)	On-line (Yes/no)	Can start from "scratch"?	Accuracy					
				Pima		Occupancy Detection-Testing set 1		Occupancy Detection-Testing set 2	
				Mean	STD [a]	Mean	STD	Mean	STD
ALMMO	**Yes**	**Yes**	**Yes**	**0.777**	**0.040**	**0.979**	**0.001**	**0.992**	**0.000**
SOM	No	No		0.728	0.046	0.974	0.007	0.946	0.010
LVQ	No	No		0.685	0.041	0.945	0.002	0.870	0.001
BPNN	No	No		0.763	0.043	0.930	0.039	0.906	0.052
Naïve Bayes	No	No		0.743	0.048	0.978	0.000	0.985	0.000
SVM-G	No	No		0.737	0.041	0.976	0.001	0.960	0.000
SVM-L	No	No		0.758	0.042	0.979	0.000	0.990	0.001
FLEXFIS	Yes	Yes	No	0.575	0.144	0.835	0.170	0.789	0.180
DENFIS	Yes	Yes	No	0.725	0.042	0.915	0.042	0.873	0.049
LSTM	No	No		0.655	0.053	0.862	0.103	0.897	0.048
AnYa	Yes	Yes	Yes	0.684	0.052	0.802	0.115	0.841	0.060
eClass0	Yes	Yes	Yes	0.602	0.078	0.948	0.020	0.871	0.019
Simpl_eClass0	Yes	Yes	Yes	0.633	0.085	0.935	0.050	0.939	0.029
FCMMS	Yes	Yes	Yes	0.533	0.111	0.896	0.091	0.833	0.072

[a] Standard deviation.

The confusion matrixes of the classification results obtained by selecting the first 90% (691 samples) of the dataset for training and using the rest of the data samples (77 samples) for validation are presented in Table 22.

For the occupancy detection dataset, the classifiers are firstly trained with the training set, and classification is conducted on the two testing sets separately with the trained classifiers in an offline scenario. 30 Monto Carlo experiments are conducted by randomly scrambling the order of the training samples and the overall accuracies of the classification results are presented in Table 21. The average true positive rates and true negative rates of the classification results on the two testing sets obtained with the classifiers trained by the original training set are tabulated in Table 23.

Table 22. Confusion matrices and the classification accuracy on PIMA dataset

Algorithm	Actual/ Classification	Negative	Positive	Accuracy
ALMMO	**Negative**	**43 samples**	**3 samples**	**0.792**
	Positive	**13 samples**	**18 samples**	
SOM	Negative	37 samples	9 samples	0.714
	Positive	13 samples	18 samples	
LVQ	Negative	39 samples	7 samples	0.597
	Positive	24 samples	7 samples	
BPNN	Negative	39 samples	7 samples	0.792
	Positive	9 samples	22 samples	
Naïve Bayes	Negative	38 samples	8 samples	0.766
	Positive	10 samples	21 samples	
SVM-G	Negative	36 samples	10samples	0.779
	Positive	7 samples	24 samples	
SVM-L	Negative	39 samples	7 samples	0.792
	Positive	9 samples	22 samples	
FLEXFIS	Negative	46 samples	0 samples	0.571
	Positive	31 samples	0 samples	
DENFIS	Negative	39 samples	7 samples	0.727
	Positive	14 samples	17 samples	
LSTM	Negative	44 samples	2 samples	0.584
	Positive	30 samples	1 samples	
AnYa	Negative	36 samples	10 samples	0.714
	Positive	12 samples	19 samples	
eClass0	Negative	24 samples	22 samples	0.597
	Positive	9 samples	22 samples	
Simpl_ eClass0	Negative	34 samples	12 samples	0.649
	Positive	15 samples	16 samples	
FCMMS	Negative	28 samples	18 samples	0.546
	Positive	17 samples	14 samples	

A comparison in an online scenario is also conducted between the fully evolving algorithms that can start “from scratch”, namely the ALMMO system, AnYa FRB classifier, eClass0, Simpl_eClass0 and FCMMS, by considering the PIMA dataset as a data stream. In this experiment, the order of the data samples in the stream is randomly determined, and the algorithms start classifying from the first data sample and keep updating the system structure along with the arrival of new data samples. Similarly, the whole occupancy detection dataset is considered as a data stream, and 30 Monto Carlo experiments are conducted by randomly scrambling the order of the data samples to evaluate the performance of the five evolving algorithms in an online scenario. The average results of the two experiments are reported in Table 24.

Table 23. The average true positive rates and true negative rates of the classification results on occupancy detection dataset

Algorithm	Occupancy Detection-Testing set 1		Occupancy Detection-Testing set 2	
	True Negative Rate	True Positive Rate	True Negative Rate	True Positive Rate
ALMMO	**0.968**	**0.998**	**0.992**	**0.994**
SOM	0.965	0.991	0.945	0.943
LVQ	0.932	0.965	0.847	0.991
BPNN	0.971	0.883	0.918	0.866
Naïve Bayes	0.968	0.995	0.982	0.993
SVM-G	0.965	0.996	0.950	0.993
SVM-L	0.967	0.998	0.990	0.992
FLEXFIS	0.810	0.826	0.870	0.745
DENFIS	0.969	0.790	0.936	0.699
LSTM	0.967	0.770	0.947	0.750
AnYa	0.985	0.482	0.967	0.348
eClass0	0.931	0.972	0.845	0.973
Simpl_eClass0	0.960	0.911	0.966	0.862
FCMMS	0.919	0.895	0.828	0.909

Table 24. Overall classification performance-online scenario

Algorithm	Accuracy	
	Pima	Occupancy Detection
ALMMO	**0.751**	**0.986**
AnYa	0.666	0.949
eClass0	0.570	0.931
Simpl_eClass0	0.584	0.968
FCMMS	0.545	0.924

From Table 21 and Table 24 one can see that, the ALMMO system provides highly accurate classification results in the numerical examples in both offline and online scenarios compared with its competitors. It is worth to be noticed that, the ALMMO system is an online classifier and can work “from scratch”. The most important point is that the ALMMO system is entirely data driven and is free from unrealistic assumptions, restrictions or problem- or user- specific prior knowledge.

6.2.2. Zero Order Autonomous Learning Multi-Model Classifier

6.2.2.1. Benchmark Problems for Evaluation

In this section, the performance of the ALMMO-0 classifier is evaluated based on the following popular benchmark datasets:

1) Banknote authentication dataset [235];

2) **Monk's** problem dataset [258];

3) **Tic-Tac-Toe** endgame dataset [259];

4) **CNAE-9** dataset [260].

The details of the banknote authentication dataset have been given in Table 9, the details of the other three benchmark datasets are given in Table 25.

Table 25. Details of benchmark datasets for evaluating ALMMO-0 classifier

Dataset	Number of Classes	Number of Features	Number of Samples
Monk's	2	6	169 (training set)
			432 (testing set)
Tic-Tac-Toe	2	9	958
CNAE-9	2	856	1080

During the numerical experiments, for the banknote authentication, Tic-Tac-Toe endgame and CNAE-9 datasets, 80% of the data samples of each class are randomly selected out for training and the rest is used for validation.

6.2.2.2. Performance Evaluation and Discussion

Firstly, the confusion matrix of the classification result of the ALMMO-0 classifier on the Monk's problem is given in Table 25. In this section, the performance of the classifier is compared with the following well-known classification algorithms:

1) SVM classifier with Gaussian kernel [142];

2) Naïve Bayes classifier [3];

3) KNN classifier [261];

4) Decision tree (DT) classifier [262];

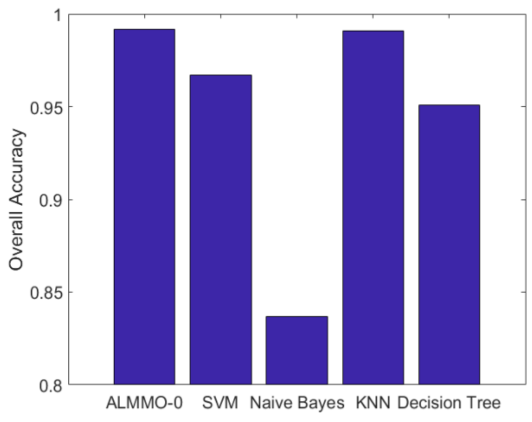
and the confusion matrices of the classification results obtained by the four algorithms are also tabulated in the same table.

Table 26. Confusion matrices of classification results on Monk's problem

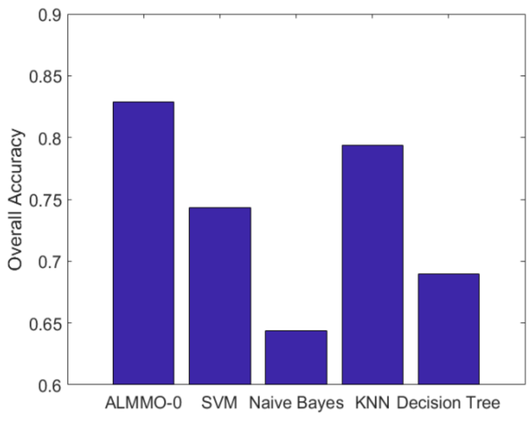
Algorithm	Actual	Classification	
		0	1
ALMMO-0	**0**	**82.07% 238 samples**	**17.93% 52 samples**
	1	**15.49% 22 samples**	**84.51% 120 samples**
SVM	0	85.17% 247 samples	14.83% 43 samples
	1	47.89% 68 samples	52.11% 74 samples
Naïve Bayes	0	90.34% 262 samples	9.66% 28 samples
	1	88.73% 126 samples	11.27% 16 samples
KNN	0	82.07% 238 samples	17.93% 52 samples
	1	26.06% 37 samples	73.94% 105 samples
DT	0	71.03% 206 samples	28.97% 84 samples
	1	35.21% 50 samples	64.79% 92 samples

For a further evaluation and comparison between the five classifiers, the overall accuracies of the classification results on the four benchmark datasets and the time consumptions for training are depicted in Figure 52 and Figure 53, respectively. Due to the very high dimensionality of the CNAE-9 dataset, the Naïve Bayes classifier failed to give any valid result on this one.

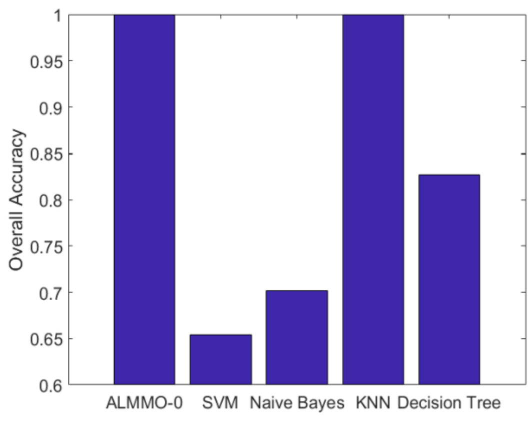
From the four numerical examples above one can see that the SVM classifier with Gaussian kernel [142] requires more time for training and it is less effective in handling high dimensional problems. The naïve Bayes classifier [3] is the fastest one due to its simplicity and its performance is quite stable, though not high. The KNN classifier [261] is also very efficient and its classification accuracies in some problems are comparable to the ALMMO-0 classifier, but it is not effective in handling high-dimensional datasets with complex structure. In addition, its interpretability is not high because it does not reveal an internal structure. The classification accuracy of decision tree classifier [262] is relatively low and it is less efficient in handling lower dimensional problems.



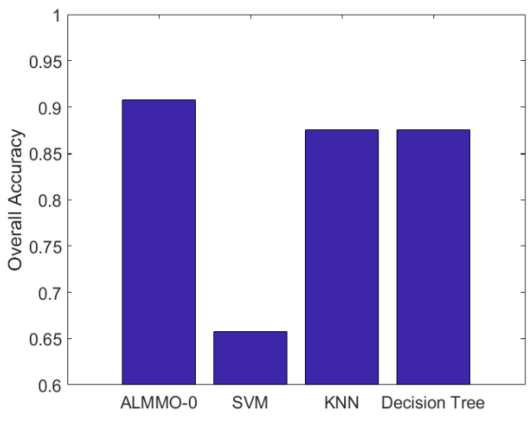
(a) Banknote authentication



(b) Monk’s problem



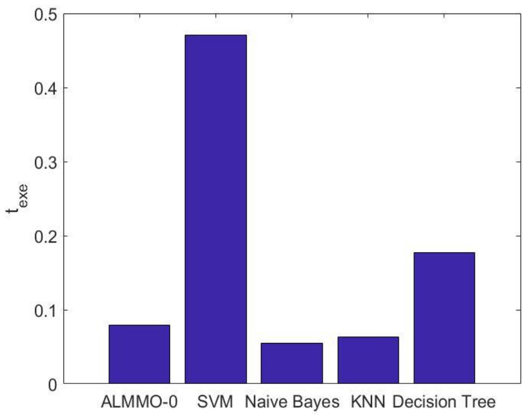
(c) Tic-Tac-Toe endgame



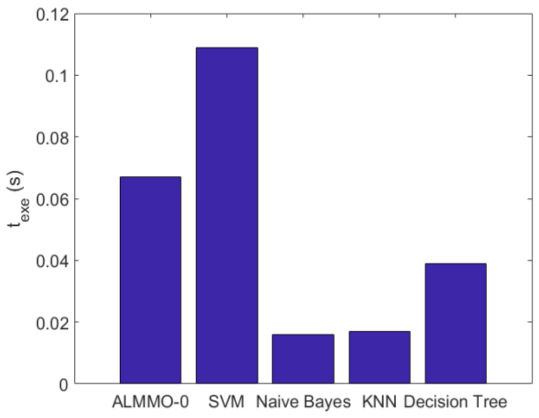
(d) CNAE-9

Figure 52. Overall classification accuracy on the four benchmark datasets.

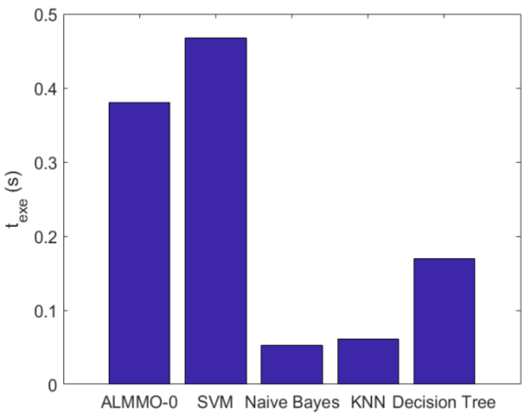
In contrast, the ALMMO-0 classifier can exhibit excellent performance in all the four real benchmark problems and, at the same time, still keeps its high computational efficiency. It is fully autonomous and offers good interpretability. Moreover, it is evolving in nature.



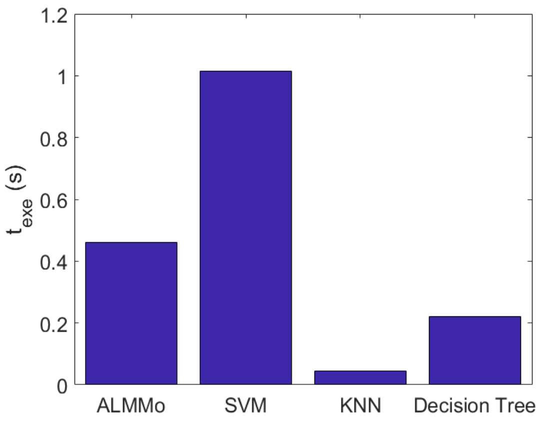
(a) Banknote authentication



(b) Monk's problem



(c) Tic-Tac-Toe endgame



(d) CNAE-9

Figure 53. Overall time consumption for training on the four benchmark datasets.

6.2.3. Self-Organising Fuzzy Logic Classifier

6.2.3.1. Benchmark Problems for Evaluation

In this section, the performance of the SOFL classifier is evaluated based on the following challenging benchmark datasets:

1) **Occupancy** detection dataset[238];

2) **Optical** recognition of handwritten digits dataset [263];

3) **Multiple** features dataset [244];

4) **Letter** recognition dataset [240].

The details of the occupancy detection and letter recognition datasets have been given in Table 11 and the details of the multiple features dataset can be found in Table 14. The optical recognition dataset consists of one training set with 3823 data samples and one testing set with 1797 data samples [263]. There are 10 classes within the dataset, and each data sample has 64 attributes.

6.2.3.2. Performance Evaluation and Discussion

Firstly, the influence of different levels of granularity on the classification results of the SOFL approach is studied, and the occupancy detection and optical recognition datasets are used in this experiment. In this example, the offline scenario is considered only and the level of granularity, G is varied from 1 to 12. The classification results are tabulated in Table 27 and the performance is measured in terms of classification accuracy (Acc), the number of identified prototypes, denoted by P ($\mathrm{P} = \sum_{i=1}^{C} P_i$) and the training time consumption in seconds, denoted by $\mathrm{t_{exe}}$. Here, the Mahalanobis distance, Euclidean distance and cosine dissimilarity are used.

In general, Euclidean distance is the most widely used distance metric, and its effectiveness and validity as the distance measure, in most cases, are guaranteed [36]. If the data generation model follows a Gaussian distribution or some similar distributions, Mahalanobis distance would be a good choice. While in high dimensional problems, cosine dissimilarity is free from the "curse of dimensionality" [264], [265] and thus, is more effective and more frequently used.

However, for the optical recognition dataset, as the co-variance matrix of the data is not always positive definite, as a result, only the results obtained using the Euclidean distance and cosine dissimilarity are considered. The results tabulated are the average of 10 Monte Carlo experiments by randomly descrambling the order of the training samples.

