Unsupervised tracking of time-evolving data streams and an application to short-term urban traffic flow forecasting

by

Amr Abdullatif
Unsupervised tracking of time-evolving data streams and an application to short-term urban traffic flow forecasting

by

Amr Abdullatif

May, 2018
Title: Unsupervised tracking of time-evolving data streams and an application to short-term urban traffic flow forecasting

Advisors: Francesco Masulli
Stefano Rovetta

Ext. Reviewers:
Acknowledgement

I am indebted to many people for their help and support I receive during my Ph.D. study and research at DIBRIS-University of Genoa.
First and foremost, I would like to express my sincere thanks to my supervisors Prof. Dr. Masulli, and Prof. Dr. Rovetta for the invaluable guidance, frequent meetings, and discussions, and the encouragement and support on my way of research.
I thanks all the members of the DIBRIS for their support and kindness during my 4 years Ph.D.
I would like also to acknowledge the contribution of the projects Piattaforma per la mobilità Urbana con Gestione delle INformazioni da sorgenti eterogenee (PLUG-IN) and COST Action IC1406 High-Performance Modelling and Simulation for Big Data Applications (cHiPSet).
Last and most importantly, I wish to thanks my family: my wife Shaimaa who stays with me through the joys and pains; my daughter and son whom gives me happiness every-day; and my parents for their constant love and encouragement.
Abstract

Data streams are a major paradigm in data science. They are always related to time, although to different degrees. They may represent actual time series or quasi-stationary phenomena that feature longer-term variability, e.g., changes in statistical distribution or a cyclical behavior. In these non-stationary conditions, a given model is expected to be appropriate only in a temporal neighborhood of where it has been validated/learned. Its validity may decrease smoothly with time (concept drift), or there may be sudden changes, for instance when switching from one operating condition to a new one (concept shift). The proposed approach, named Graded Possibilistic c-Means stream clustering (GPCM stream) method, consists in a clustering process able to adapt to streaming data, by implementing a continuous learning exploiting the input patterns as they arrive. Based on this idea we specifically exploit the ability of possibilistic clustering to cluster iteratively using both batch (sliding-window) and online (by-pattern) strategies that track and adapt to the concept drift and shift in a natural way. Measures of fuzzy “outlierness” and fuzzy outlier density are obtained as intrinsic by-products of the possibilistic clustering technique adopted. These measures are used to modulate the amount of incremental learning according to the different regimes required by non-stationary data stream clustering. The proposed method is used as a generative model to assess and improve the accuracy of forecasting models. The Robust Layered Ensemble Model (RLEM) is also proposed for short-term traffic flow forecasting. The RLEM combines the GPCM clustering method and ensembles of Time Delayed Neural Networks. The experimental validation shows that the RLEM gives an accurate forecasting of the traffic flow rates with outlier detection and shows a good adaptation to non-stationary traffic regimes. Given its characteristics of outlier detection, accuracy, and robustness, RLEM can be fruitfully integrated into real-time traffic flow management systems.
Table of Contents

List of Figures 6

List of Tables 8

Chapter 1  Introduction 9
  1.1 List of Papers and Software ........................................ 11
  1.2 Thesis outline .................................................. 12

Chapter 2  Clustering of non-stationary data streams: A survey of fuzzy partitional methods 13
  2.1 Introduction ....................................................... 13
  2.2 Data streams and the problem of clustering ..................... 14
  2.3 An overview of crisp clustering of data streams ............... 17
  2.4 Fuzzy clustering of data streams ................................ 19
    2.4.1 Fuzzy clustering based on centroids ........................ 20
      2.4.1.1 SFCM .................................................. 20
      2.4.1.2 SPFCM, sWFCM and OFCM ............................. 21
      2.4.1.3 wFCM-AC ............................................. 23
    2.4.2 Fuzzy clustering based on medoids ............................ 23
      2.4.2.1 OFCMD, HOFCMD .................................... 24
      2.4.2.2 IMMFC ............................................... 24
2.4.3 Possibilistic clustering based on centroids
  2.4.3.1 wPCM stream
  2.4.3.2 eGKPCM
  2.4.3.3 TRAC-STREAMS and GPCM stream

2.5 Comparative evaluation
  2.5.1 Quality indices
  2.5.2 Datasets
  2.5.3 Results

2.6 Summary and discussion
  2.6.1 Summary of methods properties
  2.6.2 Other approaches

Chapter 3 Graded Possibilistic c Means stream clustering method
  3.1 Clustering Non-Stationary Data Streams
  3.2 The Graded Possibilistic c-Means Model
    3.2.1 Central fuzzy clustering
    3.2.2 Maximum-entropy clustering
    3.2.3 Possibilistic clustering
    3.2.4 Graded possibilistic models
  3.3 Outlierness measurement through graded possibilistic memberships
  3.4 Learning regimes
  3.5 Graded Possibilistic c Means stream clustering method
  3.6 Experimental results
    3.6.1 Synthetic dataset
    3.6.2 KDD99

Chapter 4 Tracking time evolving data streams for short-term traffic forecasting
  4.1 Introduction
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Partitioning methods</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>Fuzzy clustering methods of data streams</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>The effect of $\eta$ on the speed of adaptation.</td>
<td>28</td>
</tr>
<tr>
<td>2.4</td>
<td>Quality indices of the fuzzy clustering methods on the Gaussian dataset: (a) PC, (b) PE, (c) XB.</td>
<td>32</td>
</tr>
<tr>
<td>2.5</td>
<td>Quality indices of the fuzzy clustering methods on the Smtp (KDDCUP99) dataset: (a) PC, (b) PE, (c) XB.</td>
<td>33</td>
</tr>
<tr>
<td>2.6</td>
<td>Quality indices of the fuzzy clustering methods on the Soil moisture and temperature dataset: (a) PC, (b) PE, (c) XB.</td>
<td>34</td>
</tr>
<tr>
<td>3.1</td>
<td>Comparison between membership profiles: (a) possibilistic model, $\alpha = 0$; (b) graded model, $\alpha = 0.8$; (c) probabilistic model, $\alpha = 1$.</td>
<td>44</td>
</tr>
<tr>
<td>3.2</td>
<td>Plot of isopleths of the outlierness value $\Omega$ for the clusters of a dataset.</td>
<td>46</td>
</tr>
<tr>
<td>3.3</td>
<td>The learning parameter $\theta$ as a function of $\rho$ for various values of its parameters.</td>
<td>48</td>
</tr>
<tr>
<td>3.4</td>
<td>Batch GPCM stream clustering</td>
<td>50</td>
</tr>
<tr>
<td>3.5</td>
<td>Online GPCM stream clustering</td>
<td>51</td>
</tr>
<tr>
<td>3.6</td>
<td>Outlier density $\rho$ with concept drift and shift. The true model is continuously evolving. The vertical lines mark points where the stream has been cut to create a discontinuity (concept shift). Samples of the training are shown below the graph.</td>
<td>54</td>
</tr>
<tr>
<td>3.7</td>
<td>Tracking error. Absolute difference between distortion w.r.t. true centroids and distortion w.r.t. learned centroids. Top left: batch. Top right: online. Bottom left: TRAC-STREAMS. Bottom right: $k$ means, statically trained.</td>
<td>55</td>
</tr>
</tbody>
</table>
3.8 Precision-recall and ROC curve. ................................................. 59

4.1 Diagram of the training stage in the RLEM. See text for details on the quantities and on the operational blocks mentioned in the diagram. ......................... 63

4.2 Diagram of the forecasting stage in the RLEM. See text for details on the quantities and on the operational blocks mentioned in the diagram. ......................... 65

4.3 The five snapshots taken during the clustering process of the Gaussian dataset (see figure 4.4). In each snapshot red stars are centroids. Dots are the 100 previous data points, with the 30 most recent in darker colour. .............................. 72

4.4 Degree of outlierness during the tracking of the Gaussian dataset. The numbers under the curve correspond to the five snapshots in figure 4.3. ......................... 73

4.5 The road network for the short-term traffic forecasting study and the corresponding graph. ................................................................. 74

4.6 PeMS dataset: Forecasting results of RLEM (measured values are in blue; forecasted values are in red). ................................................................. 78

4.7 RLEM model accuracy w.r.t $\alpha$ in UK for 3 months. ......................... 80

4.8 Results of the two forecasting problems. (a) Scatter plot between the target and the output (UK dataset). (b) Forecast output and the target (Genoa dataset). The regression curves are in blue. .................................................. 81

4.9 Forecast output and the target on the UK dataset with 0 drop rate. ................ 82

4.10 Binarized sum of membership of each chunk to all clusters during a run on the UK dataset. ................................................................. 82

A.1 QR-code linking the web site location of the online GPCM-stream implementation .................................................. 85

B.1 SUMO Traffic Simulator ................................................................. 94

B.2 Network area from Google map and node architecture .......................... 95

B.3 Network area SUMO GUI .................................................................. 95

B.4 SUMO generated data from detectors ................................................. 96

B.5 Close a particular link for 1 hour .................................................. 97
List of Tables

2.1 Comparative analysis of data stream clustering algorithms. ...................... 37

3.1 Parameters used in the experiments ...................................................... 53
3.2 KDDCup99 dataset ............................................................................. 56
3.3 KDDCup99 selected attributes. ............................................................ 57
3.4 KDDCup99 attributes description. ......................................................... 58
3.5 KDDCup99 dataset. ............................................................................ 58

4.1 RLEM model parameters used for the short-term traffic forecasting for the three datasets. ................................................................. 75
4.2 Performance comparison on PeMS dataset ............................................. 79
Chapter 1

Introduction

Data streams have arisen as a relevant topic during the past decade [5]. Today, however, they are quickly taking the stage as a major paradigm in data science. The current speed by which data are generated is growing more quickly than the memory available to practically usable computers, reversing a trend that lasted for decades, until not many years ago. According to IBM, $2.5 \times 10^{18}$ bytes of data are generated every day, and 90% of the existing data was generated during the last two years [124]. Most of these data, however, are not of archival importance. They are only interesting when they are referred to the present, and it is not useful to store all of them forever. In addition to this, pervasive computing, the Internet of things, and ubiquitous ambient and wearable sensors provide uninterrupted flows of data that need to be taken care of in real time.

In other words, many, if not most, interesting data come in the form of streams. They arise naturally from continuously observed phenomena in an ever-increasing range of fields, from the web, to wearable sensors, to intelligent transportation systems, to smart homes and cities. But, in addition to this, the size of any collection of “big data” makes single-pass methods a necessity, turning these data effectively into a special case of streaming data.

A common requirement in the task of mining data streams is the ability to tell the interesting information from the useless one. This may be for limiting the required band or memory, for summarisation purposes, or even for cognitive reasons. For instance, a sensor network may only transmit those patterns that require the attention of a human supervisor, who should be subject to as few false alarms as possible. This task goes by the name of anomaly detection [23]. One common approach to anomaly detection is unsupervised: you learn a baseline model of the phenomenon of interest, and then measure the discrepancy of subsequent data from the baseline. An anomalous observation is that which is not well explained by the model.

The concept of data stream is related to an evolution in time, with varying degrees of time cor-
relation between observations. In time series, generated at a more or less constant rate, each observation is strongly dependent on the previous ones (e.g., sensor data). “Episodic” phenomena [47] feature inter-observation dependency only within a bounded time horizon (e.g., web sessions). Observations may also be independent of each other and related only by a common underlying model (e.g., stock markets). These categories are blurred, and often in practice, this characterization is not available beforehand.

In addition, when the observation is extended in time, the source of the stream is expected to change. We distinguish between two types of change. For evolutionary, smooth changes we use the term concept drift, while a revolutionary, sudden change is labeled concept shift.

Clustering is often one of the earliest and most important steps in the streaming data analysis workflow. A wide literature is available about stream data clustering [103]; however, less attention is devoted to the fuzzy clustering approach, even though the non-stationary nature of many data streams makes it especially appealing.

In fuzzy clustering, clusters are fuzzy sets and membership to clusters is evaluated according to membership functions, usually obtained by minimizing an objective function. Most fuzzy clustering methods are of the central type, where the model is explicitly represented by means of centroids or prototypes. The seminal Fuzzy c-Means (FCM) [15] and its many derivatives are based on this representation, as are the Maximum Entropy clustering method [98, 99] or the Possibilistic Clustering method in its two variants [71, 72]. This work focuses on possibilistic clustering as a means to perform clustering of non-stationary streaming data. We show that, as opposed to other approaches, a possibilistic clustering procedure based on the Graded Possibilistic c-Means [84] is capable of both learning the model and analyzing its evolution to be able to track it. An index to measure model change is proposed, based solely on the inferred model, not on additional measures.

We study two approaches, batch (sliding-window) and online (by-pattern), to track and adapt to concept drift and shift. For both, an incremental re-training strategy is adopted, where the amount of re-training, and therefore the required computational effort, is modulated by the proposed measure of model change.

The proposed methods are based on the Graded Possibilistic c-Means [84]. This clustering method was designed to provide the robustness allowed by the classic possibilistic approach [71, 72] in a user-controllable fashion and with easier convergence. The suitability of possibilistic clustering for building models of the data has been verified in a kernel-based approach to the “one-cluster clustering” problem [33].

Our proposed methods have the ability to prevent outliers in the data stream from having a strong effect on the forecasting accuracy, and, are capable of both learning the data stream and analysing its evolution for the purpose of tracking it. To this end, an index to measure data stream change is proposed, based solely on the memberships to clusters, and not on additional measures.
The proposed methods are evaluated on a synthetic dataset, and real datasets. Moreover, a real-time short-term urban traffic flow forecasting application is proposed, taking into consideration both spatial (road links) and temporal (lag or past traffic flow values) information. The model called RLEM which combines Artificial Neural Networks and Graded Possibilistic Clustering models obtaining an accurate forecast of the traffic flow rates with outlier detection and adaptation to the changes in the traffic flow rates. The RLEM model provides promising results and given its characteristics of outlier detection, accuracy, and robustness, it can be fruitfully integrated into traffic flow management systems.

1.1 List of Papers and Software

- Papers published during the PhD:

  
  
  
  
  
Software published during PhD:

- Online Graded Possibilistic Clustering

This thesis interpolates the material from the papers [1–4] by the author.

1.2 Thesis outline

The rest of the thesis is organized as follows:

Chapter 2 provides a review of the main state of art data stream clustering algorithms focusing mainly on fuzzy methods, including their treatment of outliers and concept drift and shift. The clustering methods are compared based on their characteristics of change, types of granulation, and clustering processes. Some validity indices have been applied to compare the performances of considered methods on several datasets.

Chapter 3 provides a detailed description of relevant issues of the clustering methods for data stream. Then describes the Graded Possibilistic c Means (GPCM) clustering method [84] and the followed approach for continuous learning needed for tracking evolving data streams. Two versions of the proposed Graded Possibilistic c Means stream clustering (GPCM stream) method, batch and online, are presented together with their experimental validation.

Chapter 4 summarizes the state of the art in streaming data clustering and urban traffic modelling and then proposes the Robust Layered Ensemble Model (RLEM) for short-term traffic flow forecasting. The RLEM combines the GPCM clustering method and ensembles of Time Delayed Neural Networks. The experimental validation shows that the RLEM gives an accurate forecasting of the traffic flow rates with outlier detection and shows a good adaptation to non-stationary traffic regimes. Given its characteristics of outlier detection, accuracy, and robustness, RLEM can be fruitfully integrated into real-time traffic flow management systems.

Chapter 5. conclusions and directions for some future research are presented in this chapter.

Appendix A. contains the clustering model code that is used in the online experiments.

Appendix B. contains the implementation of the Genoa traffic simulator.
Chapter 2

Clustering of non-stationary data streams: A survey of fuzzy partitional methods

This chapter is based on the paper [2]:


It reviews the main state of art data stream clustering algorithms focusing mainly on fuzzy methods, including their treatment of outliers and concept drift and shift. The clustering methods are compared based on their characteristics of change, types of granulation, and clustering processes. Some validity indices have been applied to compare the performances of considered methods on several datasets.