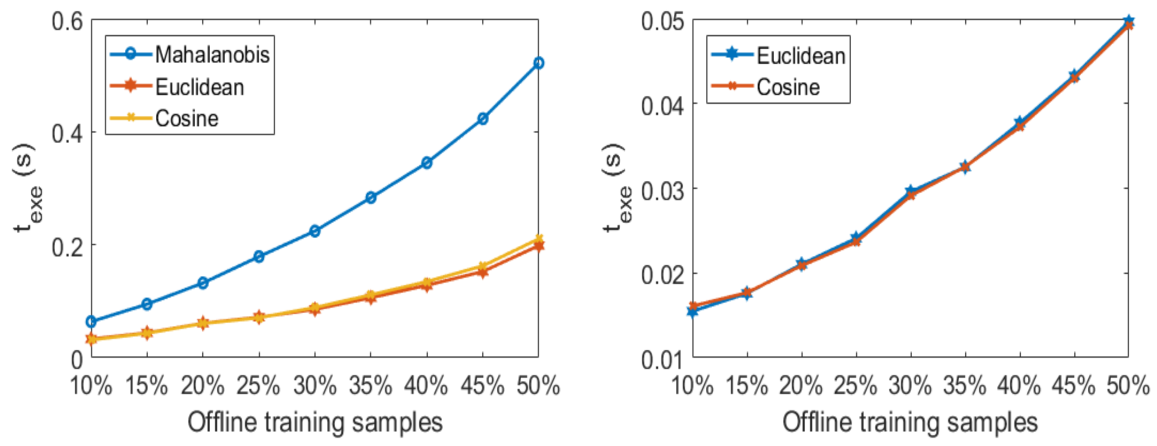
From Table 27 one can see that, in general, the higher level of granularity is chosen, the higher accuracy the SOFL classifier can exhibit during classification, but the more prototypes the classifier identifies, which can lower down the computation- and memory-efficiency. It is worth to notice that the proposed approach produced the same result in 10 Monte Carlo experiments, which demonstrates that the SOFL classifier is invariant to the changes in the order of data samples during the offline training.

One may also notice from Table 27 that the type of distance/dissimilarity measure used also influences the performance of the proposed approach. As the SOFL classifier accommodates various types of distance/dissimilarity measures, one can use the current knowledge of the problem domain to choose the appropriate distance measure.

Table 27. Influence of granularity on classification performance

Dataset	Distance	Measures	G					
			1	2	3	4	5	6
Occupancy	Mahalanobis	Acc	0.8942	0.8920	0.9038	0.9426	0.9494	0.9532
		P	14	31	55	116	217	339
		t_{exe}	2.80	2.97	3.08	3.11	3.16	3.14
	Euclidean	Acc	0.8107	0.8403	0.8618	0.9112	0.9382	0.9513
		P	16	46	77	137	201	281
		t_{exe}	2.15	2.31	2.47	2.55	2.59	2.65
	Cosine	Acc	0.8109	0.8161	0.8877	0.9261	0.9481	0.9519
		P	12	43	72	108	167	217
		t_{exe}	2.13	2.31	2.55	2.56	2.63	2.72
Optical	Euclidean	Acc	0.9160	0.9421	0.9499	0.9716	0.9766	0.9761
		P	25	48	105	214	409	643
		t_{exe}	0.09	0.10	0.10	0.09	0.10	0.10
	Cosine	Acc	0.9087	0.9421	0.9588	0.9649	0.9699	0.9733
		P	25	50	116	238	417	655
		t_{exe}	0.09	0.10	0.09	0.09	0.10	0.10
Dataset	Distance	Measures	L					
			7	8	9	10	11	12
Occupancy	Mahalanobis	Acc	0.9539	0.9543	0.9543	0.9543	0.9543	0.9543
		P	549	786	1029	1279	1433	1512
		t_{exe}	3.33	3.16	3.26	3.29	3.36	3.32
	Euclidean	Acc	0.9564	0.9579	0.9584	0.9588	0.9588	0.9588
		P	395	525	663	783	939	1094
		t_{exe}	2.72	2.68	2.69	2.68	2.74	2.70
	Cosine	Acc	0.9558	0.9557	0.9559	0.9559	0.9559	0.9559
		P	288	388	507	650	825	1007
		t_{exe}	2.78	2.68	2.75	2.70	2.79	2.77
Optical	Euclidean	Acc	0.9811	0.9833	0.9833	0.9833	0.9839	0.9839
		P	840	950	1012	1034	1046	1048
		t_{exe}	0.10	0.10	0.10	0.11	0.11	0.11
	Cosine	Acc	0.9755	0.9761	0.9761	0.9761	0.9761	0.9761
		P	843	960	1013	1039	1039	1046
		t_{exe}	0.11	0.11	0.11	0.11	0.11	0.11

Secondly, the classification performance of the SOFL classifier with different amounts of offline training samples is investigated. In this example, the letter recognition and multiple features datasets are used. As the two datasets are both highly complex, the 12th level of granularity is chosen ($G = 12$) to ensure the SOFL classifier can learn sufficient details. The percentage of offline training samples is changed from 5% to 50% and the classification is conducted on the rest 50% of the data in an offline scenario. The results are tabulated in Table 28, which are the averages of 10 Monte Carlo experiments by randomly selecting the training set and testing set. The corresponding average training time consumption (in seconds) is depicted in Figure 54.



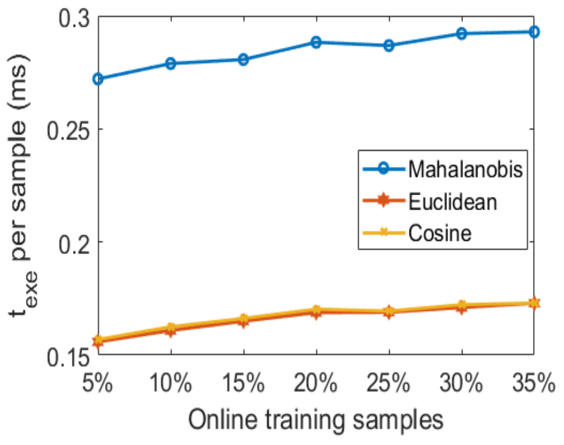
(a) Letter recognition (b) Multiple features

Figure 54. The average training time consumption with different amounts of training samples.

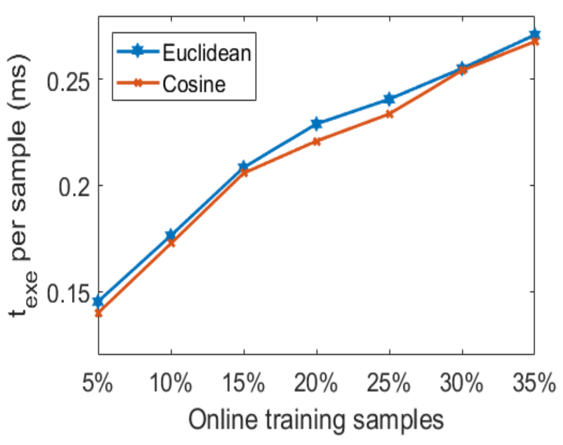
Table 28. Classification performance (in accuracy) with different amount of data for offline training

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

In order to investigate the performance of the SOFL classifier in an online scenario, an extra experiment is conducted on the two datasets. The SOFL classifier is firstly trained with 15% of the data samples in an offline scenario, and then, is trained in an online scenario by using different amounts (from 5% to 35%) of data samples on a sample-by-sample basis. The classification accuracy of SOFL classifier is evaluated on the rest 50% of data samples. The average performance is tabulated in Table 29 after 10 Monte Carlo experiments by randomly selecting the offline training set, online training set and testing set. The corresponding average time consumption per data sample (in millisecond) during the online training process is given in Figure 55. In both Table 28 and Table 29, the classification results on the multiple feature dataset using the Mahalanobis distance is not given for the same reason mentioned at the beginning of this subsection.



(a) Letter recognition



(b) Multiple features

Figure 55. The average training time consumption per sample during the online training.

Table 29. Classification performance (in accuracy) with different amount of data for online training following the offline training with 15% of the data

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

From Table 28 one can conclude that the more data samples the SOFL classifier is provided with during the offline training stage, the better performance it can exhibit in the classification stage. Table 29 shows that the performance of the SOFL classifier can be further improved through the online update with more training data samples after the offline training, which is one of the very strong advantages of the proposed approach. In real applications, new data is more often coming in the form of data streams, which may exhibit shifts and/or drifts in the data pattern [117]. With the ability of self-evolving online learning, the SOFL classifier is able to continuously follow the changing data pattern without full retraining, which largely enhances the efficiency and saves the computational resources. Figure 56 demonstrates the very high computational efficiency (less than 0.3 millisecond per data sample) for the SOFL classifier to self-evolve recursively on a sample-by-sample basis.

To further evaluate the performance of the SOFL classifier with $G = 12$, a number of state-of-the-art approaches are involved for comparison in an offline scenario based on the four benchmark datasets listed in subsection 6.2.3.1:

1) SVM classifier [142];

2) KNN classifier [132];

3) DT classifier [262];

4) SOM classifier [110];

6) DENFIS classifier [188];

7) eClass-0 classifier [160];

8) TEDAClass classifier [51].

During the comparison, the SVM classifier uses a linear kernel; for the KNN classifier, k is equal to 10; SOM classifier applies "winner takes all" principle with a net size of 9×9. As one may obtain the covariance matrices that are not positive definite from the optical recognition and multiple feature datasets, only the Euclidean distance and cosine dissimilarity are used for these two datasets during the comparison. For letter recognition and multiple features datasets, 50% of the data for training are used and the rest for testing. The performance comparison is tabulated in Table 30. The reported results are the averages of 10 Monte Carlo experiments. In the experiments, the DENFIS classifier failed in both, the optical recognition and multiple feature datasets because of the high dimensionality. From Table 30 one can see that, the SOFL classifier can exhibit very high performance on the four benchmark problems with a very short training process.

Table 30. Performance evaluation and comparison for the SOFL classifier

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

6.2.4. Autonomous Anomaly Detection Algorithm

6.2.4.1. Benchmark Problems for Evaluation

In this section, the following datasets are considered for evaluating the performance of the AAD algorithm on anomaly detection:

1) Synthetic Gaussian dataset [37];

2) **User knowledge** modelling dataset [266];

3) **Wine quality** dataset [267];

4) **Wilt** dataset [268].

The synthetic Gaussian dataset [37] contains 720 samples with 2 attributes. There is 1 larger cluster and 2 smaller ones grouping 700 data samples between them. In addition, 4 collective anomalous sets formed by 18 samples as well as 2 single anomalies were identified. The models of the three major clusters extracted from the data ($\boldsymbol{\mu}, \boldsymbol{\sigma}, S$) are as follows (in the form of model, $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma})$ and support, S):

Major cluster 1: $\boldsymbol{x} \sim N\left([0 \quad 3], \begin{bmatrix} 0.09 & 0 \\ 0 & 0.09 \end{bmatrix}\right)$, 400 samples;

Major cluster 2: $\boldsymbol{x} \sim N\left([2.5 \quad 3], \begin{bmatrix} 0.16 & 0 \\ 0 & 0.16 \end{bmatrix}\right)$, 150 samples;

Major cluster 3: $\boldsymbol{x} \sim N\left([2.5 \quad 0], \begin{bmatrix} 0.16 & 0 \\ 0 & 0.16 \end{bmatrix}\right)$, 150 samples.

The models of the 4 collectives anomalous sets are:

Anomalous set 1: $\boldsymbol{x} \sim N\left([0 \quad 1], \begin{bmatrix} 0.09 & 0 \\ 0 & 0.09 \end{bmatrix}\right)$, 5 samples;

Anomalous set 2: $\boldsymbol{x} \sim N\left([4.5 \quad 0], \begin{bmatrix} 0.09 & 0 \\ 0 & 0.09 \end{bmatrix}\right)$, 4 samples;

Anomalous set 3: $\boldsymbol{x} \sim N\left([4.5 \quad 4], \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}\right)$, 5 samples;

Anomalous set 4: $\boldsymbol{x} \sim N\left([1 \quad -1], \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}\right)$, 4 samples;

and the two single anomalies are $[2 \quad 5]$ and $[1.5 \quad 5]$.

This dataset is visualized in Figure 56, where the anomalies are circled in by red ellipses. It is important to stress that, collective anomalies and single anomaly close to the global mean of the dataset are very difficult to detect using traditional approaches.

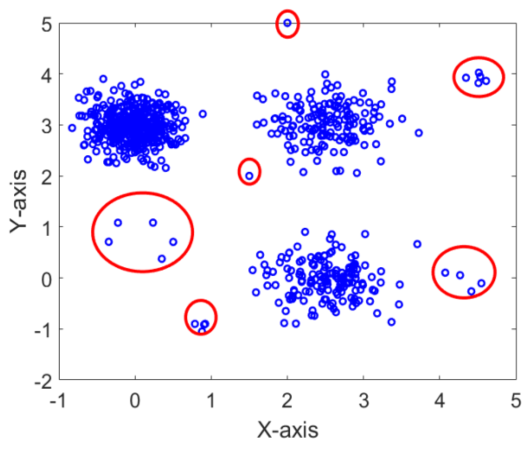


Figure 56. Visualization of the synthetic Gaussian dataset.

The user knowledge modelling dataset contains 403 samples, and each data sample has five attributes and one label, which represents the level of the user knowledge [266]. There are four levels of the user knowledge, 1) high (130 samples), 2) middle (122 samples), 3) low (129 samples) and 4) very low (50 samples). The existing anomalies in four classes are listed by their IDs as follows [37]:

1) High: 2, 10, 14, 34, 182, 187, 190, 210, 230, 246, 258, 313, 317, 318, 378, 379, 384, 391, 399 and 400;

2) Middle: 4, 13, 50, 57, 62, 65, 124, 130, 162, 207, 208, 211, 212, 214, 222, 223, 245, 250, 257, 272, 286, 362, 372, 373 and 403;

3) Low: 3, 5, 18, 53, 61, 128, 129, 131, 198, 204, 244, 319, 374, 395 and 401;

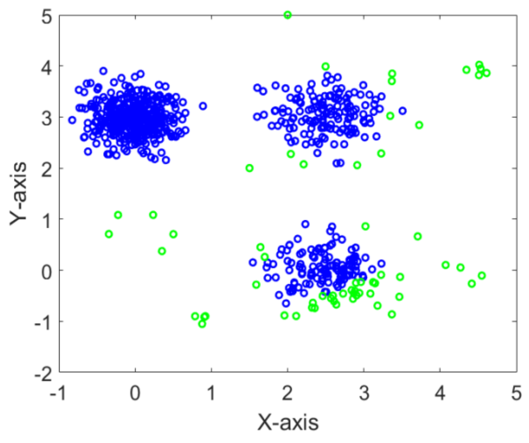
4) Very low: 1, 17, 117, 197, 209, 288, 310, 312, and 314.

Wine quality dataset is related to the quality of red Portuguese "Vinho Verde" wine. This dataset has 1599 data samples with 11 attributes and one label, which corresponds to the score of quality of the wine from 3 to 8 [267]. This dataset is not balanced as there are much more normal wines than excellent or poor ones. There are 10 samples with score 3, 53 samples with score 4, 681 samples with score 5, 638 samples with score 6, 199 samples with score 7 and 18 samples with score 8. The number of existing anomalies in each class are listed as follows: 1) Score 3: 1; 2) Score 4: 3; 3) Score 5: 50; 4) Score 6: 42; 5) Score 7: 9; 6) Score 8: 3. In total, there are 108 anomalies [37].

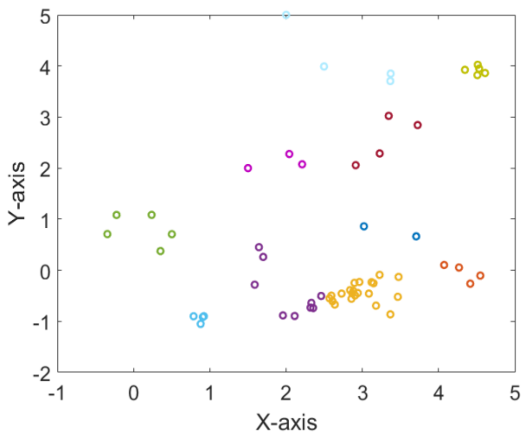
Wilt dataset comes from a remote sensing study involving detecting diseased trees in Quickbird imagery. There are two classes in the dataset: 1) "diseased trees" class (74 samples) and 2) "other land cover" class (4265 samples). Each sample has 5 attributes and 1 label (“other land cover” or diseased trees”). There are 120 anomalies with the label “other land cover” and no anomaly in the “diseased trees” class [37].

6.2.4.2. Performance Evaluation and Discussion

Using the proposed approach, 61 potential anomalies identified from the synthetic dataset in the first stage are depicted in Figure 57 (a) (the green circles). In stage 2, 10 data clouds are formed from the potential anomalies as presented in Figure 57(b), where the circles with the different colours are the data samples from different data clouds. There are 31 anomalies identified in the final stage of the proposed approach as shown in Figure 58 (red circles).



(a) The potential anomalies



(b) The data clouds

Figure 57. The identified potential anomalies and the data clouds formed by them.

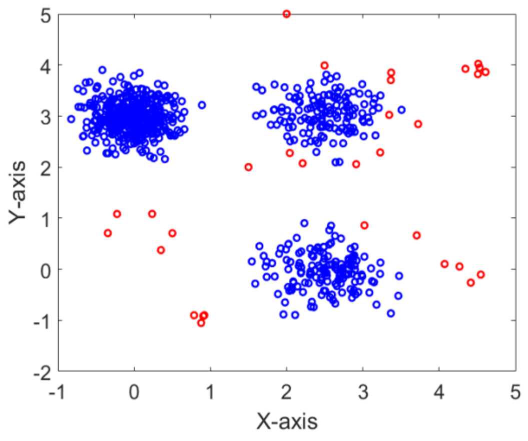


Figure 58. The identified anomalies by the AAD algorithm.

Figure 56, Figure 57 and Figure 58 show that, the AAD algorithm successfully identified all the anomalies in this dataset, because both the mutual distribution and the ensemble properties of the data samples have been considered.

For a further evaluation of the ADD algorithm, two well-known traditional approaches are used for comparison:

1) The well-known "3σ" approach [49], [203], [204];

2) Outlier detection using random walks (ODRW) approach [199].

It has to be stressed that the "3σ" approach is based on the global mean and global standard deviation. The outlier detection using random walks approach requires three parameters to be predefined: 1) error tolerance, ε; 2) similarity threshold, T and 3) number of anomalies, Na. In this subsection, $\varepsilon = 10^{-6}$ and $T = 0.9$ [199]. To make the results comparable, Na is set to be the same number of the anomalies identified by the AAD algorithm.

The global mean and the standard deviation of the dataset are $\boldsymbol{\mu} = [1.1077 \quad 2.3263]$ and $\boldsymbol{\sigma} = [1.3401 \quad 1.3228]$, and the "$3\sigma$" approach failed to detect any anomalies. The result using the ODRW approach is shown in Figure 59, where the red circles are the identified anomalies. As one can see, this approach ignored the majority of the anomalies (circled within the yellow ellipsoids).

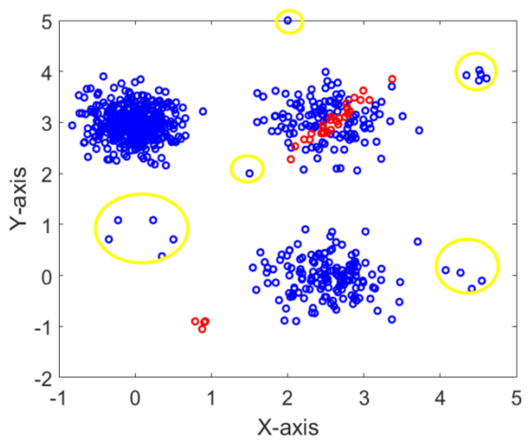


Figure 59. The identified anomalies by the ODRW algorithm.

For the user knowledge modelling dataset, the AAD algorithm identified 10 anomalies as tabulated in Table 30. It has to be stressed that the labels (Table 31) of the data are not used in the anomaly detection and they are just used for posterior comparison. From the table one can see that the detected anomalies have significantly lower or higher values compared with other members of the classes to which they may belong. Nine out of the identified 10 anomalies are in the anomaly lists in the previous subsection.

Table 31. Identified anomalies from the user knowledge modelling dataset

# ID	Values	Label
1	[0.0000 0.0000 0.0000 0.0000 0.0000]	Very low
2	[0.0800 0.0800 0.1000 0.2400 0.9000]	High
5	[0.0800 0.0800 0.0800 0.9800 0.2400]	Low
17	[0.0500 0.0700 0.7000 0.0100 0.0500]	Very low
187	[0.4950 0.8200 0.6700 0.0100 0.9300]	High
210	[0.8500 0.0500 0.9100 0.8000 0.6800]	High
222	[0.7700 0.2670 0.5900 0.7800 0.2800]	Middle
242	[0.7100 0.4600 0.9500 0.7800 0.8600]	High
399	[0.9000 0.7800 0.6200 0.3200 0.8900]	High
403	[0.6800 0.6400 0.7900 0.9700 0.2400]	Middle

For a better comparison, the following five measures [199] are used for performance evaluation:

1) Number of identified anomalies (Na): $\text{Na} = \text{True Positive} + \text{False Positive}$;

2) Precision (Pr): the rate of true anomalies in the detected anomalies, $\text{Pr} = \frac{\text{True Positive}}{\text{True Positive+False Positive}}$;

3) False alarm rate (Fa): the rate of the true negatives in the identified anomalies, $\text{Fa} = \frac{\text{False Positive}}{\text{True Negative+False Positive}}$;

4) Recall rate (Re): the rate of true anomalies the algorithms missed, $\text{Re} = \frac{\text{False Nagetaive}}{\text{True Positive+False Nagetaive}}$;

5) Execution time (t_{exe}): in seconds.

The detection results obtained by the three algorithms on the user knowledge modelling, wine quality and wilt datasets in terms of the five measures are tabulated in Table 32.

From Table 32 one can see that the proposed approach is able to detect the anomalies with higher precision and lower false alarm rate compared with the "3σ" approach and the ODRW approach.

The "3σ" approach is the fastest due to its simplicity. However, the performance of the "3σ" approach is decided by the structure of the data as it focuses only on the samples exceeding the global 3σ range around the mean. However, when the dataset is very complex i.e. contains a large number of clusters, or the anomalies are close to the global mean, "3σ" approach fails to detect all outliers.

In contrast, the AAD algorithm can identify the anomalies based on the ensemble properties of the data in a fully unsupervised and autonomous way. It takes not only the mutual distribution of the data within the data space, but also the frequencies of occurrences into consideration. It provides a more objective way for anomaly detection. More importantly, its performance is not influenced by the structure of the dataset and is equally effective in detecting collective as well as individual anomalies.

Table 32. Performance comparison of the anomaly detection algorithms

Dataset	Algorithm	Na	Pr	Fa	Re	t_{exe} (s)
User knowledge	ADD	10	90.00%	0.30%	86.96%	0.09
	3σ	1	100.00%	0.00%	98.55%	0.00
	ODRW	10	50.00%	1.50%	92.75%	0.27
Wine quality	ADD	36	63.89%	0.87%	78.70%	0.24
	3σ	141	30.05%	6.57 %	60.19%	0.01
	ODRW	36	0.00%	2.41%	100.00%	31.14
Wilt	ADD	84	71.43%	0.57%	50.00%	1.08
	3σ	176	34.66%	2.73%	49.17%	0.01
	ODRW	84	58.33%	0.83%	59.17%	863.76

6.3. Evaluation of the Transparent Deep Learning Systems

In this section, the performance of the transparent deep learning systems presented in chapter 5, namely

1) Fast feedforward nonparametric deep learning (FFNDL) network;

2) Deep rule-based (DRB) system;

3) Semi-supervised deep rule-based (SSDRB) classifier;

are evaluated based on benchmark image sets. Their performance is also compared with a number of state-of-the-art approaches.

6.3.1. Fast Feedforward Nonparametric Deep Learning Network

6.3.1.1. Benchmark Problems for Evaluation

A. Handwritten digits recognition

The MNIST database [213] is used as the benchmark dataset for evaluating the performance of the FFNDL network on handwritten digits recognition problem. The details of the MNIST datasets have been given in section 5.4.1.

B. Image classification

The first numerical experiment is to evaluate the performance of the FFNDL network on human action recognition. This numerical example is conducted based on a subset of the well-known human action dataset [269]. The dataset contains six classes (walking, jogging, running, boxing, hand waving and hang clapping) with 100 images per class randomly extracted from 18 videos with the same background (three videos per class). The visual examples of the images are presented in Figure 60. In the experiments, the original images are converted to 64 × 64 pixels size because some of the actors are not large enough within the images.

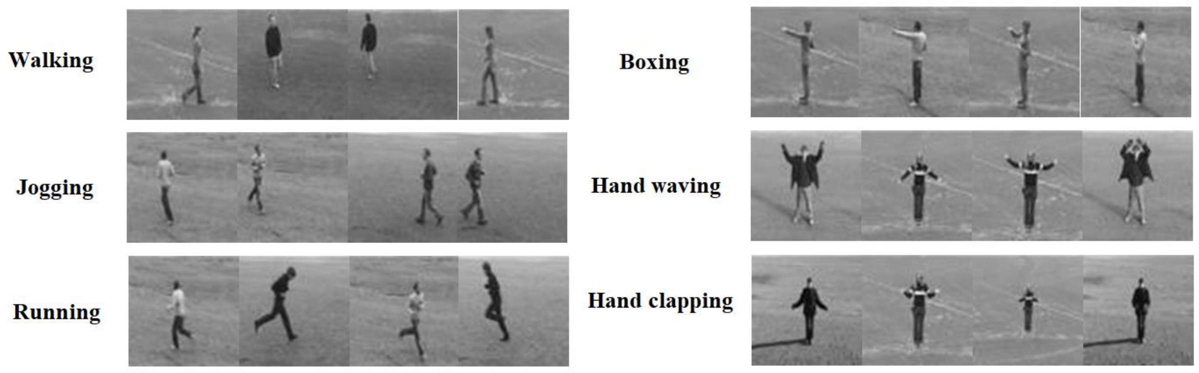


Figure 60. Example images on human action recognition problem.

The second numerical experiment is for object classification, which is based on a subset of the well-known Wang dataset [270]. The subset consists of eight classes with 40 images in each class. The eight classes are: airplanes, cars, dinosaur, dolls, doors, motorbikes, roses and sailing ships. Example images of the eight classes are given in Figure 61. The original images are converted to 64 × 64 pixels size.

The image rescaling setting used in these two experiments is decided through numerical experiments empirically. Determining the most suitable image rescaling operation for a specific problem may require some prior knowledge. Nonetheless, in general, one need to make sure that the dimensionality of the extracted feature vector is smaller than the dimensionality of the image itself to avoid over-sampling, which makes the feature vector noisier.

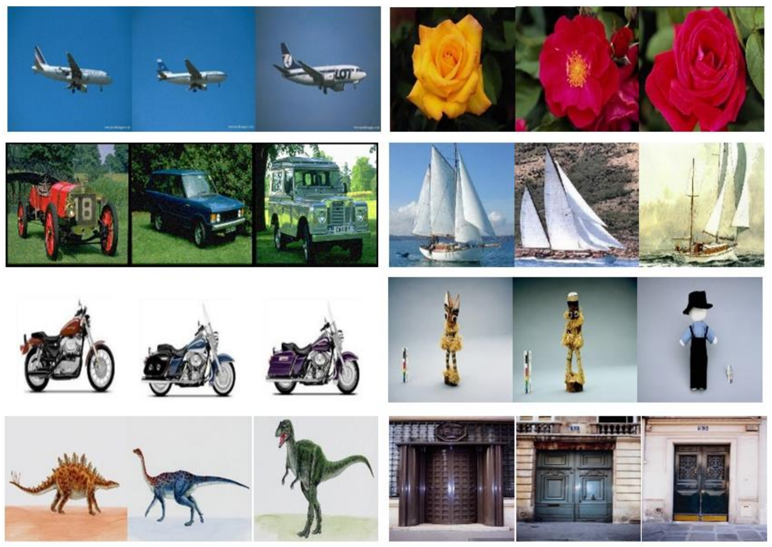


Figure 61. Example images on object classification problem.

6.3.1.2. Performance Evaluation and Discussion

In the numerical experiments in this subsection, the size of the sliding window for local aggregations extraction is 7×7 ($n = 7$); the size and stride of the sliding window for the grid segmentation is 2×2 ($k = 2$); and 1 ($w = 1$).

A. Handwritten digits recognition

In this experiment, the original handwritten digit images are used, and therefore, there is $d = 14$. The classification result is tabulated in Table 33. The corresponding amount of time consumed by the FFNDL network is presented in Table 34.

The performance of the FFNDL network is also compared with several well-known algorithms:

1) Neocognitron neural network [271];

2) eClass1 using GIST and Haar global features [160];

3)TEDAClass classifier using GIST and Haar global features [272].

The classification results are tabulated in Table 33 compared with the results of the previously published methods [160], [271], [272]. The results are visualized in Figure 62.

Table 33. Recognition results and comparison on MNIST dataset

Training set	Neocognitron	eClass1	TEDAClass	**FFNDL**
1000	94.42%	86.54%	95.92%	**92.70%**
2000	96.04%	96.42%	96.70%	**93.89%**
3000	96.34%	96.55%	96.67%	**94.93%**
4000	96.62%	96.62%	96.88%	**95.31%**
5000	96.94%	96.85%	97.16%	**95.54%**
10000	-	97.19%	97.38%	**96.31%**
20000	-	97.32%	97.53%	**96.67%**
30000	-	97.46%	97.68%	**96.86%**
40000	-	97.45%	97.66%	**96.97%**
50000	-	97.46%	97.65%	**97.03%**
60000	-	97.46%	97.63%	**97.11%**

Table 34. Time consumption for training process of the FFNDL network

Train set	1000	2000	3000	4000	5000	10000
Time	21.4	41.0	72.3	113.4	164.5	527.9
Train set	10000(4)[1]	20000	30000	40000	50000	60000
Time	390.2	1906.8	3975.3	7214.9	10672.2	15681.7

[1] 4 local workers.

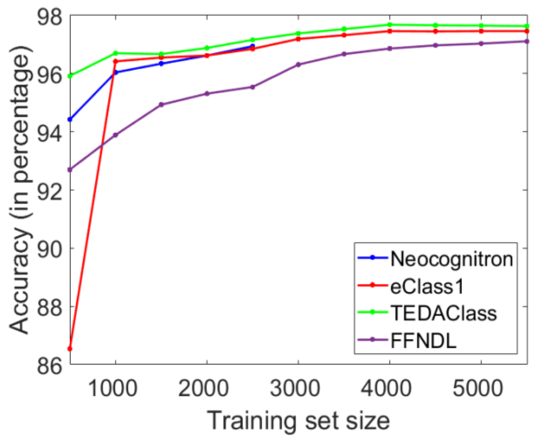


Figure 62. Curves of classification accuracy of the four methods on MNIST dataset.

As it is shown in Table 33 and Figure 62, the classification accuracy of the FFNDL network reaches 97.11% after all 60000 training images are used, which is slightly worse than the eClass1 and TEDAClass but outperforms Neocognitron (and other approaches, i.e. neural networks, k-nearest neighbours classifiers [38]). One can also see that the performance of the FFNDL network keeps increasing if more training images are provided. In contrast, the eClass1 reaches its maximum accuracy after 40000 training images being processed. TEDAClass reaches its maximum accuracy after 30000 images being processed, with more training images, the accuracy of the TEDAClass decreases. For the FFNDL network, with more training samples and time consumptions, one can always obtain a higher accuracy.

Table 34 shows that the training time consumption grows with the amount of training dataset. Combining Table 33 and Table 34 one can see that it only takes 113.4 seconds (using WIN10 OS and MATLAB and no parallelisation) to train the network using 4000 images and the classification accuracy has already achieved over 95%. In addition, moving to Linux and C or Python can further speed up to an order of magnitude. For the published algorithms based on the global features (i.e. GIST and Haar), it already takes a larger amount of time (220.7 seconds) to only extract the GIST features from 4000 images. The Neocognitron neural network failed to give the consistent result as the network has a large number of parameters and the training process for 5000 training images takes more than 5 hours [271], [272].

In addition, the FFNDL network supports parallel processing. The computation can be distributed to a number of processers, which largely reduces the amount of time consumed by the training process. As it is presented in Table 34, by distributing the computation to local workers, the training process becomes much faster. It has to be stressed that, this parallelisation experiments are not real parallel computation as all the training is still conducted within a single dual core PC. With more processers, or using GPUs, the training process will be even faster, and, critically, this algorithm allows parallelisation at many levels.

B. Image classification

Table 35. Experimental results of the FFNDL network on image classification

Dataset	Training images per class	Testing images per class	Classification errors	Error rate	Time consumption
Human action recognition	60	40	5	2.1%	39.2s
	80	20	2	1.7%	48.7s
Object recognition	25	15	6	5%	30.6s
	30	10	1	1.3%	34.5s

In the following two numerical experiments for image classification, namely, human action recognition and object classification, d is set to be 32. The experimental results of the FFNDL network obtained from the two problems are presented in Table 35, which demonstrates that the FFNDL network can be applied in different areas and is able to perform highly accurate classification results even if using a small amount of training images.

6.3.2. Deep Rule-Based System

6.3.2.1. Benchmark Problems for Evaluation

To illustrate the performance of the proposed DRB classifier, the following four different challenging problems are considered:

A. Handwritten digits recognition

MNIST dataset [213] is used for evaluating the performance of the DRB systems on handwritten digits recognition. The details of this dataset have been given in section 5.4.1.

B. Face recognition

The one of the most widely used benchmark dataset for face recognition, database of faces [273] is used for evaluation. This dataset contains 40 subjects with 10 different images taken with different illumination, angle, face expression and face details (glasses/no glasses, mustaches, etc.). The size of each image is 92×112 pixels, with 256 grey levels per pixel. The examples of the database of faces are given in Figure 63.

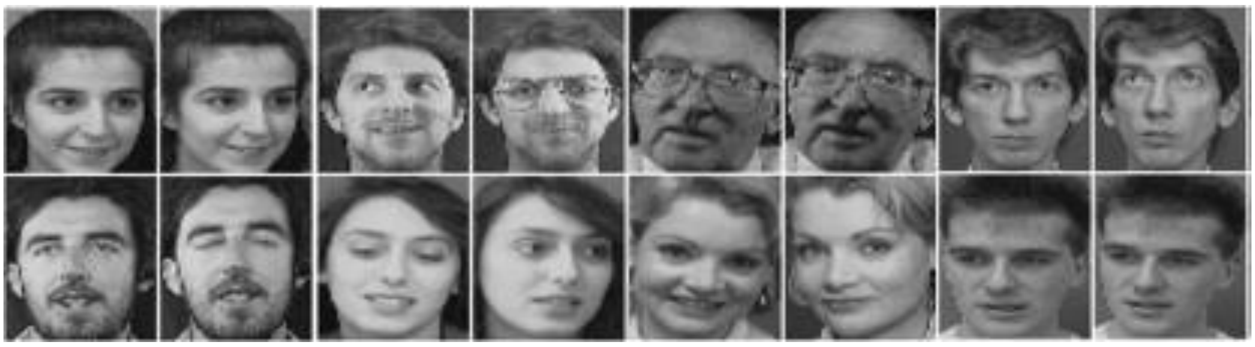


Figure 63. Examples of images from the database of faces.