2.1 Introduction

Our society is investing massively in the collection and processing of data of all kinds, on scales unimaginable until recently [27]. The speed by which data are currently generated is growing more quickly than the memory available to practically usable computers (information overload challenge), reversing a trend that lasted for decades, until not many years ago. Most of these data, however, are not of archival importance. They are only interesting when they are referred to the present, and it is not useful to store all of them forever. In addition to this, pervasive computing, the Internet of things, and ubiquitous ambient and wearable sensors provide uninterrupted flows of data that need to be taken care of in real time. In other words, many, if not most, interesting data come in the form of streams.

Mining data streams is a big challenge as they always depend on time, although to different
degrees. They may represent actual time series, with a strong dependency on time, or quasi-stationary phenomena whose variability can be appreciated only in the long term, e.g., as evolutions in statistical distribution or a cyclical behavior. In these non-stationary conditions, any model is expected to be appropriate only in a neighborhood of the point in time where it has been learned. Over time, its validity may decrease (concept change). Change may be gradual (concept drift), or occur suddenly, for instance when switching from one operating condition to a new one (concept shift).

The unsupervised analysis in the form of data clustering provides a useful tool to mine data streams. Clustering methods [32, 58, 116] have been successfully applied in many fields such as data mining, image segmentation, fraud detection, and bioinformatics.

The aim of data clustering is to group objects on the basis of a dissimilarity (or similarity) criterion obtaining clusters that are sets containing similar objects. The dissimilarity between objects is usually measured by a distance function defined on pairs of objects. The most popular distance function is the Euclidean distance [28], which is often used to measure dissimilarity between two objects and is known to work well when all clusters are well separated.

Let \( X = \{x_1, \ldots, x_n\} \) be a dataset composed by \( n \) patterns for which every \( x_i \in \mathbb{R}^d \). The codebook \( V \) is defined as the set \( V = \{v_1, \ldots, v_c\} \), typically with \( c \ll n \). Each element \( v_i \in \mathbb{R}^d \) is called a centroid and usually is in a one-to-one relationship with a cluster. The inclusion of observation number \( i \) in clusters number \( j \) is measured by the membership \( \mu_{ij} \). Centroids are locations in the data space that minimize their average squared distance to points in their cluster. An alternative representation of a cluster is by means of the point in the cluster that minimizes the average absolute distance from all others. This is termed a medoid.

In the following, we will refer to the data stream as generated by a data source with given characteristics, possibly non-stationary. A clustering model will consist in the implementation of clusters as inferred from the data, along with their describing parameters. The problem of data stream clustering is then summarized as that of finding an optimal clustering model in the presence of source variations. The representation of the model, as well as the objective that measures model quality, depend on the specific clustering method and algorithm.

In the next section, we briefly introduce the problem of data streams and approaches to represent the partitional clustering models.

### 2.2 Data streams and the problem of clustering

The challenges of data streams clustering can be summarized as follows:

- **Single-pass operation**: Stream data naturally impose a single-pass constraint on the algorithms.
• **Robustness to outliers**: Outliers are isolated observations which are clearly not explained by the clustering model, while the majority of observations are well fitted by the current clustering model. The clustering model should be insensitive to these episodic anomalies.

• **Tracking model changes**: To remain relevant over time, the clustering model should be able to quickly adapt to concept drift and shift.

• **Balance between robustness and tracking ability**: Insensitivity to outliers and reactivity to changes are two competing goals. The balance between them is a problem-dependent design choice.

In this work, data streams are assumed to generate independent, random instances of objects from an underlying distribution, the data source. The distribution is non-stationary, i.e., evolving, but its rate of change is mostly slow with respect to the sampling rate (this is concept drift according to our terminology) so that clustering is feasible. In the presence of concept drift only, we assume that there exists an interval around the present instant in time where the distribution can be considered stationary. We refer to this assumption as the “local i.i.d.” hypothesis. Occasionally, we assume that this hypothesis is broken. These changes can either be pointwise in time (outliers) or stable over a period of time, so that the source cannot be described by the same model as before (concept shift).

Chapter 3 provides a detailed description of relevant issues of the clustering methods for data streams. Then describes the Graded Possibilistic \( c \) Means (GPCM) clustering method [84] and the followed approach for continuous learning needed for tracking evolving data streams. Two versions of the proposed **Graded Possibilistic \( c \) Means stream clustering** (GPCM stream) method, batch and online, are presented with togheter their experimental validation. Any applicative problems can be modeled in this way. As an example, to monitor the health and well-being of elderly people at home alone, human activity recognition may be inferred from ambient (motion, presence, location) and wearable (3D acceleration, 3D orientation, odometry, bio-signal) sensors. In this scenario, observed patterns will be clustered in groups corresponding, not necessarily one-to-one, to individual everyday activities. Outliers are activities that are occasionally performed in an unusual way with respect to the norm, due to temporary reasons (e.g., lunch time delayed because of a long phone call from a relative). Concept shift is represented by changes in ambient conditions that induce changes in behavior and routines (e.g., starting the habit of an outdoor walk in the evening when the weather becomes sufficiently mild), while drift can indicate either improvement (e.g., the acquisition of better skills in physical exercise) or decline (e.g., gradual loss of interest in cooking) in daily habits.

---

1 Of course, depending on the length of the time interval considered and on the magnitude of the model change, the definitions of concept shift and drift can blur into each other. This depends on the method, on the source, and on the user’s decision strategy. For simplicity of presentation here we assume that drift and shift are sufficiently distinguishable.
Another application can be in short-term road traffic forecasting [1]. The forecast of vehicle flow and density on a given road in the next few minutes can be based on a model of traffic conditions that represent different situations (different times of the day, different days of the week) as individual clusters. Usually, these are stationary, but the importance of some routes may decrease or increase in time as a result, for instance, of the closing of a sports center or the construction of a new shopping center (concept drift). Unusual patterns may be observed in case of exceptional events or road accidents (outliers). Sometimes, these changed conditions may persist as a new set of traffic patterns, as in the case of extended road works (concept shift).

Let \( O = \{ o_1, \ldots, o_n \} \) denote a stream of \( n \) objects, e.g., traffic flow patterns on a street network or web activity profiles on an e-commerce website. In the streaming data model, \( n \) is very large (in principle, unbounded) and objects are observed sequentially.

Individual observations in the stream are represented by a fixed set of features. Specifically, the methods considered here assume that each observed object is associated to a numeric feature vector: \( o_i \rightarrow x_i \subset \mathbb{R}^d \). Selecting the best representative features of the objects plays an important role in ensuring a good clustering performance.

In general, incremental clustering processes data in chunks [24] of size \( s \ll n \). It can be online if chunks are trivially reduced to one observation (\( s = 1 \)), batch if they consist of a window of larger size \( s > 1 \), either sliding (overlapping) or disjoint. Schemes using disjoint windows introduce a clear distinction between current observations, which are retained, and old ones, which are summarized and discarded. Temporal weights (damped models [103]) can also be used to account for gradual ageing, either over the whole history or on a fixed-size window. With fixed window size, an important parameter that controls the tracking ability is the memory length of the method, which is related to window size and to the damping factor when fading weights are used.

Clustering methods can be roughly divided into the two categories of partitioning and hierarchical methods [58]. We focus on the partitional methods summarized in figure 2.1 because in the literature this is the area where most of the research effort is spent. Partitions of a set of data (clusters) can be of two main types: crisp, or fuzzy [15, 71].

Crisp (or hard) clusters are characterized by binary, integer-valued memberships, \( \mu_{ij} \in \{0, 1\} \subset \mathbb{N} \). Also, if \( \mu_{ij} = 1 \), then \( \mu_{ik} = 0 \ \forall k = 1 \ldots c, k \neq j \). In other words, each object can be a member of only one cluster, for which it has membership degree of one. Mutual exclusion can also be expressed with the following probabilistic constraint:

\[
\sum_{j=1}^{c} \mu_{ij} = 1 \ \forall i : 1 \ldots n . \tag{2.1}
\]

In contrast to crisp partitions, fuzzy (or soft) partitions admit real-valued memberships, \( \mu_{ij} \in [0, 1] \subset \mathbb{R} \). Each object can be a member of all clusters to which it has in general different
membership degrees, each less than one. The probabilistic constraint of equation (2.1) can either be enforced or not. In the latter case, the model is termed *possibilistic*. [71, 72]

2.3 An overview of crisp clustering of data streams

Before focusing on fuzzy methods, this section concisely reviews the main approaches to crisp clustering of data streams.