C. Remote sensing

The first dataset from the remote sensing area is the Singapore dataset [274]. This dataset was constructed from a large satellite image of Singapore. This dataset consists of 1086 images with 256×256 pixels size with nine scene categories: 1) airplane, 2) forest, 3) harbor, 4) industry, 5) meadow, 6) overpass, 7) residential, 8) river, and 9) runway. Examples of the images of the nine classes are shown in Figure 64.

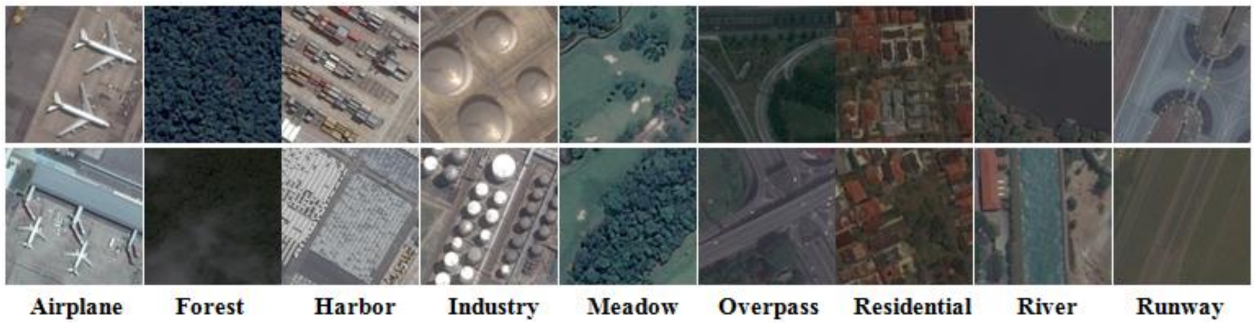


Figure 64. Examples of images from Singapore dataset.

The second dataset is the UCMerced dataset [211], which consists of fine spatial resolution remote sensing images of 21 challenging scene categories (including airplane, beach, building, etc.). Each category contains 100 images of the same image size (256×256 pixels). The example images of the 21 classes are shown in Figure 65.

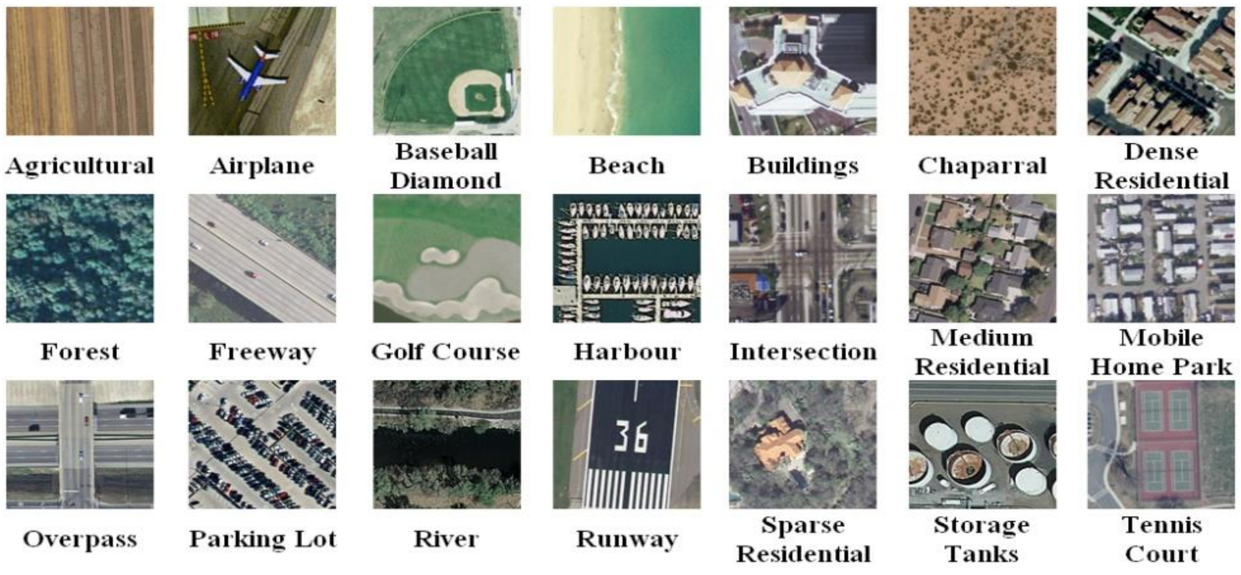


Figure 65. Example Images from UCMerced dataset.

D. Object recognition

The well-known Caltech 101 dataset [212] is used for evaluating the performance of the DRB system on object recognition. This dataset contains 9144 pictures of objects belonging to 101 categories and one background categories. The number of images in each class varies from 33 to 800 images per category. The size of each image is roughly 300×200 pixels. This data set contains both classes corresponding to rigid object (like bikes and cars) and classes corresponding to non-rigid object (like animals and flowers). Therefore, the shape variance is significant. The examples of this dataset are presented in Figure 66.

Figure 66. Example images of Caltech 101 dataset.

As the five benchmark datasets are very different from each other, five different, but same as in the publications [76], [213], [274], [275], experimental protocols will be used for the five datasets correspondingly.

6.3.2.2. Performance Evaluation and Discussion

A. Handwritten digits recognition

For the MNIST dataset, the DRB ensemble as presented in section 5.4.1 is used. During the experiment, the feature descriptor used by the DRB ensemble is GIST, HOG or the combined GIST and HOG (CGH) features. However, due to the different descriptive abilities of these features, the performance of the DRB ensemble is somewhat different. The recognition accuracy of the proposed DRB classifier using different feature descriptors is tabulated in Table 36. The corresponding average training times for the 10 fuzzy rules are tabulated in Table 37. By further combining the DRB ensemble trained with GIST features and the DRB ensemble trained with HOG features, it achieve a better recognition performance, which is tabulated in Table 36 as well. The DRB cascade [42] as described in section 5.4.2 is able to achieved the best performance, which is also presented in Table 36. The SVM conflict resolution classifier only applies to a small number (about 5%) of the validation data for which the decision-maker was not certain (there were two possible winners with close overall scores). By using the SVM conflict resolution classifier, the accuracy of the DRB cascade increases 0.11% [42], which is small but critical because it allows the DRB approach to outperform the current best alternative approach [72] (without using elastic distortion).

One of the most distinctive advantages of the DRB system is its evolving ability, which means that there is no need for complete re-training the classifier when new data samples are coming. To illustrate this advantage, the DRB classifier is trained with images in the form of an image stream, meanwhile, the execution time and the recognition accuracy are recorded during the process. In this example, the original training set without rescaling or rotation is used, which speeds up the process significantly. The relationship curves of the training time (the average for each of the 10 fuzzy rules) and recognition accuracy with the growing amount of the training samples are depicted in Figure 67.

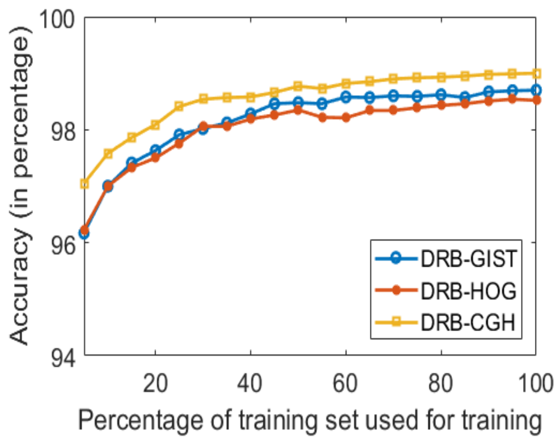
Table 36. Comparison between the DRB ensembles and the state-of-the-art approaches

Algorithm	DRB-GIST	DRB-HOG	DRB-CGH	DRB-GIST + DRB-HOG	DRB Cascade
Accuracy	99.30%	98.86%	99.32%	99.44%	99.55%
Training Time	Less than 2 minute for each part				
PC-Parameters	Core i7-4790 (3.60GHz), 16 GB DDR3				
GPU Used	None				
Elastic Distortion	No				
Tuned Parameters	No				
Iteration	No				
Randomness	No				
Parallelisation	Yes				
Evolving Ability	Yes				

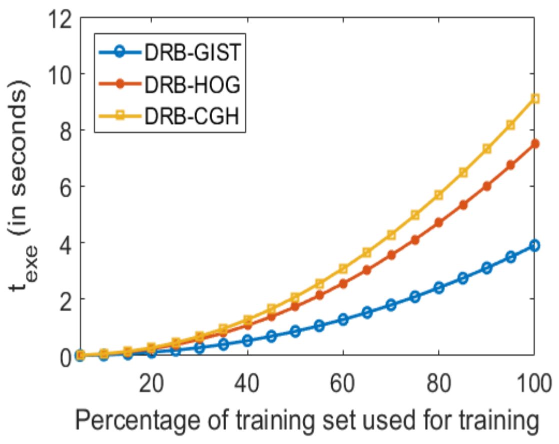
Algorithm	Large Convolutional Networks [276]	Large Convolutional Networks [72]	Committee of 7 Convolutional Neural Networks [24]	Committee of 35 Convolutional Neural Networks [22]
Accuracy	99.40%	99.47%	99.73%$\pm$ 2%	99.77%
Training Time	No Information	No Information	Almost 14 hours for each one of the DNNs.	
PC-Parameters			Core i7-920 (2.66GHz), 12 GB DDR3	
GPU Used			2 $\times$ GTX 480 & 2 $\times$ GTX 580	
Elastic Distortion	No	No	Yes	
Tuned Parameters	Yes	Yes	Yes	
Iteration	Yes	Yes	Yes	
Randomness	Yes	Yes	Yes	
Parallelisation	No	No	No	
Evolving Ability	No	No	No	

Table 37. Computation time for the learning process per sub-system (in seconds)

Fuzzy Rule #		1	2	3	4	5
Digital		“0”	“1”	“2”	“3”	“4”
Feature	GIST	39.26	32.39	41.95	45.72	37.17
	HOG	72.03	70.99	82.47	92.73	73.46
	CGH	96.54	88.93	99.21	113.52	91.53
Fuzzy Rule #		6	7	8	9	10
Digital		“5”	“6”	“7”	“8”	“9”
Feature	GIST	34.90	37.36	35.89	42.99	36.90
	HOG	67.53	68.48	77.93	75.83	69.90
	CGH	85.19	91.92	89.12	104.08	92.26



(a) Accuracy



(b) Training time

Figure 67. The relationship curve of training time and recognition accuracy with different amount of training samples.

In order to evaluate the performance of the DRB system, the state-of-the-art approaches reporting the current best and the second best results (with and without elastic distortion) [24], [72] are also reported in Table 36.

As one can see, the approaches reported in [22], [24] using elastic distortion can achieve slightly better results than the approaches in [72], [276] as well as the DRB systems. However, this comes at a price of using elastic distortion. This kind of distortion exhibits a significant randomness that may turn an unrecognizable digit into a recognizable one and vice versa, which also casts doubt on the effectiveness of the approaches in real applications. In addition, elastic distortion puts in question the achieved results’ repeatability and requires a cross-validation that further obstructs the online applications and the reliability of the results as discussed in [41].

Without using elastic distortion, the current published best result is 99.47% [72], which is comparable with the DRB ensemble, but worse than the DRB cascade [42]. However, one needs to notice that the convolution networks require a large number of parameters to be tuned, and cannot start “from scratch” nor evolve with the data stream and are not interpretable.

B. Face recognition

The architecture of the DRB classifier for face recognition does not include scaling and rotation, which is shown in Figure 68. In this test, the DRB classifier consists of the following layers:

1) Normalization layer;

2) Segmentation layer that splits each image into smaller pieces by a 22×32 size sliding window with the step size of 5 pixels in both horizontal and vertical directions (this setting is obtained empirically through experiments). The segmentation layer cuts one image into 255 pieces;

3) Feature descriptor, which extracts the combined GIST and HOG features from each segment;

4) FRB system, which consists of 40 fuzzy rules trained based on the segments of the 40 subjects’ images (one rule per subject);

5) Decision-maker, which generates the labels using equation (5.16).

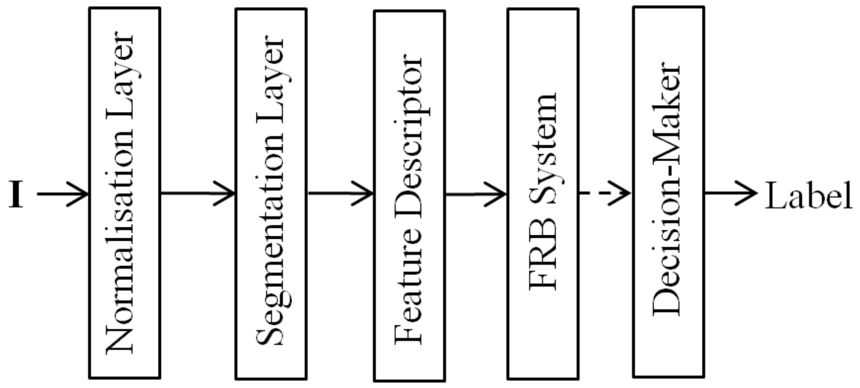


Figure 68. Architecture of the DRB classifier for face recognition.

Following the commonly used experimental protocol [275], for each subject, k images are randomly selected for training and 1 image for testing. The experiment was repeated 50 times and the average recognition accuracy of the DRB classifier with different k ($k = 1$ to 5) is tabulated in Table 38, and the DRB classifier is compared with the state-of-the-art approaches [275], [277]–[279] as follows:

1) Regularized Shearlet Network (RSN) [277];

2) Sparse Fingerprint Classification (SFC) [275];

3) Adaptive Sparse Representation (ASR) [278];

4) Sparse Discriminant Analysis via Joint $L_{2,1}$-norm Minimization ($SDAL_{21}M$) [279].

Table 38. Comparison between the DRB classifier and the-state-of-the-art approaches

k	Method	Accuracy (%)
1	RSN	88
	SFC	89
	DRB	**90**
2	ASR	82
	SFC	96
	DRB	**97**
3	ASR	89
	$SDAL_{21}M$	82
	SFC	98
	DRB	**99**
4	ASR	93
	SFC	**99**
	DRB	**99**
5	ASR	96
	$SDAL_{21}M$	93
	SFC	**100**
	DRB	**100**

One can see from Table 38 that the DRB classifier can achieve higher recognition accuracy with a smaller amount of training samples. For a better illustration, examples of the AnYa type fuzzy rules extracted during experiments are given in Table 39, where the segments are enlarged for visual clarity. These segments in Table 39 are the visual prototypes of the fuzzy rules, and thanks to them, one can always check the learning results obtained by the DRB classifier intuitionistically and make necessary modification for a better performance by adding, removing or exchaning prototypes. This is much simplier than tuning DCNNs, which contain hundreds of millions of parameters that are not interpretable to human.

The recognition accuracy of the DRB classifier and the average corresponding time consumption for each fuzzy rule in the training process with different amount of training samples is tabulated in Table 40. One can see that, the training process is very efficient. The proposed classifier can be trained for less than 3 seconds and achieve 100% accuracy in face recognition of individuals.

Table 39. Visual examples of the AnYa type fuzzy rules

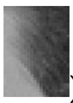
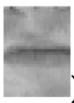
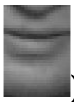
Fuzzy Rules
IF **(Sg ~**  **)** ***OR*** **(Sg ~** **)** ***OR … OR*** **(Sg ~** **)** ***OR*** **(Sg ~**  **)** ***THEN*** **(** **)**
IF **(Sg ~** **)** ***OR*** **(Sg ~** **)** ***OR … OR*** **(Sg ~** **)** ***OR*** **(Sg ~** **)** ***THEN*** **(** **)**
IF **(Sg ~** **)** ***OR*** **(Sg ~** **)** ***OR … OR*** **(Sg ~** **)** ***OR*** **(Sg ~** **)** ***THEN*** **(** **)**
IF **(Sg ~** **)** ***OR*** **(Sg ~** **)** ***OR … OR*** **(Sg ~** **)** ***OR*** **(Sg ~**  **)** ***THEN*** **(** **)**

Table 40. Results with different amount of training samples

k	1	2	3	4	5	6	7	8	9
Accuracy (%)	90	97	99	99	100	100	100	100	100
$\mathrm{t}_{\mathrm{exe}}$ (in seconds)	0.11	0.48	1.03	1.81	2.84	4.14	5.69	8.32	10.65

C. Remote sensing

The architecture of the DRB classifier for the remote sensing problems has been given in Figure 41. More specifically, in the numerical examples based on Singapore dataset, the sliding window with the window size of $(6 \times 6)/(8 \times 8)$ of image size and step size of $2/8$ width in the horizontal and $2/8$ length in the vertical direction is used (this setting is obtained empirically through experiments).

Following the commonly used experimental protocol [274], the DRB classifier is trained with randomly selected 20% of the images of each class and use the remainder as a testing set. The experiment is repeated 5 times and the average accuracy is reported in Table 41.

The performance of the DRB is compared with the state-of-the-art approaches as follows:

1) Transfer learning with deep representations (TLDP) [280];

2) Two-level feature representation (TLFP) [274];

3) Bag of visual words (BOVW) [145];

4) Scale-invariant feature transform with sparse coding (SIFTSC) [281];

5) Spatial pyramid matching kernel (SPMK) [282];

and the recognition accuracies of the comparative approaches are reported in Table 41 as well. One can see that, the DRB classifier is able to produce a significantly better recognition result than the best current methods.