The following types of algorithms are available in the literature:

- **Sample-based** clustering algorithms: One of the common strategies is using sampling on the data, then clustering is executed on the samples. Popular sample-based methods include CLARA [63], CLARANS [89], CURE [44], and the coresets algorithms [48];

- **Single-pass** algorithms: These algorithms adopt the single pass model to deal with the data stream requirement that data can be examined only once. According to this model, as data are scanned, summaries of past data are stored to leave enough memory for processing new incoming data. These algorithms are categorized into incremental approaches [17, 18, 79] and divide-and-conquer approaches [6, 43]. Incremental approaches may either deal with one object at a time, or collect data into chunks. The goal of incremental clustering is
to find $K$ representative (clusters) to represent the whole dataset and determine the final clustering results;

- **Data transformation** algorithms: These algorithms alter the structure of the data themselves so that they can be more effectively accessed. Popular data transformation algorithms include BIRCH, [122] CLUTO, [91] and GARDEN [62];

- **Density-based** clustering algorithms: These approaches aim to find clusters of arbitrary shape by modelling clusters as dense regions separated by sparse regions [8]. Popular density-based algorithms include DBSCAN [30], OPTICS [10], DENCLUE [51], DenStream [19], rDen-Stream [78], and DD-STREAM [60]. Their common strategy revolves around finding core samples in high-density areas and expanding clusters from them, which is especially effective when the data contain clusters of similar density.

Many of the methods mentioned so far only focus on the progressive nature of the task, with less attention to tracking evolutions in the data stream, i.e., they do not include mechanisms to take into account possible changes in the source, but only to build the model in an incremental fashion. The ability to track changes, however, is a basic requirement for methods to be suitable for non-stationary stream learning.
2.4 Fuzzy clustering of data streams

Most fuzzy clustering methods are modeled over Fuzzy $c$-Means [16], a probabilistic method where the goal is to minimize the following objective function:

$$J_{FCM} = \sum_{j=1}^{c} \sum_{i=1}^{n} \mu_{ij}^m \|x_i - v_j\|^2,$$

(2.2)

where $m \geq 1$ is a fuzzification parameter. This results in a “soft partitioning” method, which allows observation to belong to all clusters to varying degrees, provided that equation (2.1) is satisfied (i.e., the total membership of a point to all clusters must be 1).

As already noted there are mainly two types of fuzzy clustering methods. If equation (2.1) is enforced we term them probabilistic, otherwise they are possibilistic. The idea of possibilistic clustering is to view membership degrees as typicalities. This allows points to have low memberships to all clusters, which is an indication of a point being an outlier.

Fuzzy clustering, despite being less used in stream data analysis, offers potential advantages over crisp partitioning:

- In the case of non-stationary streams, i.e., in the presence of concept drift, the progressive loss of validity of a clustering model can be better represented by the real-valued membership which changes smoothly, without having to make abrupt changes in the model;

- With crisp membership, for which only two values are admissible, detection of a concept change requires a data sample sufficiently large to allow reliable statistical estimate, while with fuzzy models a pointwise evaluation is possible, thus allowing much faster detection;

- In the particular case of possibilistic models, the additional robustness property allows pointwise detection of outliers in addition to source change detection.

Fuzzy clustering also has some disadvantages that should be taken into account, stemming from the fact that memberships are real-valued quantities. The search space is larger, convergence is asymptotic, and the arithmetic employed in software implementations is floating-point. These aspects make fuzzy methods intrinsically slower. However, for data streams these issues are mitigated by the need to work within a limited temporal horizon to reduce the effects of non-stationarity, so this disadvantage may not be relevant in many applications.

The fuzzy stream clustering methods that are reviewed in the following are summarized in figure 2.2.
2.4.1 Fuzzy clustering based on centroids

In these methods, centroids are computed by using an existing “batch” fuzzy clustering method on each chunk of data. Changes from one chunk to the next are smoothed by including in the next chunk a suitably summarized version of the previous chunk (a “synopsis”).

A minimal synopsis is usually represented by the location of past centroids. Additional information is typically included, mostly in the form of weights associated to points. The introduction of weights into the FCM objective has been studied in the context of data reduction [29] and results in a method termed Weighted Fuzzy c Means (wFCM) [102]. Singleton points from the current chunk receive unit weights, while centroids from the previous iteration(s) are weighted taking into account their membership mass and previous weight (which at iteration $t = 0$ are initialised as all ones):

$$w_h(t) = \sum_{j=1}^{c} u_{hj}w_j(t-1), \quad 1 \leq h \leq c, t \geq 1.$$  (2.3)

The weights so obtained, or computed according to other formulas for other methods, are used to scale the contribution of the points to which they refer when computing averages. In this way, the importance of points can be differentiated when updating the model.

The methods described in this subsection use wFCM and differ mostly in the details of how weighting is computed and in the historical information that is included in each “batch” along with the current chunk of data.

As a convenient standard for describing the different methods, we will draw inspiration from the Dynamic Clustering Cube (DCC) framework [94]. DCC views clustering from the granular computing standpoint whereby clusters represent a coarser level of granulation with respect to raw data objects.

The DCC is based on three crucial dimensions to describe any clustering algorithm:

- **Characteristics of change**: The types of concept change that are taken into account;
- **Types of granulation**: The representation of clusters;
- **Clustering processes**: The details of how the method builds and maintains the clustering model.

2.4.1.1 SFCM

The Streaming Fuzzy c Means (SFCM) method [53] is a streaming variant of the Fuzzy c Means, with the goal of balancing the trade-off between tracking an evolving distribution and summarizing the data seen so far. This trade-off is tuned by varying past history usage.
The method starts by clustering $s$ data points that are present at time instant $t$ and clusters are summarized by $c$ cluster centroids. Each centroid is weighted by the sum of all the points’ memberships in that cluster. For $s$ observations at time $t$, the weight of centroid $j$ is computed as a sum of memberships:

$$w_j = \sum_{i=1}^{s} \mu_{ij}, \quad 1 \leq j \leq c.$$  

(2.4)

In the next chunk, the centroids from the last chunks will be used as initial centroids for clustering the new points. The length of the history taken into account, i.e., how many past centroids are stored, depends on the size of the available memory.

**Dimension 1: Characteristics of change.** SFCM can track any type of variation in the data, as long as they don’t imply a change in the number of clusters which is fixed in advance.

**Dimension 2: Types of granulation.** The data are assumed to arrive in chunks where each chunk is processed as a whole. Clusters are computed on each chunk. The number of clusters stored before updating depends on the size of the available memory and can be defined by the user.

The method is based on fuzzy $c$ means clustering and cluster representation is by means of centroids and a membership function.

**Dimension 3: Clustering processes.** The first chunk’s cluster centroids are initialized randomly while the other chunks’ are initialized as the last chunk’s cluster centroids. Using the past result as initialisation encourages, but does not guarantee, similar cluster positions from one chunk to the following one. Consequently, the dynamics of change, in general, is stepwise rather than continuous.

SFCM works well for large datasets. However, it uses a fixed number of clusters, which may be sub-optimal for non-stationary data streams. Additionally, the centroids are updated by doing re-clustering after a certain number of chunks defined by the user and not through an incremental learning mechanism.

### 2.4.1.2 SPFCM, sWFCM and OFCM

Both Single-Pass Fuzzy $c$ Means (SPFCM) [54] and Online Fuzzy $c$ Means (OFCM) [55] are incremental fuzzy methods \(^2\). Similar to SFCM, they process incoming data chunk by chunk. SPFCM and OFCM differ in the way they handle the centroids of each chunk.