Table 41. Comparison between the DRB classifier and the state-of-the-art approaches on Singapore dataset

Algorithm	Accuracy (%)
DRB-VGG	**97.70**
TLDP	82.13
TLFP	90.94
BOVW	87.41
SIFTSC	87.58
SPMK	82.85

To show the evolving ability of the DRB classifier, 20% of the images of each class are randomly selected for validation and the DRB is trained with 10%, 20%, 30%, 40%, 50%, 60%, 70% and 80% of the dataset. The experiment is repeated five times and the average accuracy is tabulated in Table 42. The average time for training is also reported, however, due to the unbalanced classes, the training time as tabulated in Table 42 is the overall training time of the 9 fuzzy rules.

Table 42. Results with different amount of training samples on the Singapore dataset

Ratio	10%	20%	30%	40%
Accuracy (%)	96.02	97.56	98.55	98.91
t_{exe} (in seconds)	5.1730	20.78	49.33	87.17
Ratio	50%	60%	70%	80%
Accuracy (%)	99.10	99.36	99.55	99.62
t_{exe} (in seconds)	135.00	195.57	270.89	346.14

For the UCMerced dataset, the DRB classifier with the same architecture as used in the previous example (Figure 42) is employed. Following the commonly used experimental protocol [274], 80% of the images of each class are randomly selected for training and the remainder is used as a testing set. The experiment is repeated five times, and the average accuracy is reported in Table 43. The performance of the DRB classifier is also compared with the state-of-the-art approaches as follows:

1) Two-level feature representation (TLFP) [274];

2) Bag of visual words (BoVW) [145];

3) Scale-invariant feature transform with sparse coding (SIFTSC) [281];

4) Spatial pyramid matching kernel (SPMK) [282], [283];

5) Multipath unsupervised feature learning (MUFL) [284];

6) Random convolutional network (RCNET) [79];

7) Linear SVM with pre-trained CaffeNet (SVM+Caffe) [140];

8) LIBLINEAR classifier with the VGG-VD-16 features (LIBL+VGG) [210];

9) Linear SVM with the VGG-VD-16 features (SVM+VGG).

Table 43. Comparison between the DRB classifier and the state-of-the-art approaches on UCMerced dataset

Algorithm	Accuracy	Algorithm	Accuracy
DRB	**96.14%**	MUFL	88.08%
TLFR	91.12%	RCNET	94.53%
BOVW	76.80%	SVM+ Caffe	93.42%
SIFTSC	81.67%	SVM+VGG	94.48%
SPMK	74.00%	LIBL+VGG	95.21%

Through the comparison in Table 43 one can see that, the DRB classifier, again, produced the best classification performance. Similarly, 20% of the images of each class are selected for validation and the DRB classifier is trained with 10%, 20%, 30%, 40%, 50%, 60% and 70% of the dataset. The experiment is repeated 5 times, and the average accuracy and time consumption for training (per rule) are tabulated in Table 44, where one can see that, the DRB classifier can achieve 95%+ classification accuracy with only less than 20 seconds' training for each fuzzy rule.

Furthermore, by creating an ensemble of the DRB classifier as described in section 5.4.3, the classification performance can be further improved to 97.10% [43].

Table 44. Results with different amount of training samples on the UCMerced dataset

Ratio	10%	20%	30%	40%
Accuracy (%)	83.48	88.57	90.80	92.19
t_{exe} (in seconds)	0.27	1.36	3.96	5.83
Ratio	50%	60%	70%	80%
Accuracy (%)	93.48	94.19	95.14	96.10
t_{exe} (in seconds)	10.29	11.52	15.49	18.15

D. Object recognition

The architecture of the DRB classifier for the object recognition is depicted in Figure 69 which is the same as the latter part of the DRB classifier for remote sensing problems as presented in Figure 41. The images of the Caltech 101 dataset [285] are very uniform in presentation, aligned from left to right, and usually not occluded, therefore, the rotation and segmentation are not necessary.

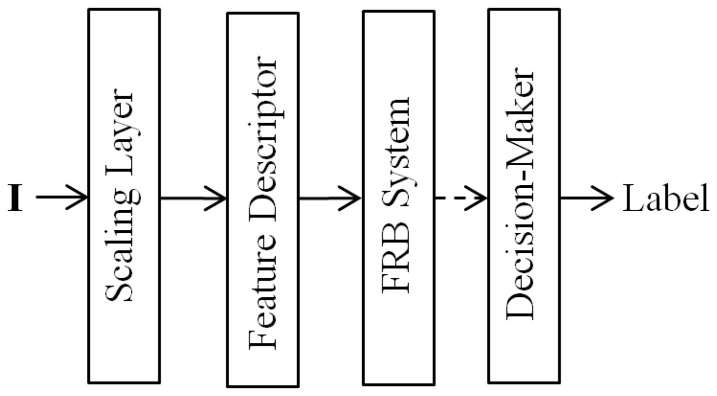


Figure 69. Architecture of the DRB classifier for object recognition.

Following the commonly used protocol [76], experiments are conducted by selecting 15 and 30 training images from each class for training and using the rest for validation. The experiment is repeated 5 times and the average accuracy is reported in Table 45. The DRB classifier is also compared with the state-of-the-art approaches as follows:

1) Convolutional deep belief network (CBDN) [286];

2) Learning convolutional feature hierarchies (CLFH) [287];

3) Deconvolutional networks (DECN) [288];

4) Linear spatial pyramid matching (LSPM) [289];

5) Local-constraint linear voding (LCLC) [290];

6) DEFEATnet [76];

7) Convolutional sparse autoencoders (CSAE) [291];

8) Linear SVM with the VGG-VD-16 features (SVM+VGG).

As one can see from Table 45, the DRB classifier easily outperforms all the comparative approaches in the object recognition problem. Same as the previous example, 1, 5, 10, 15, 20, 25, and 30 images of each class are selected for training the DRB classifier and use the rest for validation. The experiment is repeated 5 times, and the average accuracy and time consumption for training (per rule) are tabulated in Table 46, where one can see that, it only requires less than 2 seconds to train a single fuzzy rule.

Table 45. Comparison between the DRB classifier and the state-of-the-art approaches on Caltech 101 dataset

Algorithm	Accuracy (%)	
	15 Training	30 Training
DRB	**81.9**	**84.5**
CBDNET	57.7	65.4
CLFH	57.6	66.3
DECNNET	58.6	66.9
LSPM	67.0	73.2
LCLC	65.4	73.4
DEFEATnet	71.3	77.6
CSAE	64.0	71.4
SVM+VGG	78.9	83.5

Table 46. Results with different amount of training samples on the Caltech 101 dataset

Training Number	1	5	10	15	20	25	30
Accuracy (%)	61.1	76.4	80.4	81.9	83.5	83.6	84.5
t_{exe} (in seconds)		0.14	0.39	0.99	1.02	1.25	1.42

As one can see from the numerical examples presented in this subsection, the DRB classifier is able to offer extremely high classification accuracy comparable with human abilities on par or surpassing the best published mainstream deep learning alternatives. It is a general approach for various problems and serves as a strong alternative to the state-of-the-art approaches by providing a fully human-interpretable structure after a very fast (in orders of magnitude faster than the mainstream deep learning methods), transparent, nonparametric training process.

6.3.3. Semi-Supervised Deep Rule-Based Classifier

6.3.3.1. Benchmark Problems for Evaluation

In this section, the performance of SSDRB classifier is evaluated based on the following three challenging benchmark datasets:

1) UCMerced dataset [211];

2) Singapore dataset [274];

3) Caltech 101 dataset [212].

The details of the datasets have been given in the previous section. During the numerical experiments, the SSDRB classifier will not learn new classes, which means the algorithm for actively learning new classes (Figure 32) and Condition 30 (equation (5.19)) are not used.

6.3.3.2. Performance Evaluation and Discussion

First of all, the performance of the SSDRB classifier is investigated with the UCMerced dataset. Firstly, the influence of different values of φ on the performance of the SSDRB classifier is studied. Eight images are randomly picked out from each class as the labelled training set and the rest of the images are used as the unlabelled training set to continue to train the SSDRB classifier in both offline and online scenarios. In the online scenario, the semi-supervised learning is conducted on both sample-by-sample basis and chunk-by-chunk basis. For the former case, the order of the unlabelled images is descrambled randomly; while in the latter case, the unlabelled training samples are randomly divided into two chunks, which have exactly the same number of images. During this experiment, the value of φ varies from 1.05 to 1.30. The average number of classification errors of the SSDRB classifier on the unlabelled training set (1932 unlabelled images in total) is reported after 50 Monte Carlo experiments in Table 47. The average numbers of prototypes identified are reported in the same table. The performance of the DRB classifier is also reported as the baseline. The corresponding average accuracy of the SSDRB classifier with different values of φ is depicted in Figure 70.

Table 47. Performance of the SSDRB classifier with different values of φ

	φ	1.05	1.10	1.15	1.20	1.25	1.3
DRB	NE[a]	469.5					
	NP[b]	161.1					
Offline SSDRB	NE	423.8	413.7	417.1	429.1	436.2	450.5
	NP	1637	1402.8	1194.9	1015.8	874.7	759.1
Online SSDRB sample-by-sample	NE	483.3	457	450.9	453.1	459.0	462.9
	NP	1432.6	1192.8	1008.2	862.8	746.2	648.9
Online SSDRB chunk-by-chunk	NE	441.9	430.7	432.1	445.8	451.1	459.1
	NP	1581.6	1337.8	1127.7	960.1	825.1	712.6

[a] Number of errors (NE); [b] Number of prototypes (NP).

As one can see from Table 47 and Figure 70, the higher the value of φ is, the less prototypes the SSDRB classifier identified during the semi-supervised learning process, and, thus, the system structure is less complex and the computational efficiency is higher. However, at the same time, it is obvious that the accuracy of the classification results is not linearly correlated with the value of φ. There is a certain range of φ values for the SSDRB classifier to achieve the best accuracy. Trading off the overall performance and system complexity, the best range of φ values for the experiments performed is $[1.1,1.2]$. For the consistence, $\varphi = 1.2$ is used in the rest of the numerical examples in this section. However, one can also set different value for φ.

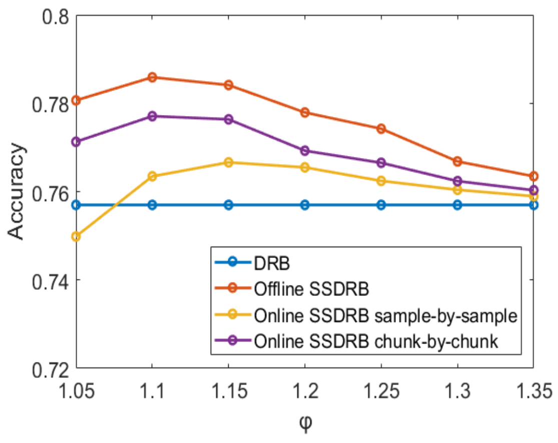


Figure 70. The average accuracy curve of the SSDRB classifier with different values of φ.

Secondly, $L = 1, 2, 3, \ldots, 10$ images are randomly picked out from each class as the labelled training images and the rest are used as the unlabelled ones to train the SSDRB classifier in both offline and online scenarios. Similar to the previous experiment, the semi-supervised learning is conducted on both sample-by-sample basis and chunk-by-chunk basis in the online scenario. The average numbers of classification errors of the SSDRB classifier on the unlabelled training set are reported after 50 Monte Carlo experiments in Table 47. The corresponding average accuracy of the DRB classifier with different number of labelled images is depicted in Figure 71.

Table 48. Performance of the SSDRB classifier with different values of L

	Number of Errors				
L	1	2	3	4	5
DRB	949.6	789.9	700.5	620.8	579.1
Offline SSDRB	887.7	724.8	627.8	566.5	521.0
Online SSDRB sample-by-sample	1010.5	805.6	683.2	605.2	561.3
Online SSDRB chunk-by-chunk	914.9	748.9	652.5	583.2	542.3
	Number of Errors				
L	6	7	8	9	10
DRB	530.5	502.9	469.5	448.4	425.6
Offline SSDRB	478.6	455.7	429.1	412.2	387.3
Online SSDRB sample-by-sample	517.5	485.4	453.1	435.1	412.8
Online SSDRB chunk-by-chunk	497.8	472.2	445.8	421.3	399.7

From Table 48 and Figure 71, one can see that, with $\varphi = 1.2$, the SSDRB classifier performs best in an offline scenario, which is due to the fact that, the DRB classifier is able to achieve an comprehensive understanding of the ensemble properties of the static image set. In the chunk-by-chunk learning mode, the DRB classifier can only study the ensemble properties of the unlabelled images within each chunk. And its performance deteriorates further if the semi-supervised learning is conducted on a sample-by-sample basis as each unlabelled training image is actually isolated from each other.

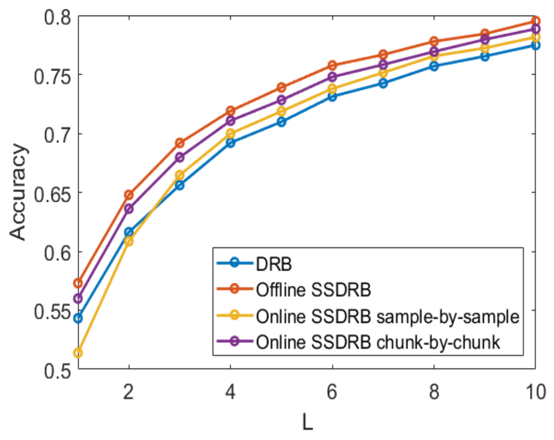


Figure 71. The average accuracy of the SSDRB classifier with different values of L.

In order to evaluate the performance of the SSDRB classifier, it is compared with the following well-known classification approaches:

1) SVM classifier [142];

2) KNN classifier [132].

The SVM (with linear kernel function) and KNN classifiers are the two main generic classifiers used in the transfer learning approaches based on per-trained DCNNs and are able to produce highly accurate classification results [77], [139]–[141]. As the DRB classifier presented in this thesis also involves the pre-trained DCNN as a feature descriptor, the two classifiers (SVM and KNN) are the most representative alternative approaches used for comparison.

The state-of-the-art semi-supervised approaches are also involved for comparison:

3) Laplacian support vector machine (LAPSVM) classifier [176], [177];

4) Local and global consistency (LGC) based semi-supervised classifier [165];

5) AnchorGraph-based semi-supervised classifier with kernel weights (AnchorK) [179];

6) AnchorGraphReg-based semi-supervised classifier with LAE weights (AnchorL) [179];

7) Greedy gradient Max-Cut (GGMC) based semi-supervised classifier [168].

There are also other well-known SVM- or graph-based semi-supervised approaches, i.e. Transductive SVM (TSVM) [173], $\nabla TSVM$[166] and Gaussian fields and harmonic functions based approaches [164]. However, previous work showed that the LAPSVM, LGC, GGMC and GGMC are, in general, able to produce more accurate classification results [168].

Therefore the comparison is only limited to the seven algorithms listed above. For the LapSVM, the "one versus all" strategy is used for all the benchmark problems.

Table 49. Comparison of the semi-supervised approaches on UCMerced dataset

	Number of Errors				
L	1	2	3	4	5
SSDRB	**887.7**	**724.8**	**627.8**	**566.5**	**521.0**
SVM	1322.5	1009.8	840.9	739.2	650.1
KNN	1186.7	1160.8	1075.3	979.4	932.6
LAPSVM	1624.6	1339.7	1134.7	920.8	800.3
LGC	1846.5	889.3	694.0	620.0	590.4
AnchorK	948.6	837.1	737.5	700.3	662.9
AnchorL	875.7	748.7	663.8	637.5	595.3
GGMC	1845.5	1032.9	829.6	772.8	701.6
	Number of Errors				
L	6	7	8	9	10
SSDRB	**478.6**	**455.7**	**429.1**	**412.2**	**387.3**
SVM	571.5	527.5	492.2	452.9	415.5
KNN	889.9	855.6	827.9	813.0	781.0
LAPSVM	701.9	626.5	552.1	499.9	463.7
LGC	556.6	552.6	544.8	537.7	533.9
AnchorK	631.9	609.1	575.9	553.0	534.3
AnchorL	568.0	540.5	515.7	480.0	473.0
GGMC	674.5	648.5	660.6	656.5	642.1

The experiment given in the previous example is repeated to test the performance of the seven comparative algorithms on the UCMerced dataset with different number of labelled training samples in an offline scenario. For a fair comparison, only the performance of the offline SSDRB classifier is considered. For the graph-based approaches, including KNN, LGC and GGMC, due to the very small number of labelled training samples, the value of k is set to be the same as L. All the free parameters of the semi-supervised approaches stay the same as the ones reported in the literature [165], [168], [176], [179]. The comparison results in terms of number of errors on the $(2100 - 21L)$ unlabelled training images are tabulated in Table 49. The accuracy curves are presented in Figure 72. All the reported results are the average after 50 Monte Carlo experiments.

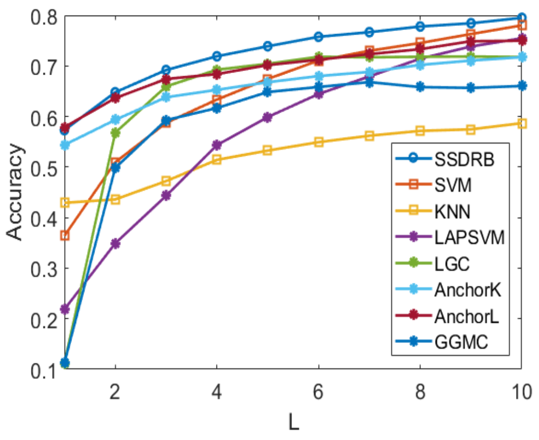


Figure 72. Accuracy comparison of the semi-supervised approaches on UCMerced dataset.

For the Singapore dataset, $L = 1, 2, 3, \ldots, 10$ images from each class is randomly selected out as the labelled training images and the rest are used as unlabelled ones to train the SSDRB classifier in an offline scenario. Its performance is also compared with the seven algorithms listed above. The average numbers of classification errors on the $(1086 - 9L)$ unlabelled training images are tabulated in Table 50. The accuracy curves are presented in Figure 73. All the reported results are the averages after 50 Monte Carlo experiments.

For the Caltech 101 image dataset, the commonly used experimental protocol is followed by randomly picking out 30 images from each class as the training set. Then, similarly, $L = 1, 2, 3, \ldots, 5$ images from each class are randomly picked out as labelled training images and the rest are used as unlabelled ones to train the SSDRB classifier in an offline scenario. Then, its performance is compared with the seven algorithms listed above. 50 Monte Carlo experiments are conducted and the average numbers of classification errors on the $(3030 - 101L)$ unlabelled training images are reported in Table 51. The accuracy curves are presented in Figure 74.

From Table 49, Table 50, Table 51, Figure 72, Figure 73 and Figure 74 one can see that, the SSDRB classifier is able to provide highly accurate classification results with only a very small number of labelled training images. It consistently outperforms all the seven comparative classification algorithms (both the most widely used ones and the "state-of-the-art" semi-supervised ones) in all the three popular benchmarks in the field of computer vison.

Moreover, compared with the existing semi-supervised approaches, the unique advantages of the SSDRB classifier thanks to its prototype-based nature include: 1) supporting online training and 2) classifying out-of-sample images. These are also noticeable from the numerical examples in this section. Therefore, one can conclude that the SSDRB classifier is a strong alternative to the existing approaches.

Table 50. Comparison of the semi-supervised approaches on Singapore dataset

	Number of Errors				
L	1	2	3	4	5
SSDRB	**887.7**	**724.8**	**627.8**	**566.5**	**521.0**
SVM	1322.5	1009.8	840.9	739.2	650.1
KNN	1186.7	1160.8	1075.3	979.4	932.6
LAPSVM	1624.6	1339.7	1134.7	920.8	800.3
LGC	1846.5	889.3	694.0	620.0	590.4
AnchorK	948.6	837.1	737.5	700.3	662.9
AnchorL	**875.7**	748.7	663.8	637.5	595.3
GGMC	1845.5	1032.9	829.6	772.8	701.6
	Number of Errors				
L	6	7	8	9	10
SSDRB	**62.0**	**56.1**	**48.0**	**49.0**	**46.3**
SVM	138	120.6	108.1	113.2	93.2
KNN	288.6	259.9	260.3	248.2	231.9
LAPSVM	391.6	335	280.6	247.7	209.4
LGC	84.0	84.2	86.0	84.4	91.8
AnchorK	141.8	130.9	135.4	130.4	122.0
AnchorL	110.7	116.2	110.6	99.6	97.0
GGMC	187.0	196.8	217.7	204.3	172.7

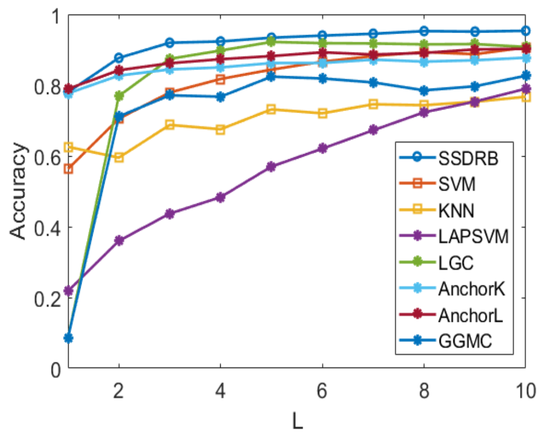


Figure 73. Accuracy comparison of the semi-supervised approaches on Singapore dataset.

Table 51. Comparison of the semi-supervised approaches on Caltech 101 dataset

	Number of Errors				
L	1	2	3	4	5
SSDRB	**1154.4**	**897.4**	**758.0**	**679.5**	**624.1**
SVM	1964.6	1474.9	1225.6	1035.0	909.4
KNN	2663.0	2329.0	2279.3	2120.9	1986.3
LAPSVM	1461.1	1161.3	951.7	796.3	705.1
LGC	2254.9	1008.5	826.8	737.8	683.8
AnchorK	1816.8	1521.9	1249.1	1062.1	940.4
AnchorL	1581.2	1308.2	1093.4	979.4	869.8
GGMC	2259.9	1071.3	854.0	767.1	716.2

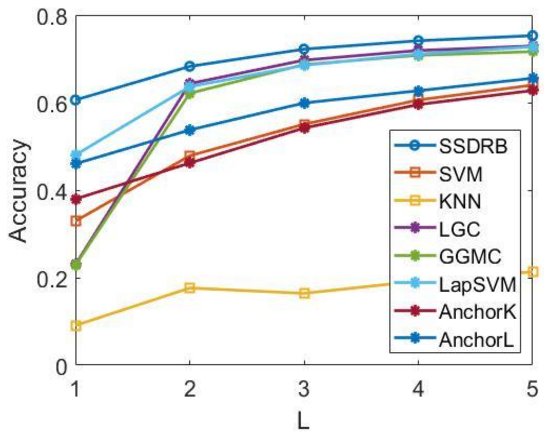


Figure 74. Accuracy comparison of the semi-supervised approaches on Caltech 101 dataset.

6.4. Conclusion

In this chapter, the numerous numerical examples based on challenging benchmark datasets are presented to demonstrate the validity and effectiveness of the self-organising transparent machine learning algorithms and deep learning systems. A number of state-of-the-art approaches are also involved for a better evaluation.

The experiment results show the strong performance of the proposed approaches compared with other approaches and also reveal their ability in the real applications.

7. Conclusion and Future Work

7.1. Key Contribution

This research consists of three main topics: 1) unsupervised self-organising machine learning techniques; 2) supervised self-organising machine learning techniques and 3) transparent self-organising deep learning networks, all of which are developed on the theoretical basis of the Empirical Data Analytics framework and AnYa type fuzzy rule-based system.

The works described in this thesis serve as powerful alternatives to the traditional data analysis, computational intelligence and machine learning methodologies:

1) Four different novel clustering/data partitioning algorithms are proposed, which are autonomous, self-organising, nonparametric and free from prior assumptions as well as user- and problem- specific parameters.

In contrast with the state-of-the-art clustering approaches, the proposed clustering/data partitioning algorithms do not impose any data generation models on the data as a priori. They are driven by the empirically observed data and are able to produce the objective results without the need of prior knowledge of the problems. In addition, they are highly efficient and suitable for large-scale static/streaming data processing.

2) Four novel approaches for regression, classification and anomaly detection are proposed, which share the same advantages of the unsupervised machine learning techniques proposed in this thesis thanks to the merits of the nonparametric EDA operators.

Without relying on the predefined free parameters and assumptions, the presented supervised self-organising learning algorithms are able to produce strong, objective results on various problems. With the ability of self-organising and self-evolving, these approaches are very suitable for real time streaming data processing.

3) Self-organising transparent deep learning networks with human-level performance and interpretable structure are proposed as the alternative to the popular deep learning models of the black-box type.

Traditional deep learning approaches are able to achieve very high performance on many problems; however, the lack of transparency and interpretability is one of the major drawbacks preventing them to be widely applied. The deep learning networks presented in this thesis, however, have a prototype-based nature and a self-organising and self-evolving structure. They are able to demonstrate very high performance on the image classification problems currently with a fully transparent, highly efficient and parallelisable learning process, which can be very powerful and attractive in real applications. The semi-supervised learning strategy allows the introduced deep learning networks to learn from very little training images while exhibit very high accurate classification results and to learn new knowledge actively without supervision by human experts.

7.2. Future Work Plans

The following directions are to be considered in the future for improvement of the machine learning algorithms and deep learning systems:

A. Unsupervised self-organising machine learning techniques

1) The optimality of the proposed clustering/data partitioning algorithms needs to be investigated, which is of great importance for real applications as well as for the research purposes. The optimality of the solution is the proof of the validity and effectiveness of a learning algorithm.

2) The performance of the clustering/data partitioning algorithms, including the ones presented in this thesis, is more or less subjective to the choice of distance metric/dissimilarity. The question of when to use which type of distance metric/dissimilarity requires a careful study. One possible approach is to conduct a systematic investigation on the differences in the behaviours of different distance metrics/dissimilarities in the real data space.

3) The choice of the most suitable clustering/data partitioning algorithm is always problem-specific. However, it will be of great interest to carefully study the advantages and deficiencies of each algorithm, depending on which one can always select the most suitable algorithm for a given problem.

4) New learning algorithms that are not only free from the prior assumptions and user- and problem-specific parameters, but also free from the influence of distance metrics/dissimilarity can be very useful for investigating the data pattern objectively.

5) A new data partitioning algorithm that is free from hard-coded mathematical rules can be very helpful for different applications.

B. Supervised self-organising machine learning techniques

1) The stability of the first order autonomous learning multi-model system needs to be investigated and proven. Stability analysis of a learning system is of paramount importance for real-world applications and provides the theoretical guarantees for the convergence.

2) The zero order autonomous learning multi-model system can be further improved by introducing dynamically changing threshold derived from data directly.

3) The sensitivity of the learning systems to the different experimental setting requires further study.

4) The computational complexity analysis of the learning systems and statistical analysis of the numerical results need to be done in the future, which allow a better understanding of the properties of the proposed learning systems.

5) The online version of autonomous anomaly detection algorithm can be very useful for the fault detection in data streams. In the real-world applications, new data often continuously arrives in a form of a stream. Identifying the anomalies from the stream in real-time is critical for identifying faults in an earlier stage, which may prevent a serious accident that may happen in the near future.

C. Transparent self-organising deep learning networks

1) The deep rule-based systems presented in this thesis employ the pre-trained deep convolutional neural network as the feature descriptor without any tuning or modification. However, the performance of the systems can be further improved if proper tuning is involved.

2) The deep rule-based systems can be extended to learn from images that contain multiple sub-regions of different classes, which could be of great importance for understanding the semantic meaning of the images.

References

[1] T. Bayes, “An essay towards solving a problem in the doctrine of chances,” *Philos. Trans. R. Soc.*, vol. 53, p. 370, 1763.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed. Chichester, West Sussex, UK,: Wiley-Interscience, 2000.

[3] C. M. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006.

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

[5] J. B. MacQueen, “Some methods for classification and analysis of multivariate observations,” *5th Berkeley Symp. Math. Stat. Probab.*, vol. 1, no. 233, pp. 281–297, 1967.

[6] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *International Conference on Knowledge Discovery and Data Mining*, 1996, vol. 96, pp. 226–231.

[7] A. Lemos, W. Caminhas, and F. Gomide, “Adaptive fault detection and diagnosis using an evolving fuzzy classifier,” *Inf. Sci. (Ny).*, vol. 220, pp. 64–85, 2013.

[8] M. Pratama, S. G. Anavatti, and E. Lughofer, “Evolving fuzzy rule-based classifier based on GENEFIS,” *IEEE Int. Conf. Fuzzy Syst.*, 2013.

[9] P. Angelov and X. Zhou, “Evolving fuzzy systems from data streams in real-time,” in *2006 International Symposium on Evolving Fuzzy Systems*, 2006, pp. 29–35.

[10] R. Hyde and P. Angelov, “A fully autonomous data density based clustering technique,” in *IEEE Symposium on Evolving and Autonomous Learning Systems*, 2014, pp. 116–123.

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

[12] R. Dutta Baruah and P. Angelov, “Evolving local means method for clustering of streaming data,” *IEEE Int. Conf. Fuzzy Syst.*, pp. 10–15, 2012.

[13] R. R. Yager and D. P. Filev, “Generation of fuzzy rules by mountain clustering,” *J. Intell. Fuzzy Syst.*, vol. 2, no. 3, pp. 209–219, 1994.

[14] S. L. Chiu, “A cluster extension method with extension to fuzzy model\nidentification,” *Proc. 1994 IEEE 3rd Int. Fuzzy Syst. Conf.*, pp. 1240–1245, 1994.

[15] S. L. Chiu, “Fuzzy model identification based on cluster estimation,” *J. Intell. Fuzzy Syst.*, vol. 2, no. 3, pp. 267–278, 1994.

[16] K. Fukunaga and L. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition,” *IEEE Trans. Inf. Theory*, vol. 21, no. 1, pp. 32–40, 1975.

[17] C. Wang, S. Member, J. Lai, and D. Huang, “SVStream : A Support Vector Based Algorithm for Clustering Data Streams,” *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 1410–1424, 2011.

[18] P. Angelov and A. Kordon, “Adaptive inferential sensors based on evolving fuzzy models,” *IEEE Trans. Syst. Man, Cybern. - Part B Cybern.*, vol. 40, no. 2, pp. 529–539, 2010.

[19] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nat. Methods*, vol. 13, no. 1, pp. 35–

35, 2015.

[20] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, “Multimodal Deep Learning,” *Proc. 28th Int. Conf. Mach. Learn.*, pp. 689–696, 2011.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances In Neural Information Processing Systems*, 2012, pp. 1097–1105.

[22] D. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

[23] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations*, 2015, pp. 1–14.

[24] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Convolutional neural network committees for handwritten character classification,” in *International Conference on Document Analysis and Recognition*, 2011, vol. 10, pp. 1135–1139.

[25] P. P. Angelov, X. Gu, J. Principe, and D. Kangin, “Empirical data analysis - a new tool for data analytics,” in *IEEE International Conference on Systems, Man, and Cybernetics*, 2016, pp. 53–59.

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

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

[28] P. P. Angelov, X. Gu, G. Gutierrez, J. A. Iglesias, and A. Sanchis, “Autonomous data density based clustering method,” in *IEEE World Congress on Computational Intelligence*, 2016, pp. 2405–2413.

[29] X. Gu and P. P. Angelov, “Autonomous data-driven clustering for live data stream,” in *IEEE International Conference on Systems, Man, and Cybernetics*, 2016, pp. 1128–1135.

[30] X. Gu, P. P. Angelov, G. Gutierrez, J. A. Iglesias, and A. Sanchis, “Parallel computing TEDA for high frequency streaming data clustering,” in *INNS Conference on Big Data*, 2016, pp. 238–253.

[31] X. Gu, P. P. Angelov, and J. C. Principe, “A method for autonomous data partitioning,” *Inf. Sci. (Ny).*, vol. 460–461, pp. 65–82, 2018.

[32] X. Gu, P. Angelov, D. Kangin, and J. Principe, “Self-organised direction aware data partitioning algorithm,” *Inf. Sci. (Ny).*, vol. 423, pp. 80–95, 2018.

[33] X. Gu, P. P. Angelov, D. Kangin, and J. C. Principe, “A new type of distance metric and its use for clustering,” *Evol. Syst.*, vol. 8, no. 3, pp. 167–178, 2017.

[34] P. P. Angelov, X. Gu, and J. C. Principe, “Autonomous learning multi-model systems from data streams,” *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 4, pp. 2213–2224, 2018.

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

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

36–51, 2018.

[37] X. Gu and P. Angelov, “Autonomous anomaly detection,” in *IEEE Conference on Evolving and Adaptive Intelligent Systems*, 2017, pp. 1–8.

[38] P. Angelov, X. Gu, and J. Principe, “Fast feedforward non-parametric deep learning network with automatic feature extraction,” in *International Joint Conference on Neural Networks*, 2017, pp. 534–541.

[39] P. P. Angelov and X. Gu, “Deep rule-based classifier with human-level performance and characteristics,” *Inf. Sci. (Ny).*, vol. 463–464, pp. 196–213, 2018.

[40] X. Gu and P. P. Angelov, “Semi-supervised deep rule-based approach for image classification,” *Appl. Soft Comput.*, vol. 68, pp. 53–68, 2018.

[41] P. P. Angelov and X. Gu, “MICE: Multi-layer multi-model images classifier ensemble,” in *IEEE International Conference on Cybernetics*, 2017, pp. 436–443.

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

[43] X. Gu, P. P. Angelov, C. Zhang, and P. M. Atkinson, “A massively parallel deep rule-based ensemble vlassifier for remote sensing scenes,” *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 3, pp. 345–349, 2018.

[44] G. Grimmett and D. Welsh, *Probability: an introduction*. Oxford University Press, 2014.

[45] R. C. Tolman, *The principles of statistical mechanics*. Courier Corporation, 1938.

[46] A. N. Kolmogorov, *Foundations of the theory of probability*. England: Chelsea: Oxford, 1950.

[47] V. Vapnik and R. Izmailov, “Statistical inference problems and their rigorous solutions,” *Stat. Learn. Data Sci.*, vol. 9047, pp. 33–71, 2015.

[48] P. Angelov, “Outside the box: an alternative data analytics framework,” *J. Autom. Mob. Robot. Intell. Syst.*, vol. 8, no. 2, pp. 53–59, 2014.

[49] P. P. Angelov, “Anomaly detection based on eccentricity analysis,” in *2014 IEEE Symposium Series in Computational Intelligence, IEEE Symposium on Evolving and Autonomous Learning Systems, EALS, SSCI 2014*, 2014, pp. 1–8.

[50] P. Angelov, “Typicality distribution function – a new density- based data analytics tool,” in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.

[51] D. Kangin, P. Angelov, and J. A. Iglesias, “Autonomously evolving classifier TEDAClass,” *Inf. Sci. (Ny).*, vol. 366, pp. 1–11, 2016.

[52] J. G. Saw, M. C. K. Yang, and T. S. E. C. Mo, “Chebyshev inequality with estimated mean and variance,” *Am. Stat.*, vol. 38, no. 2, pp. 130–132, 1984.

[53] L. A. Zadeh, “Fuzzy sets,” *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.