In SPFCM the data are assumed to be loaded into memory, and chunks are sampled randomly. In the hypotheses stated in the previous section, data from the stream are equivalent to a random

---

\(^2\)The C implementations OFCM and SPFCM that were used for this study are available at: [http://www.csee.usf.edu/~lohall/scalable/](http://www.csee.usf.edu/~lohall/scalable/).
sampling, so this method can be applied to streams. The centroids obtained from the last chunks are combined into the upcoming chunk, with weights that are computed from memberships, and at the end, a final set of centroids for the entire dataset is generated. The weight of centroid $j$ in each chunk is computed as follows:

$$w_j = \frac{n+c}{\sum_{i=1}^{c} \mu_{ij}w_i}, \quad 1 \leq j \leq c.$$  \hspace{1cm} (2.5)

The weights of the $n$ singleton objects are all set to one. The *Stream Weighted Fuzzy c Means* (sWFCM) [111] method is essentially similar to SPFCM. In OFCM, the weights are computed as a combination of past weights and memberships, as follows:

$$w_j = \sum_{i=1}^{n} \mu_{ij}w_i, \quad 1 \leq j \leq c, \quad w_i = 1, \forall 1 \leq i \leq n.$$ \hspace{1cm} (2.6)

Centroids are computed independently on each chunk, and a consensus post-processing is required.

**Dimension 1: Characteristics of change.** Both methods can handle large data and are able to track drift, but again they are not suitable for detecting concept change in data streams because they lack a mechanism to distinguish between outliers and steady source changes.

**Dimension 2: Types of granulation.**

SPFCM reads data according to the partial data accesses (PDA) model. Data are randomly accessed. The PDA size is defined by the user.

sWFCM and OFCM assume data arriving as a stream of chunks where the chunk size is defined by the user.

The clustering model is centroid-based, with a membership function.

**Dimension 3: Clustering processes.** SPFCM starts by clustering the first PDA into $c$ cluster centroids using fuzzy $c$ means. Then the data in memory is condensed into $c$ weighted points. Those weighted points will be clustered with new points in the next PDA.

OFCM also clusters data in each chunk by fuzzy $c$ means. The maximum number of clusters is assumed to be known. After data in one chunk is clustered, memory is freed by condensing the clustering solution in the memory into $c$ weighted examples. After a certain amount of chunks defined by the user, the weighted clustering solutions are merged into $c$ final clusters. The merging operation is done by clustering all the weighted centroids in the ensemble using their weights using Weighted FCM (wFCM).

In principle, SPFCM requires randomly sampling the data. If the source obeys the “local i.i.d.” assumption, this is approximately true and the method is applicable.

These methods require a priori knowledge about the number of clusters.
Algorithm 1 wFCM-AC method [86]

1: Apply the wFCM on the data with current cluster number \( c \)
2: Apply the wFCM on the data with cluster number \( c - 1 \) and choose the best structure.
3: Apply the wFCM on the data with cluster number \( c + 1 \) and choose the best structure.
4: Choose the structure which improves the quality measure.

2.4.1.3 wFCM-AC

The problems of detecting outliers and tracking concept change are not considered in the previously described methods, where concept change is handled by re-clustering. To cope with evolving nature of data streams, an extension to sWFCM called \textit{WFCM with Adaptive Cluster number} (wFCM-AC) [86] was presented. The adaptive process is a local search by varying the number of clusters, as summarized in algorithm 1. The quality of the searched configurations is measured by the Xie-Beni index [93, 115]. The advantage of this algorithm is the dynamic determination of the optimal number of clusters, although at any given step the search is limited to the options \( \{c - 1, c, c + 1\} \). The methods work well in tracking concept drift in data streams, but it doesn’t have an outlier/change detection mechanism. It is also slower than the other algorithms because it looks for the optimal number of clusters in each chunk of data.

\textbf{Dimension 1: Characteristics of change.} The method adapts to concept drift/shift including changes in the number of clusters.

\textbf{Dimension 2: Types of granulation.} The data are assumed to arrive in chunks where each chunk is processed as a whole. FCM is applied to normalized chunks from the stream, where each object is standardised by subtracting the mean and dividing the result by the standard deviation. Each cluster is weighted by summing the membership values that belong to it.

\textbf{Dimension 3: Clustering processes.} The method starts by clustering the first normalized chunk using fuzzy \( c \) means. The obtained centroids are used as an initialization for clustering the next chunk. For each chunk, the algorithm tries to find the best number of clusters by increasing and decreasing the number of clusters by one. Quality of clustering structure is measured for each iteration of the algorithm and the best structure is then chosen.

wFCM-AC requires multiple evaluations for each chunk to find the best clustering structure, which makes it slow with respect to other algorithms.

2.4.2 Fuzzy clustering based on medoids

These algorithms are medoid-based. Medoids are intrinsically more robust to noise and outliers than centroids since their location is not computed but selected from the observations actually present in a cluster. For the methods considered here, a medoid is an object in a cluster that has
a minimum average dissimilarity to all the other objects in that cluster.

A drawback shared by methods based on medoids consists in the reduced reactivity to concept changes, because robust methods are inherently based on the concept of outlier rejection rather than detection.

2.4.2.1 OFCMD, HOFCMD

The Online Fuzzy c Medoids (OFCMD) [77] method is similar to OFCM in that the final set of reference points (medoids rather than centroids) is generated by the weighted fuzzy medoid (wFCMD) algorithm on medoids obtained from all the chunks.

In the History-Based Online Fuzzy c Medoids (HOFCMD) [77] model, similar to SPFCM, a subset of medoids from the previous chunk are combined with the upcoming chunk as history information. The final set of medoids is generated after processing all the chunks. These methods add robustness to the basic methods on which they are respectively modeled.

2.4.2.2 IMMFC

Different from previous two methods, the Incremental Multiple Medoids-based Fuzzy Clustering (IMMFC) [112] model provides a new mechanism by:

- Selecting multiple medoids instead of one to represent the clusters in each chunk,
- Iteratively updating both fuzzy membership and cluster weight,
- Automatically generating pairwise constraints from the medoids obtained from every chunk, which is used to help the final data partition.

The IMMFC proves to be relatively insensitive to the order of data and it was mainly used for handling large multi-dimensional data.

2.4.3 Possibilistic clustering based on centroids

In general, possibilistic partitions can be obtained by the Possibilistic c Means (PCM) [72], a possibilistic variant of FCM. Since clusters are not mutually exclusive, in this method cluster memberships do not “compete” with each other, and the extension of each cluster must be specified with an explicit, additional scale or width parameter. In general, this is a scalar, reflecting the assumption that in the data space there are no preferential directions. One method, however, uses full matrices to be able to specify arbitrary hyperellipsoidal clusters.
Possibilistic clustering was designed to be robust to outliers, so it lends itself very well to noisy stream learning. On the other hand, the number of parameters to train is higher.

It has been observed that PCM may produce overlapping clusters [12]. In the context of non-stationary streams, clusters that start collapsing on each other are used to identify a possible inadequacy in the number of clusters, providing an additional indicator of concept change. However, this often requires some additional processing. One of the methods described below [3] is based on a variant of possibilistic clustering [84] that limits this phenomenon by allowing the user to set the desired balance between competitive behavior (partitional or “probabilistic” clustering) and cooperative behavior (mode-seeking or possibilistic clustering). This is beneficial because it avoids the need of an accurate initialization or post-processing. The possibilistic stream clustering methods considered here are all based on the PCM method.

2.4.3.1 wPCM stream

The Weighted Possibilistic c Means for streams (wPCM stream) [59] method is a modified version of WFCM in a possibilistic variant (WPCM, Weighted Possibilistic c Means) to cope with changes in data streams. It assigns weights to the incoming data streams based on the existence of concept drift, which must be assessed separately by the user.

If concept change is absent, all the weights are set equal to one for all data elements:

\[ w_j = 1, \ j = 1, \ldots, n. \]  \hspace{1cm} (2.7)

In case of the existence of concept change, the weights are updated as follow:

\[ w_{j+1} = w_j 2^{\lambda}, \ w_1 = 1, \ j = 1, \ldots, n, \]  \hspace{1cm} (2.8)

where \( \lambda > 0 \) is a decay rate parameter which reflects the speed of forgetting the influence on the clustering results of the old data chunks.

**Dimension 1: Characteristics of change.** The method is suitable for non-stationary streams, but its sensitivity to change is pre-determined by the user-selected parameter \( \lambda \).

**Dimension 2: Types of granulation.** The method is based on possibilistic clusters described by a location (centroid) and a scale.

**Dimension 3: Clustering processes.** The possibilistic \( c \) means method is used on each chunk. The method is of the fixed-window-size, damped memory type.

Due to the user-selected parameter \( \lambda \) and to the fact that there is no adaptivity in weight computation, the authors report that the method performs well when it is tuned to the type of change present in the source, but is outperformed by the base PCM method when no concept change is present.
2.4.3.2 eGKPCM

The evolving Gustafson-Kessel possibilistic c Means (eGKPCM) [73] approach is a possibilistic, stream-oriented variation over the Gustafson-Kessel clustering method. In this approach, the scalar width parameters of possibilistic clustering are replaced by scale matrices, so as to allow general elliptic clusters. This is the “Gustafson-Kessel Possibilistic c-Means” (GKPCM). To obtain the evolving-stream oriented variant, data are evaluated for their typicality. Typical points are discarded after storing summary information, in the form of location and (matrix) scale of centroids. Atypical points are stored in a buffer and used to find new centroids.

The algorithm starts by defining a buffer size for storing data objects, minimum membership, fuzziness value, and a termination criteria. In the initialization also the first data objects stored in a buffer are used to calculate the initial cluster centers using GKPCM algorithm. When a new data sample is obtained, the membership of the object to the current set of cluster prototypes is calculated. If the maximal membership is bigger than the minimum initial membership then the sample belongs to the cluster with the biggest membership. If it is lower, the data object is stored in the buffer. When the data buffer is full, new clusters are identified by using GKPCM algorithm on the buffer data.

**Dimension 1: Characteristics of change.** The method is able to track several degrees of changes from concept drift to shift, including the ability to vary the number of centroids.

**Dimension 2: Types of granulation.** The cluster model, although in a possibilistic variant, is based on the Gustafson-Kessel method, so it includes both a centroid and a scale matrix.

**Dimension 3: Clustering processes.** The method is based on a fixed window and its operation can be interpreted as a non-damped, weighted model. Weights are implicitly implemented by selecting which data objects to store and which ones to discard.

The model’s ability to represent non-spherical clusters gives this method an added flexibility. On the other hand, the number of model and optimisation parameters is higher, with several pre-specified quantities, for instance on the maximum number of prototypes, the maximum aspect ratio for ellipsoids, the minimum cluster size, the typicality threshold for inclusion in the buffer, and others. The method, being specific for evolving streams, can adapt to non-stationary clusters. On the other hand, it does this by subsequent re-clusterings based on whole chunks, not by incremental updates of the model.

2.4.3.3 TRAC-STREAMS and GPCM stream

The two methods described here exploit the robustness of the possibilistic clustering model to evaluate the degree of typicality of incoming points. This allows making decisions for each observation, so they are able to update the clustering model with continuity without the need to work by chunks, i.e., in batch mode. They both belong to the family of damped models.
The **Tracking Robust Adaptive Clusters in evolving data STREAMS** (TRAC-STREAMS) [88] method builds a continuous synopsis including centroid locations and scales, and a time parameter (e.g., the number of points so far). Weights are computed from the observation-centroid distance and from the difference between their time of occurrence and the current time. The method focuses on more recent data thanks to the time-dependent weights and distinguishes between typical data and outliers by testing against a threshold obtained by the Chebychev inequality. The Chebychev bound allows creating new clusters (within a pre-set maximum number) and merging “compatible” ones, i.e., those too similar to be kept distinct.

The **Graded Possibilistic $c$ Means stream clustering** (GPCM stream) [3] method will be presented in detail in chapter 3. It uses as basic clustering model the Graded Possibilistic $c$ Means (GPCM) [84] adapted to on-line (or by-object) operation. GPCM stream learns from either individual input objects as soon as they arrive, or a sliding window of fixed size that includes the newest object while forgetting the oldest one. Measures of fuzzy outlierness and fuzzy outlier density are computed from memberships. Specifically, for each incoming observation $i$ its outlierness is its total membership to all clusters.

$$\Omega_i = \sum_{j=1}^{c} \mu_{ij},$$  

and outlier density is a time-discounted average of outlierness so far

$$\rho_i = \eta \Omega_i + (1 - \eta) \rho_{i-1},$$  

with $0 < \eta < 1$ an ageing factor depending on the desired reactivity of the method. Outlier density is then used as a feedback control signal to speed up or slow down the updates to the clustering model, modulating the amount of incremental learning. This gives the method the ability to operate in the different regimes required by non-stationary data stream clustering:

- Baseline learning (Concept drift),
- No learning (Outliers),
- Re-learning (Concept shift).

---

3A MATLAB implementation of GPCM stream is available at: https://it.mathworks.com/matlabcentral/fileexchange/64318-onlinegradedpossibilisticclustering. Appendix A of this thesis describes its implementation.
Figure 2.3: The effect of $\eta$ on the speed of adaptation.

Assume $\Omega$ is a step function, where

$$
\Omega_i = \begin{cases} 
0, & 0 \leq i \leq 10 \\
1, & 10 \leq i \leq 100 
\end{cases}
$$

(2.11)

Figure 2.3 illustrates the effect of $\eta$ on the speed of adaptation i.e., the reactivity. By modulating $\eta$, we are making the system able to respond more or less quickly. So in the case of many outliers and you don’t want learn them, you can set a low value of $\eta$.

**Dimension 1: Characteristics of change.** Both methods operate on an object-by-object basis, and include adaptive parameters to be able to track changes of different intensity from drift to shift. TRAC-STREAMS measures outlierness by means of the Chebychev inequality, while GPCM stream employs a user-selected criterion to update learning parameters on the basis of outlierness.

**Dimension 2: Types of granulation.** The model representation includes centroids and scale, as usual for possibilistic methods. TRAC-STREAMS additionally stores current and historical weights. In GPCM stream, there is a user-selectable parameter that sets the degree of possibility, making it possible to balance the trade-off between robust, mode-seeking behaviour of possibilistic models and avoidance of degenerated solutions of probabilistic models [84].
**Dimension 3: Clustering processes.** Despite being based on possibilistic clustering, neither of the two methods employs PCM directly. TRAC-STREAMS uses a density-based procedure, while GPCM stream is based on a stochastic version of the maximum entropy clustering criterion [97].

These methods have good reactivity to change, with GPCM stream giving the user more control over the reactivity-rejection trade-off.

The drawback of both algorithms is a downside of the same feature, i.e., the need for setting several initial parameters, although for both some guidance is given in the original proposal.

### 2.5 Comparative evaluation

This section deals with the comparative experimental evaluation of the fuzzy-based clustering algorithms.

For comparison the following methods were selected: OFCM, SPFCM, eGKPCM, and GPCM stream, as representative of the various approaches. The first two are representative algorithms of probabilistic clustering, and the second two are representative algorithms of possibilistic clustering.

#### 2.5.1 Quality indices

For the comparison between fuzzy and possibilistic clustering methods of data streams, we have selected three validity indices, extended to encompass possibilistic models as described by Yang and Wu [117]. They normalize the possibilistic c-memberships \{\mu_1, \ldots, \mu_c\}_p to be \{\mu'_1, \ldots, \mu'_c\}_F where for an input \(x\) the \(\mu'_j\) is defined as follows:

\[
\mu'_j(x) = \frac{\mu_j(x)}{\sum_{k=1}^{c} \mu_k(x)}, \quad j = 1, \ldots, c .
\]

The first and second validity indices involve only the membership values and are based on the assumption that the outputs are better if they are close to a crisp partition. The third validity index takes into account a measure of compactness involving data and clusters, and a separation measure between clusters.