[54] P. P. Angelov and X. Gu, “Empirical fuzzy sets,” *Int. J. Intell. Syst.*, vol. 33, no. 2, pp. 362–395, 2017.

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

[56] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. L. Massart, “The Mahalanobis distance,”

Chemom. Intell. Lab. Syst., vol. 50, no. 1, pp. 1–18, 2000.

[57] B. McCune, J. B. Grace, and D. L. Urban, *Analysis of Ecological Communities*. 2002.

[58] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man. Mach. Stud.*, vol. 7, no. 1, pp. 1–13, 1975.

[59] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst. Man. Cybern.*, vol. 15, no. 1, pp. 116–132, 1985.

[60] P. Angelov and R. Yager, "A new type of simplified fuzzy rule-based system," *Int. J. Gen. Syst.*, vol. 41, no. 2, pp. 163–185, 2011.

[61] W. Pedrycz, "Fuzzy relational equations with generalized connectives and their applications," *Fuzzy Sets Syst.*, vol. 10, no. 1–3, pp. 185–201, 1983.

[62] P. Angelov, "An approach for fuzzy rule-base adaptation using on-line clustering," *Int. J. Approx. Reason.*, vol. 35, no. 3, pp. 275–289, 2004.

[63] C. C. Lee, "Fuzzy Logic in Control Systems : Fuzzy Logic Controller - Part 1," *IEEE Trans. Syst. Man Cybern.*, vol. 20, no. 2, pp. 404–418, 1990.

[64] 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.

[65] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Crambridge, MA: MIT Press, 2016.

[67] F. Gers, "Long short-term memory in recurrent neural networks," 2001.

[68] A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[69] N. K. Kasabov, "NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Networks*, vol. 52, pp. 62–76, 2014.

[70] K. Fukushima and S. Miyake, "Neocognitron: a self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets,* Springer Berlin Heidelberg, 1982, pp. 267–285.

[71] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[72] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *IEEE International Conference on Computer Vision,* 2009, pp. 2146–2153.

[73] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," *Doc. Anal. Recognition, 2003. Proceedings. Seventh Int. Conf.*, pp. 958–963, 2003.

[74] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep

reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[75] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2155–2162.

[76] S. Gao, L. Duan, and I. W. Tsang, "DEFEATnet—A deep conventional image representation for image classification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 3, pp. 494–505, 2016.

[77] K. Charalampous and A. Gasteratos, "On-line deep learning method for action recognition," *Pattern Anal. Appl.*, vol. 19, no. 2, pp. 337–354, 2016.

[78] T. Guha, S. Member, and R. K. Ward, "Sequential deep learning for human action recognition," *Hum. Behav. Underst.*, pp. 29--39, 2011.

[79] L. Zhang, L. Zhang, and V. Kumar, "Deep learning for remote sensing data," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 22–40, 2016.

[80] Y. Li, H. Zhang, X. Xue, Y. Jiang, and Q. Shen, "Deep learning for remote sensing image classification: a survey," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. e1264, 2018.

[81] H. Zhang, Y. Li, Y. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery using a dual-channel convolutional neural network," *Remote Sens. Lett.*, vol. 8, no. 5, pp. 438–447, 2017.

[82] Y. Li, H. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery with 3D convolutional neural network," *Remote Sens.*, vol. 9, no. 1, 2017.

[83] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, 1962.

[84] S. K. Pal, S. Bandyopadhyay, and S. S. Ray, "Evolutionary computation in bioinformatics: A review," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 36, no. 5, pp. 601–615, 2006.

[85] T. Back, U. Hammel, and H. P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 3–17, 1997.

[86] H. J. Bremermann, "Optimization through evolution and recombination," *Self-organizing Syst.*, vol. 93, p. 106, 1962.

[87] G. E. P. Box, "Evolutionary operation: a method for increasing industrial productivity," *Appl. Stat.*, vol. 6, no. 2, pp. 81–101, 1957.

[88] R. M. Friedberg, "A learning machine: part I," *IBM J. Res. Dev.*, vol. 2, no. 1, pp. 2–13, 1958.

[89] I. Rechenberg, "Cybernetic solution path of an experimental problem," *R. Aircr. Establ. Libr. Transl.*, vol. 1122, 1965.

[90] L. J. Fogel, "Autonomous automata," *Ind. Res.*, vol. 4, pp. 14–19, 1962.

[91] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[92] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *International Conference on Parallel Problem Solving From Nature*, Springer, Berlin, Heidelberg, 2000, pp. 849–858.

[93] D. Du, D. Simon, and M. Ergezer, "Biogeography-based optimization combined with

evolutionary strategy and immigration refusal," in *IEEE International Conference on Systems, Man and Cybernetics*, 2009, pp. 997–1002.

[94] J. R. Koza, "Genetic programming as a means for programming computers by natural selection," *Stat. Comput.*, vol. 4, no. 2, pp. 87–112, 1994.

[95] C. Zhang, H. Shao, and Y. Li, "Particle swarm optimisation for evolving artificial neural network," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2000, pp. 2487–2490.

[96] T. Chen, Q. Shen, P. Su, and C. Shang, "Fuzzy rule weight modification with particle swarm optimisation," *Soft Comput.*, vol. 20, no. 8, pp. 2923–2937, 2016.

[97] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Handbook of evolutionary computation*. CRC Press, 1997.

[98] O. Maimon and L. Rokach, *Data mining and knowledge discovery handbook*. Springer, Boston, MA, 2005.

[99] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[100] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.

[101] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *Computer (Long. Beach. Calif).*, vol. 32, no. 8, pp. 68–75, 1999.

[102] W. H. E. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *J. Classif.*, vol. 1, pp. 7–24, 1984.

[103] A. Gucnoche, P. Hansen, and B. Jaumard, "Efficient algorithms for divisive hierarchical clustering with the diameter criterion," *Joumal Classif.*, vol. 8, pp. 5–30, 1991.

[104] T. Xiong, S. Wang, A. Mayers, and E. Monga, "DHCC: Divisive hierarchical clustering of categorical data," *Data Min. Knowl. Discov.*, vol. 24, pp. 103–135, 2012.

[105] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science (80-.).*, vol. 315, no. 5814, pp. 972–976, 2007.

[106] S. Zhong, "Efficient online spherical k-means clustering," in *Proceedings of the International Joint Conference on Neural Networks*, 2005, pp. 3180–3185.

[107] L. B. Neuristique and Y. Bengio, "Convergence properties of the K-means algorithms," *Adv. Neural Inf. Process. Syst.*, pp. 585--592, 1995.

[108] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids," in *Statistical Data Analysis Based on the L1 Norm and Related Methods*, North-Holland, 1987, pp. 405–416.

[109] T. Kohonen, *Self-organizing maps*. Berlin: Springer, 1997.

[110] P. Płoński and K. Zaremba, "Self-organising maps for classification with metropolis-hastings algorithm for supervision," in *International Conference on Neural Information Processing*, 2012, pp. 149–156.

[111] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial-temporal data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, 2007.

[112] P. Viswanath and V. Suresh Babu, "Rough-DBSCAN: A fast hybrid density based clustering

method for large data sets," *Pattern Recognit. Lett.*, vol. 30, no. 16, pp. 1477–1488, 2009.

[113] S. Kisilevich, F. Mansmann, and D. Keim, "P-DBSCAN: A density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos," in *International Conference and Exhibition on Computing for Geospatial Research and Application*, 2010, pp. 1–4.

[114] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[115] K. L. Wu and M. S. Yang, "Mean shift-based clustering," *Pattern Recognit.*, vol. 40, no. 11, pp. 3035–3052, 2007.

[116] L. Abdallah and I. Shimshoni, "Mean Shift Clustering Algorithm for Data with Missing Values," in *International Conference on Data Warehousing and Knowledge Discovery*, 2014, pp. 426–438.

[117] E. Lughofer and P. Angelov, "Handling drifts and shifts in on-line data streams with evolving fuzzy systems," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 2057–2068, 2011.

[118] R. Hyde, P. Angelov, and A. R. MacKenzie, "Fully online clustering of evolving data streams into arbitrarily shaped clusters," *Inf. Sci. (Ny).*, vol. 382–383, pp. 96–114, 2017.

[119] A. Corduneanu and C. M. Bishop, "Variational Bayesian model selection for mixture distributions," *Proc. Eighth Int. Conf. Artif. Intell. Stat.*, pp. 27–34, 2001.

[120] C. A. McGrory and D. M. Titterington, "Variational approximations in Bayesian model selection for finite mixture distributions," *Comput. Stat. Data Anal.*, vol. 51, no. 11, pp. 5352–5367, 2007.

[121] D. M. Blei and M. I. Jordan, "Variational methods for the Dirichlet process," *Proc. twenty-first Int. Conf. Mach. Learn.*, p. 12, 2004.

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

[123] T. Kimura, T. Tokuda, Y. Nakada, T. Nokajima, T. Matsumoto, and A. Doucet, "Expectation-maximization algorithms for inference in Dirichlet processes mixture," *Pattern Anal. Appl.*, vol. 16, no. 1, pp. 55–67, 2013.

[124] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Comput. Geosci.*, vol. 10, no. 2–3, pp. 191–203, 1984.

[125] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *J. Cybern.*, vol. 3, no. 3, 1973.

[126] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *J. Cybern.*, vol. 4, no. 1, pp. 95–104, 1974.

[127] L. Fu and E. Medico, "FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data.," *BMC Bioinformatics*, vol. 8, no. 1, p. 3, 2007.

[128] S. Krinidis and V. Chatzis, "A robust fuzzy local information c-means clustering algorithm," *IEEE Trans. Image Process.*, vol. 19, no. 5, pp. 1328–1337, 2010.

[129] M. N. Murty and V. S. Devi, *Introduction to pattern recognition and machine learning*. World Scientific., 2015.

[130] S. T. K. Koutroumbas, *Pattern Recognition 4th Edition*. New York: Elsevier, 2009.

[131] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, 1967.

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

[133] K. Fukunage and P. M. Narendra, “A branch and bound algorithm for computing k-nearest neighbors,” *IEEE Trans. Comput.*, vol. C-24, no. 7, pp. 750–753, 1975.

[134] P. Horton and K. Nakai, “Better prediction of protein cellular localization sites with the k nearest neighbors classifier,” in *International Conference on Intelligent Systems for Molecular Biology*, 1997, pp. 147–152.

[135] H. Franco-Lopez, A. R. Ek, and M. E. Bauer, “Estimation and mapping of forest stand density, volume, and cover type using the k-nearest neighbors method,” *Remote Sens. Environ.*, vol. 77, no. 3, pp. 251–274, 2001.

[136] T. Seidl and H.-P. Kriegel, “Optimal multi-step k-nearest neighbor search,” *ACM SIGMOD Rec.*, vol. 27, no. 2, pp. 154–165, 1998.

[137] J. M. Keller and M. R. Gray, “A fuzzy k-nearest neighbor algorithm,” *IEEE Trans. Syst. Man Cybern.*, vol. 15, no. 4, pp. 580–585, 1985.

[138] E.-H. (Sam) Han, G. Karypis, and V. Kumar, “Text categorization using weight adjusted k-nearest neighbor classification,” in *Advances in Knowledge Discovery and Data Mining*, 2001, pp. 53–65.

[139] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European Conference on Computer Vsion*, 2014, pp. 818–833.

[140] A. B. Penatti, K. Nogueira, and J. A. Santos, “Do deep features generalize from everyday objects to remote sensing and aerial scenes domains ?,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 44–51.

[141] A. Avramovi and V. Risojevi, “Block-based semantic classification of high-resolution multispectral aerial images,” *Signal, Image Video Process.*, vol. 10, pp. 75–84, 2016.

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

[143] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, “Support vector machine classification and validation of cancer tissue samples using microarray expression data,” *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[144] W. Huang, Y. Nakamori, and S.-Y. Wang, “Forecasting stock market movement direction with support vector machine,” *Comput. Oper. Res.*, vol. 32, no. 10, pp. 2513–2522, 2005.

[145] T. Joachims, “Text categorization with support vector machines: learning with many relevant features,” in *European Conference on Machine Learning*, 1998, pp. 137–142.

[146] W. S. Noble, “What is a support vector machine?,” *Nat. Biotechnol.*, vol. 24, no. 12, pp. 1565–1567, 2006.

[147] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, 1998.

[148] V. Vapnik and A. Lerner, “Pattern recognition using generalized portrait method,” *Autom. Remote Control*, vol. 24, no. 6, pp. 774–780, 1963.

[149] D. Kangin and P. Angelov, “Recursive SVM based on TEDA,” in *International Symposium on Statistical Learning and Data Sciences*, 2015, pp. 156–168.

[150] K. Polat and S. Güneş, “Classification of epileptiform EEG using a hybrid system based on decision tree classifier and fast Fourier transform,” *Appl. Math. Comput.*, vol. 187, no. 2, pp. 1017–1026, 2007.

[151] W. Du, W. Du, Z. Zhan, and Z. Zhan, “Building decision tree classifier on private data,” in *The IEEE International Conference on Privacy, Security and Data Mining*, 2002, pp. 1–8.

[152] S. S. Sivatha Sindhu, S. Geetha, and A. Kannan, “Decision tree based light weight intrusion detection using a wrapper approach,” *Expert Syst. Appl.*, vol. 39, no. 1, pp. 129–141, 2012.

[153] L. Lu, L. Di, and Y. Ye, “A decision-tree classifier for extracting transparent plastic-mulched Landcover from landsat-5 TM images,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 7, no. 11, pp. 4548–4558, 2014.

[154] S. R. Safavian and D. Landgrebe, “A survey of decision tree classifier methodology,” *IEEE Trans. Syst. Man Cybern.*, vol. 21, no. 3, pp. 660–674, 1991.

[155] M. Learning, M. Learning, K. A. Publishers, K. A. Publishers, A. C. Sciences, A. C. Sciences, R. August, and R. August, “Induction of decision trees,” *Expert Syst.*, pp. 81–106, 2007.

[156] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen., *Classification and regression trees*. CRC press, 1984.

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

[158] J. R. Quinlan, *C4.5: programs for machine learning*. Elsevier, 2014.

[159] P. P. Angelov, “Evolving fuzzy rule-based models,” *J. Chinese Inst. Ind. Eng.*, vol. 17, no. 5, pp. 459–468, 2000.

[160] 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.

[161] P. Angelov, E. Lughofer, and X. Zhou, “Evolving fuzzy classifiers using different model architectures,” *Fuzzy Sets Syst.*, vol. 159, no. 23, pp. 3160–3182, 2008.

[162] R. D. Baruah, P. P. Angelov, and J. Andreu, “Simpl _ eClass : simplified potential-free evolving fuzzy rule-based classifiers,” in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2011, pp. 2249–2254.

[163] P. Angelov, D. Kangin, and D. Kolev, “Symbol recognition with a new autonomously evolving classifier AutoClass,” in *IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 2014, pp. 1–7.

[164] X. Zhu, Z. Ghahraman, and J. D. Lafferty, “Semi-supervised learning using gaussian fields and harmonic functions,” in *International conference on Machine learning*, 2003, pp. 912–919.

[165] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, “Learning with local and global consistency,” in *Adv. Neural. Inform. Process Syst*, 2004, pp. 321–328.

[166] O. Chapelle and A. Zien, “Semi-supervised classification by low density separation,” in *AISTATS*, 2005, pp. 57–64.

[167] M. Guillaumin, J. J. Verbeek, and C. Schmid, “Multimodal semi-supervised learning for image classification,” in *IEEE Conference on Computer Vision & Pattern Recognition*, 2010, pp. 902–909.

[168] J. Wang, T. Jebara, and S. F. Chang, “Semi-supervised learning using greedy Max-Cut,” *J. Mach. Learn. Res.*, vol. 14, pp. 771–800, 2013.

[169] A. Iwayemi and C. Zhou, “SARAA: Semi-Supervised Learning for Automated Residential Appliance Annotation,” *IEEE Trans. Smart Grid*, vol. 8, no. 2, p. 779, 2017.

[170] F. Wang, C. Zhang, H. C. Shen, and J. Wang, “Semi-supervised classification using linear neighborhood propagation,” in *IEEE Conference on Computer Vision & Pattern Recognition*, 2006, pp. 160–167.

[171] S. Xiang, F. Nie, and C. Zhang, “Semi-supervised classification via local spline regression,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 11, pp. 2039–2053, 2010.

[172] B. Jiang, H. Chen, B. Yuan, and X. Yao, “Scalable graph-based semi-supervised learning through sparse bayesian model,” *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2758–2771, 2017.

[173] J. Thorsten, “Transductive inference for text classification using support vector machines,” *Int. Conf. Mach. Learn.*, vol. 9, pp. 200–209, 1999.

[174] O. Chapelle, V. Sindhwani, and S. Keerthi, “Optimization techniques for semi-supervised support vector machines,” *J. Mach. Learn. Res.*, vol. 9, pp. 203–233, 2008.

[175] V. Sindhwani, P. Niyogi, and M. Belkin, “Beyond the point cloud: from transductive to semi-supervised learning,” in *International Conference on Machine Learning*, 2005, vol. 1, pp. 824–831.

[176] M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold regularization: a geometric framework for learning from labeled and unlabeled examples,” *J. Mach. Learn. Res.*, vol. 7, no. 2006, pp. 2399–2434, 2006.

[177] L. Gómez-Chova, G. Camps-Valls, J. Munoz-Mari, and J. Calpe, “Semisupervised image classification with Laplacian support vector machines,” *IEEE Geosci. Remote Sens. Lett.*, vol. 5, no. 3, pp. 336–340, 2008.

[178] F. Noorbehbahani, A. Fanian, R. Mousavi, and H. Hasannejad, “An incremental intrusion detection system using a new semi-supervised stream classification method,” *Int. J. Commun. Syst.*, vol. 30, no. 4, pp. 1–26, 2017.