- **PC (partition coefficient)** [15]:

\[
PC(c) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} \mu_{ij}^2,
\]
where $\frac{1}{c} \leq PC(c) \leq 1$. The higher the value of $PC$ the better the clustering quality.

- **PE (partition entropy) [13, 14]:**

  $$PE(c) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} \mu_{ij}^2 \log_2 \mu_{ij},$$  

  (2.14)

  where $0 \leq PE(c) \leq \log_2 \mu_{ij}$. The lower the value of $PE$ the better the clustering quality.

- **Xie-Beni(XB) [93, 115] validity function proposed by Xie and Beni with $m = 2$ and then generalized by Pal and Bezdek is defined by:**

  $$XB(c) = \sum_{j=1}^{c} \frac{\sum_{i=1}^{n} \mu_{ij} \|x_i - v_j\|^2}{n \min_{i,j} \|v_j - v_i\|^2} = \frac{J_{FCM}(\mu,v)}{n} / \text{Sep}(v).$$  

  (2.15)

  $J_{FCM}$ is the compactness measure, and $\text{Sep}(a)$ is the separation measure between clusters. The lower the value of $XB$ the better the clustering quality.

Data are organized in chunks of observations, where the size of each chunk is $\geq 1$. For each chunk, we compute the validity indices and summarised the index distribution in a box-and-whisker plot. Box-and-whisker plots are based on a five-number summary: $w_1, q_1, q_2, q_3$, and $w_2$, where:

- the box extends from $q_1$ to $q_3$, the first and third quartiles;
- the intermediate line is at $q_2$, the median;
- the lower whisker extends to $w_1 = q_1 - 1.5(q_3 - q_1)$;
- the upper whisker extends to $w_2 = q_3 + 1.5(q_3 - q_1)$;
- values below $w_1$ or above $w_2$ are individually marked with crosses.

This provides information about the distribution of the variations in quality of the models along the duration of each experiment, and therefore about their ability to track changes and to remain adequate even in the presence of shift, drift, and outliers.

### 2.5.2 Datasets

The datasets employed in our experimental analysis are the following:
• **Gaussian dataset** is a synthetic dataset with four evolving two-dimensional Gaussian distributions [26]. Along time, one new data point is added and one removed so that the total number stays constant. However, the underlying data source (centroid positions) is slowly changed, leading to concept drift. Concept shift is obtained by removing a whole segment of the sequence at time 4,000 where the stream changes abruptly. The dataset was generated using the Matlab program ConceptDriftData.m⁴.

• **Smtp (KDDCUP99) dataset** is the original KDD Cup 1999 dataset⁵ contains 41 attributes (34 continuous, and 7 categorical), however, they are reduced to 4 attributes (service, duration, src_bytes, dst_bytes) as these attributes are regarded as the most basic attributes, where only ”service” is categorical. Here, only ”smtp” service data is used. The original dataset has 3,925,651 attacks 80.1% out of 4,898,431 records. A smaller set is forged by having only 3,377 attacks 0.35% of 976,157 records, where attribute ”logged in” is positive. From this forged dataset 95,156 ”smtp” service data is used to construct the Smtp (KDDCUP99) dataset. The stream is stationary but contains outliers.

• **Soil moisture and temperature dataset** contains air temperature and soil temperature from the Soil Climate Analysis Network (SCAN) site 2026, ”Walnut Gulch”, Soil Moisture Percent: SMS.I-1.-8, Temperature Soil Temperature Observed: STO.I-1.-8. We have selected the data from the United State department of agriculture⁶ as follows:
  - Report: Soil moisture and temperature(1999-03-19),
  - Time: Hourly,
  - Format: csv,
  - Year: 2001 (Calendar year-all days).

This dataset contains concept drift and few outliers.

### 2.5.3 Results

Figures 2.4, 2.5 and 2.6 show the results on the Gaussian, Smtp, and Soil datasets, respectively, in the form of box plots as previously described. (In the figures GPCM stream appears shortened asGPCMs for space reasons.)

For the Gaussian dataset, the number of clusters was 4, and the input data were organized as follows: For OFCM, SPFCM and eGKPCM they were split in chunks of size $s = 800$. For GPCM stream the same size was used for initializing the centroids, then the chunk size was reduced to $s = 1$ for by-object learning.

⁴https://github.com/gditzler/ConceptDriftData  
⁵http://archive.ics.uci.edu/ml/datasets/kdd+cup+1999+data  
⁶https://wcc.sc.egov.usda.gov/nwcc/site?sitenum=2026
Figure 2.4: Quality indices of the fuzzy clustering methods on the Gaussian dataset: (a) PC, (b) PE, (c) XB.
Figure 2.5: Quality indices of the fuzzy clustering methods on the Smtp (KDDCUP99) dataset: (a) PC, (b) PE, (c) XB.
Figure 2.6: Quality indices of the fuzzy clustering methods on the Soil moisture and temperature dataset: (a) PC, (b) PE, (c) XB.
Figure 2.4 shows that according to the PC and PE indexes the fuzzy possibilistic models outperform the probabilistic ones when dealing with non-stationary data. According to XB the GPCM model experiences a higher variability of results. This is the only method that has the time resolution of one observation, so this behavior is understandable and may indicate a higher sensitivity of the XB index with respect to the other two.

Regarding the experiments on the Smtp (KDDCUP99) dataset, here the chunk size selected is $s = 3,000$ and the number of clusters was fixed at 4.

Figure 2.5 shows the performance drop in OFCM and SPFCM in the presence of outliers in data streams. eGKPCM shows a very low separation between clusters and its XB value was not even included in the figure. According to the three indexes, GPCM stream appears to be the most suitable method in the presence of outliers.

Finally, with the Soil moisture and temperature dataset, the selected chunk size is $s = 800$ and the number of clusters was fixed at 4.

The PC and PE indexes do not reveal strong differences between the four methods, while XB seems to indicate GPCM stream as the method that achieves the best top performance, although with the largest variation; again this is understandable, given that it is the only method that deals with the stream object by object rather than chunk by chunk, so pointwise variations are not averaged out within chunks.

2.6 Summary and discussion

2.6.1 Summary of methods properties

In table 2.1 we summarize the properties of some clustering algorithms, of both the fuzzy and non-fuzzy types, showing the computational complexities of each one. In the table, the assumption is made that the number of iterations required for termination is fixed. The following parameters are used: $n$ is the number of objects in the $s$-dimensional chunk; $c$ is the number of clusters; $m$ is the number of micro-clusters for Den-Stream; $c_p$ clustering results of the DUCstream in bits; $g$ is the number of grid points in the grid-based clustering algorithms.

The columns contain the acronym of the clustering method (Method), the basic clustering technique on which the stream method is based (Type), the input organization, by chunks or online (Input), the expected shape of clusters, the degree of sensitivity to concept changes, and finally the time complexity.

About computational complexity, we remark that the convergence of all these optimization procedures depends on the data, so the time required to reach a stable state is difficult to estimate in general terms. In addition, computational complexity is defined as an asymptotic limit. This
means that it is only indirectly relevant since, even when re-clustering is performed, the data are always split into smaller chunks. The details of the implementation (for instance, compiled versus interpreted software; integer versus float; single precision versus double precision; data acquisition and storage overhead) and the selected chunk length may be more important than the asymptotic behavior.

To help in the selection of the most suitable method, the following guidelines may be adopted:

- When irregular cluster shapes are expected, density-based and grid-based method may be preferred.
- If non-stationarity is expected, methods with a higher ability to track concept change are required. For strongly non-stationary streams, methods able to work by-pattern may ensure faster tracking.
- Soft clustering (fuzzy or possibilistic) is to be preferred when concept drift is prevalent over concept shift and the user wants to avoid discontinuities in cluster memberships. This may happen with crisp clustering when observations suddenly switch from a cluster to a neighbouring one.
- Asymmetric clusters can be dealt with by medoid-based methods.
- To counterbalance the effect of working in high dimensionality \( d \) [65, 101], one strategy consists in using order statistics rather than means [100]. This translates again into a preference for medoid-based methods.
- Computational complexity should be used only as a general estimation; for time-critical problems, e.g., in control applications with real-time requirements, software profiling on the actual data should be used.