[179] W. Liu, J. He, and S.-F. Chang, “Large graph construction for scalable semi-supervised learning,” in *International Conference on Machine Learning*, 2010, pp. 679–689.

[180] T. Isobe, E. D. Feigelson, M. G. Akritas, and G. J. Babu, “Linear regression in astronomy,” *Astrophys. J.*, vol. 364, pp. 104–113, 1990.

[181] R. E. Precup, H. I. Filip, M. B. Rədac, E. M. Petriu, S. Preitl, and C. A. Dragoş, “Online identification of evolving Takagi-Sugeno-Kang fuzzy models for crane systems,” *Appl. Soft Comput. J.*, vol. 24, pp. 1155–1163, 2014.

[182] S. Heddam and N. Dechemi, “A new approach based on the dynamic evolving neural-fuzzy inference system (DENFIS) for modelling coagulant dosage (Dos): case study of water treatment plant of Algeria,” *Desalin. water Treat.*, vol. 53, no. 4, pp. 1045–1053, 2015.

[183] V. Bianco, O. Manca, and S. Nardini, “Electricity consumption forecasting in Italy using linear

regression models," *Energy*, vol. 34, no. 9, pp. 1413–1421, 2009.

[184] X. Gu, P. P. Angelov, A. M. Ali, W. A. Gruver, and G. Gaydadjiev, "Online evolving fuzzy rule-based prediction model for high frequency trading financial data stream," in *IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 2016, pp. 169–175.

[185] X. Yan and X. Su, *Linear regression analysis: theory and computing*. World Scientific, 2009.

[186] J. S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man Cybern.*, vol. 23, no. 3, pp. 665–685, 1993.

[187] P. P. Angelov, D. P. Filev, and N. K. Kasabov, *Evolving intelligent systems: methodology and applications*. 2010.

[188] N. K. Kasabov and Q. Song, "DENFIS : Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-Series Prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, 2002.

[189] P. Angelov and R. Buswell, "Identification of evolving fuzzy rule-based models," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 5, pp. 667–677, 2002.

[190] W. Leigh, R. Hightower, and N. Modani, "Forecasting the New York stock exchange composite index with past price and interest rate on condition of volume spike," *Expert Syst. Appl.*, vol. 28, no. 1, pp. 1–8, 2005.

[191] A. Al-Hmouz, Jun Shen, R. Al-Hmouz, and Jun Yan, "Modeling and simulation of an adaptive neuro-fuzzy inference system (ANFIS) for mobile learning," *IEEE Trans. Learn. Technol.*, vol. 5, no. 3, pp. 226–237, 2012.

[192] P. Angelov and R. Buswell, "Evolving rule-based models: a tool for intelligent adaption," in *IFSA world congress and 20th NAFIPS international conference*, 2001, pp. 1062–1067.

[193] P. P. Angelov, *Evolving rule-based models: a tool for design of flexible adaptive systems*. Springer Berlin Heidelberg, 2002.

[194] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," *ACM Comput. Surv.*, vol. 41, no. 3, p. Article 15, 2009.

[195] A. Bernieri, G. Betta, and C. Liguori, "On-line fault detection and diagnosis obtained by implementing neural algorithms on a digital signal processor," *IEEE Trans. Instrum. Meas.*, vol. 45, no. 5, pp. 894–899, 1996.

[196] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *Proceedings of the 2000 Acm Sigmod International Conference on Management of Data*, 2000, pp. 1–12.

[197] N. Abe, B. Zadrozny, and J. Langford, "Outlier detection by active learning," in *ACM International Conference on Knowledge Discovery and Data Mining*, 2006, pp. 504–509.

[198] V. Hautam and K. Ismo, "Outlier Detection Using k-Nearest Neighbour Graph," in *International Conference on Pattern Recognition*, 2004, pp. 430–433.

[199] H. Moonesinghe and P. Tan, "Outlier detection using random walks," in *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, 2006, pp. 532–539.

[200] M. Salehi, C. Leckie, J. C. Bezdek, T. Vaithianathan, and X. Zhang, "Fast Memory Efficient Local Outlier Detection in Data Streams," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp.

3246–3260, 2016.

[201] L. Kuncheva, *Combining pattern classifiers: methods and algorithms*. Hoboken, New Jersey: John Wiley & Sons, 2004.

[202] R. R. Yager and D. P. Filev, “Approximate clustering Via the mountain method,” *IEEE Trans. Syst. Man. Cybern.*, vol. 24, no. 8, pp. 1279–1284, 1994.

[203] D. E. Denning, “An Intrusion-Detection Model,” *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 2, pp. 222–232, 1987.

[204] C. Thomas and N. Balakrishnan, “Improvement in intrusion detection with advances in sensor fusion,” *IEEE Trans. Inf. Forensics Secur.*, vol. 4, no. 3, pp. 542–551, 2009.

[205] F. A. Allah, W. I. Grosky, and D. Aboutajdine, “Document clustering based on diffusion maps and a comparison of the k-means performances in various spaces,” in *IEEE Symposium on Computers and Communications*, 2008, pp. 579–584.

[206] N. Dehak, R. Dehak, J. Glass, D. Reynolds, and P. Kenny, “Cosine Similarity Scoring without Score Normalization Techniques,” in *Proceedings of Odyssey 2010 - The Speaker and Language Recognition Workshop (Odyssey 2010)*, 2010, pp. 71–75.

[207] R. G. Casey, “Moment Normalization of Handprinted Characters,” *IBM J. Res. Dev.*, vol. 14, no. 5, pp. 548–557, 1970.

[208] T. M. Lehmann, C. Gönner, and K. Spitzer, “Survey: interpolation methods in medical image processing.,” *IEEE Trans. Med. Imaging*, vol. 18, no. 11, pp. 1049–1075, 1999.

[209] P. Thevenaz, T. Blu, and M. Unser, “Interpolation revisited,” *IEEE Trans. Med. Imaging*, vol. 19, no. 7, pp. 739–758, 2000.

[210] G.-S. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, and L. Zhang, “AID: a benchmark dataset for performance evaluation of aerial scene classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3965–3981, 2017.

[211] Y. Yang and S. Newsam, “Bag-of-visual-words and spatial extensions for land-use classification,” in *International Conference on Advances in Geographic Information Systems*, 2010, pp. 270–279.

[212] L. Fei-Fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, 2006.

[213] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[214] R. Keys, “Cubic convolution interpolation for digital image processing,” *IEEE Trans. Acoust.*, vol. 29, no. 6, pp. 1153–1160, 1981.

[215] J. W. Hwang and H. S. Lee, “Adaptive image interpolation based on local gradient features,” *IEEE Signal Process. Lett.*, vol. 11, no. 3, pp. 359–362, 2004.

[216] S. B. Park, J. W. Lee, and S. K. Kim, “Content-based image classification using a neural network,” *Pattern Recognit. Lett.*, vol. 25, no. 3, pp. 287–300, 2004.

[217] A. Oliva and A. Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope,” *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[218] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pp. 886–893.

[219] R. Jensen and Q. Shen, "Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 12, pp. 1457–1471, 2004.

[220] R. Jensen and Q. Shen, "Fuzzy-rough sets assisted attribute selection," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 1, pp. 73–89, 2007.

[221] R. Diao and Q. Shen, "Feature selection with harmony search," *IEEE Trans. Syst. Man. Cybern. B. Cybern.*, vol. 42, no. 6, pp. 1509–23, 2012.

[222] R. Diao and Q. Shen, "Nature inspired feature selection meta-heuristics," *Artif. Intell. Rev.*, vol. 44, no. 3, pp. 311–340, 2015.

[223] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang, "Large-scale image classification: Fast feature extraction and SVM training," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1689–1696.

[224] V. N. Vapnik., *The Nature of Statistical Learning Theory*. New York: Springer, 1995.

[225] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science (80-.).*, vol. 344, no. 6191, pp. 1493–1496, 2014.

[226] J. Li, S. Ray, and B. G. Lindsay, "A nonparametric statistical approach to clustering via mode identification," *J. Mach. Learn. Res.*, vol. 8, no. 8, pp. 1687–1723, 2007.

[227] W. Barbakh and C. Fyfe, "Online clustering algorithms.," *Int. J. Neural Syst.*, vol. 18, no. 3, pp. 185–194, 2008.

[228] P. Franti, O. Virmajoki, and V. Hautamaki, "Probabilistic clustering by random swap algorithm," in *IEEE International Conference on Pattern Recognition*, 2008, pp. 1–4.

[229] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen.*, vol. 7, no. 2, pp. 179–188, 1936.

[230] I. Kärkkäinen and P. Fränti, "Dynamic local search algorithm for the clustering problem," 2002.

[231] P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognit.*, vol. 39, no. 5, pp. 761–775, 2006.

[232] A. K. Jain and M. H. C. Law, "Data clustering : a user ' s dilemma," *Lect. Notes Comput. Sci.*, vol. 3776, pp. 1–10, 2005.

[233] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, pp. 1–30, 2007.

[234] H. Chang and D. Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognit.*, vol. 41, no. 1, pp. 191–203, 2008.

[235] V. Lohweg, J. L. Hoffmann, H. Dörksen, R. Hildebrand, E. Gillich, J. Hofmann, and J. Schaede, "Banknote authentication with mobile devices," in *Media Watermarking, Security, and Forensics*, 2013, p. 866507.

[236] D. Ayres-de-Campos, J. Bernardes, A. Garrido, J. Marques-de-Sa, and L. Pereira-Leite, "SisPorto 2.0: A program for automated analysis of cardiotocograms," *J. Matern. Fetal. Med.*,

vol. 9, no. 5, pp. 311–318, 2000.

[237] F. Alimoglu and E. Alpaydin, “Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition,” in *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium*, 1996, pp. 1–8.

[238] L. M. Candanedo and V. Feldheim, “Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models,” *Energy Build.*, vol. 112, pp. 28–39, 2016.

[239] R. K. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jiřina, J. Klaschka, E. Kotrč, P. Savický, S. Towers, A. Vaiciulis, and W. Wittek, “Methods for multidimensional event classification: A case study using images from a Cherenkov gamma-ray telescope,” *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 516, no. 2–3, pp. 511–528, 2004.

[240] P. W. Frey and D. J. Slate, “Letter recognition using Holland-style adaptive classifiers,” *Mach. Learn.*, vol. 6, no. 2, pp. 161–182, 1991.

[241] S. Aeberhard, D. Coomans, and O. de Vel, “Comparison of classifiers in high dimensional settings,” 1992.

[242] M. Buscema, “Metanet*: The theory of independent judges.,” *Subst. Use Misuse*, vol. 33, no. 2, pp. 439–461, 1998.

[243] P. Fränti, O. Virmajoki, and V. Hautamäki, “Fast agglomerative clustering using a k-nearest neighbor graph,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1875–1881, 2006.

[244] A. K. Jain, R. P. W. Duin, and J. Mao, “Statistical pattern recognition: a review,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, 2000.

[245] QuantQuote Second Resolution Market Database, *https://quantquote.com/historical-stock-data*.

[246] Standard and poor (S&P) index data, *https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC*.

[247] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, “Using the ADAP learning algorithm to forecast the onset of diabetes mellitus,” in *Annual Symposium on Computer Application in Medical Care*, 1988, pp. 261–265.

[248] P. Angelov, “Fuzzily connected multimodel systems evolving autonomously from data streams,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 41, no. 4, pp. 898–910, 2011.

[249] C. Nadungodage, Y. Xia, F. Li, J. Lee, and J. Ge, “StreamFitter: A real time linear regression analysis system for continuous data streams,” in *International Conference on Database Systems for Advanced Applications*, 2011, vol. 6588 LNCS, no. PART 2, pp. 458–461.

[250] K. Tschumitschew and F. Klawonn, “Effects of drift and noise on the optimal sliding window size for data stream regression models,” *Commun. Stat. - Theory Methods*, p. 10.1080/03610926.2015.1096388, 2016.

[251] H. J. Rong, N. Sundararajan, G. Bin Huang, and P. Saratchandran, “Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction,” *Fuzzy Sets Syst.*, vol. 157, no. 9, pp. 1260–1275, 2006.

[252] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, “PANFIS : a novel incremental

learning machine," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 1, pp. 55–68, 2014.

[253] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," *IEEE Trans. Syst. Man, Cybern. Part B*, vol. 31, no. 6, pp. 902–918, 2001.

[254] J. Tan and C. Quek, "A BCM theory of meta-plasticity for online self-reorganizing fuzzy-associative learning," *IEEE Trans. Neural Networks*, vol. 21, no. 6, pp. 985–1003, 2010.

[255] P. Angelov and D. Filev, "Simpl _ eTS : A simplified method for learning evolving Takagi-Sugeno fuzzy models," in *IEEE International Conference on Fuzzy Systems*, 2005, pp. 1068–1073.

[256] E. Lughofer, P. Angelov, and X. Zhou, "Evolving single- and multi-model fuzzy classifiers with FLEXEIS-Class," in *IEEE International Conference on Fuzzy Systems*, 2007, pp. 1–6.

[257] A. Graves, *Supervised sequence labelling with recurrent neural networks*. Heidelberg: Springer, 2012.

[258] J. Wnek and R. S. Michalski, "Hypothesis-driven constructive induction in AQ17-HCI: a method and experiments," *Mach. Learn.*, vol. 14, no. 2, pp. 139–168, 1994.

[259] C. J. Matheus and L. a Rendell, "Constructive induction on decision trees," in *International Joint Conference on Arti cial Intelligence*, 1989, pp. 645–650.

[260] P. M. Ciarelli and E. Oliveira, "Agglomeration and elimination of terms for dimensionality reduction," in *International Conference on Intelligent Systems Design and Applications*, 2009, pp. 547–552.

[261] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *ACM SIGMOD Rec.*, pp. 427–438, 2000.

[262] S. R. Safavian and D. Landgrebe, "A survey of decsion tree clasifier methodology," *IEEE Trans. Syst. Man. Cybern.*, vol. 21, no. 3, pp. 660–674, 1990.

[263] E. Alpaydin and C. Kaynak, "Cascading classifiers," *Kybernetika*, vol. 34, no. 4, pp. 369–374, 1998.

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

[265] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is 'nearest neighbors' meaningful?," in *International Conference on Database Theoryheory*, 1999, pp. 217–235.

[266] H. T. Kahraman, S. Sagiroglu, and I. Colak, "The development of intuitive knowledge classifier and the modeling of domain dependent data," *Knowledge-Based Syst.*, vol. 37, pp. 283–295, 2013.

[267] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decis. Support Syst.*, vol. 47, pp. 547–553, 2009.

[268] B. A. Johnson, R. Tateishi, and N. T. Hoan, "A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees," *Int. J. Remote Sens.*, vol. 34, no. 20, pp. 6969–6982, 2013.

[269] C. Schuldt, L. Barbara, and S.- Stockholm, "Recognizing human actions : a local SVM approach," in *IEEE International Conference on Pattern Recognition*, 2004, pp. 32–36.

[270] J. Li and J. Z. Wang, "Automatic linguistic indexing of pictures by a statistical modeling approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 9, pp. 1075–1088, 2003.

[271] K. Fukushima, "Neocognitron for handwritten digit recognition," *Neurocomputing*, vol. 51, pp. 161–180, 2003.

[272] D. Kangin and P. Angelov, "Evolving clustering, classification and regression with TEDA," in *Proceedings of the International Joint Conference on Neural Networks*, 2015, pp. 1–8.

[273] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *IEEE Workshop on Applications of Computer Vision*, 1994, pp. 138–142.

[274] J. Gan, Q. Li, Z. Zhang, and J. Wang, "Two-level feature representation for aerial scene classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 11, pp. 1626–1630, 2016.

[275] T. Larrain, J. S. J. Bernhard, D. Mery, and K. W. Bowyer, "Face recognition using sparse fingerprint classification algorithm," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 7, pp. 1646–1657, 2017.

[276] M. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[277] M. A. Borgi, D. Labate, M. El Arbi, and C. Ben Amar, "Regularized shearlet network for face recognition using single sample per person," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 514–518.

[278] J. Wang, C. Lu, M. Wang, P. Li, S. Yan, and X. Hu, "Robust face recognition via adaptive sparse representation," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2368–2378, 2014.

[279] X. Shi, Y. Yang, Z. Guo, and Z. Lai, "Face recognition by sparse discriminant analysis via joint L2,1-norm minimization," *Pattern Recognit.*, vol. 47, no. 7, pp. 2447–2453, 2014.

[280] A. B. Sargano, X. Wang, P. Angelov, and Z. Habib, "Human Action Recognition using Transfer Learning with Deep Representations," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 463–469.

[281] A. M. Cheriyadat, "Unsupervised feature learning for aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 1, pp. 439–451, 2014.

[282] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features : spatial pyramid matching for recognizing natural scene categories," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2169–2178.

[283] Y. Yang and S. Newsam, "Spatial pyramid co-occurrence for image classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 1465–1472.

[284] J. Fan, T. Chen, and S. Lu, "Unsupervised feature learning for land-use scene recognition," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 4, pp. 2250–2261, 2017.

[285] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories," *Comput. Vis. Image Underst.*, vol. 106, no. 1, pp. 59–70, 2007.

[286] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Annual International Conference on Machine Learning*, 2009, pp. 1–8.

[287] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun, "Learning convolutional feature hierarchies for visual recognition," in *Advances in neural information processing systems*, 2010, pp. 1090–1098.

[288] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus, "Deconvolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2528–2535.

[289] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1794–1801.

[290] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3360–3367.

[291] W. Luo, J. Li, J. Yang, W. Xu, and J. Zhang, "Convolutional sparse autoencoders for image classification," *IEEE Trans. Neural Networks Learn. Syst.*, pp. 1–6, 2017.