While a good practice in data science is to try different methods with the same dataset, we remark that the literature also offers ensemble or consensus clustering frameworks [36, 74] that may be adapted to the stream setting to achieve more consistent clustering results.

### 2.6.2 Other approaches

In this chapter, we have focused mainly on surveying different types of fuzzy stream clustering methods. The literature about source change detection and tracking is vast and spans several different types of approaches. However, a large part of this literature is not directly applicable to clustering, since it focuses on supervised tasks and relies on a well-defined loss function that makes it easier to compute indicators of change.
The methods that we have described are all heuristic procedures. In principle, a number of methods with more solid theoretical grounds can be used instead of the fuzzy/possibilistic formalism. In the following discussions, we highlight some reasons why in the case of data streams these more principled solutions may, in practice, be less convenient.

Among the competing approaches available for the unsupervised case, we can distinguish between probabilistic and statistical methods. **Probabilistic** methods for source tracking, change detection, and outlier detection/rejection are based on explicit models of data density, for instance, mixture models (Gaussian or other). Their advantage is clearly provided that (1) the model is appropriate and (2) its parameters can be fitted reliably. Unfortunately, the validity of condition (1) is difficult to assess, unless prior knowledge is available. Failing that, the value of a mixture-based method is similar to that of a heuristic procedure, so it becomes essentially equivalent to a fuzzy probabilistic model. Coming to requisite (2), the main fitting algorithm, Expectation-Maximization (EM), may not converge very quickly [87]. In this case, too, heuristic algorithms may become equivalent, but simpler. Stochastic-approximation-type versions do exist [22]. However, having to fit more parameters unavoidably requires more data than optimization of simpler cluster models. This limits its applicability to cases where the “local i.i.d.” assumption holds on extended intervals, i.e., cases with slow concept change.

One particular approach within the probabilistic family is Bayesian filtering. The Kalman filter has been employed in this task [69], resulting in a speed-up of one order of magnitude with respect to EM iterations. However, this is exactly applicable only when source change follows a linear dynamics plus Gaussian evolution. Discontinuity, i.e., concept shift, is difficult to deal with, unless, again, prior knowledge about the possible evolutions is available. In this case, one option (with drawbacks similar to the general Bayesian filtering approach) is particle filtering [121] on an explicit model of shift, e.g., a Markov model.
Regarding statistical methods, these have the advantage that they may be non-parametric, that is, they may not require the identification of a probabilistic model. The statistics used are indicators of change computed on (suitably-sized) samples. A basic but common example of this class of methods is the CUSUM criterion [92], as well as other forms of sequential analysis [40]. Concentration inequalities can also be used (this is also done in TRAC-STREAMS with a Chebychev bound), but these have the disadvantage of becoming trivial, and therefore of little use, when the quantity of data considered is not sufficient.

Other, more sophisticated statistics are based on permutation tests [45], multi-dimensional variants of statistical tests [37], distances between density estimates [9]. These are often computationally expensive, requiring the construction of combinatorial graphs or the computation of complete density estimates.

Some of these indicators are sensitive only to some types of changes. For instance, CUSUM in its basic form only detects unidirectional changes in a scalar estimator. Finally, the focus of most of these tests is in detecting concept shift, not a trend that indicates concept drift.

It should be noted that all of the cited methods compare stationary distributions, and usually, the theory behind them does not cover damping factors or forgetting weights to model explicitly the gradual loss of validity in time. This makes the “local i.i.d.” assumption even more critical.

As a conclusion, despite being only heuristically motivated, fuzzy approaches actually constitute an interesting solution to the problem, especially under resource and time constraints, and in the absence of detailed knowledge about the source and its evolution.
Chapter 3

Graded Possibilistic \( c \) Means stream clustering method

This chapter is based on the paper [3]:


The chapter, after a detailed description of relevant issues of the clustering methods for data streams, describes the Graded Possibilistic \( c \) Means (GPCM) clustering method [84] and the followed approach for continuous learning needed for tracking evolving data streams. Two versions of the proposed Graded Possibilistic \( c \) Means stream clustering (GPCM stream) method, batch and online, are then presented, with together their experimental validations.

3.1 Clustering Non-Stationary Data Streams

Clustering streams requires tackling related, but different, problems: Handling unbounded, possibly large data; Detecting model changes (drift and shift); Adapting to model changes. These are often treated independently in the literature.

Model shift, either for rejection or tracking, has been extensively studied, also under the name of change detection or change point identification. The key to detect model differences is measuring the fit of observations. An observation that does not fit a given model is either called novelty [23] or outlier [50] depending on the focus of the inquiry. Individual outliers can be
detected or rejected by evaluating their estimated probability or membership and comparing it to a threshold, using distance-based or density-based criteria [66], or simply not taken into account by using robust methods [56]. Alternatively, it is possible to gather a new sample and apply sequential statistical tests for comparing empirical distributions [11]. Regarding fuzzy methods, techniques have been proposed mainly for robustness [64], although some approaches directly include measures of outlierness [88] that can indicate the inadequacy of a given current model.

Model drift is a much more difficult issue, usually simply tackled by continuous learning. This brings us to the third problem, learning strategies for dynamically changing models. Continuous learning can be done either by recomputing, if affordable, or by incremental updates. Most fuzzy clustering models are fit with a batch training procedure, owing to their derivation from (crisp) \( k \) Means. This batch optimization, iterative and prone to local minima, makes them unsuitable for datasets of very large/virtually unbounded cardinality. However, the same prototype-based representation lends itself very well to alternative training methods, e.g., online or “by pattern” [83], which are naturally suitable for incremental updating. Due to the non-stationarity, it is not possible to resort to the extensive literature about the convergence of stochastic approximation or related methods [75]. However, a measure of model inadequacy can be used to modulate the required amount of update, avoiding unnecessary waste of computational resources especially when the data are large in size. In addition, it might be appropriate to dynamically change the number of centroids. In [95] this was addressed with a cross-validation approach. However, we didn’t take this problem into consideration.

We consider the case where the data are represented by numerical feature vectors. In general, they will be multi-dimensional, even in the case of scalar-valued time series where time-lag encoding is used. We assume that the data are generated independently by some underlying probability distribution, but in the long run, the distribution may change. As long as the underlying distribution doesn’t “change much”, it can be considered stationary. However, for sufficiently long observation periods this approximation is no longer valid. The extent of validity of a stationary approximation depends on the rate and intensity of change of the source distribution, which is clearly difficult to estimate. In this work we, don’t consider this problem, but provide methods for two different scenarios.

The goal of the analysis is to summarize these data by (fuzzy) clustering, with a process that should learn continuously from the input patterns as they arrive. The data are stored in a sliding window \( W \) of constant size \( w \geq 1 \) which is updated every \( s \) observations by deleting the oldest \( s \) patterns and adding the new ones, so that, at each time, the current \( W \) has an overlap of \( w - s \) patterns with the previous one. We collect the \( s \) incoming observations in a probe set \( S \) before incorporating them in the window; the probe set is used to adaptively tune the learning parameters.

We assume that, within \( W \), the data can be considered independent and identically distributed i.i.d. As anticipated in the introduction, we will focus on the following two scenarios:
• The source change rate so slow that $W$ is sufficient to infer the clustering structure;
• The rate is so high that $W$ is not sufficient to perform a complete clustering, and an incremental procedure is required.

Accordingly, in this work we consider the cases $w > 1$, $s > 1$ (batch learning, batch density estimate) and $w = s = 1$ (online learning, online density estimate).

### 3.2 The Graded Possibilistic c-Means Model

#### 3.2.1 Central fuzzy clustering

A set of $c$ fuzzy clusters is represented by means of $c$ centroids

$$y_j, \ j \in \{1, \ldots, c\}.$$  (3.1)

Each cluster has associated a fuzzy cluster indicator (or membership) function

$$u_j(x_l) \in [0, 1] \subset \mathbb{R}.\quad (3.2)$$

Memberships depend on distances between the centroids and the observations $x_l (l \in \{1, \ldots, n\})$.

Many methods are derived as the iterative optimization of a constrained objective function [15]. Different methods are obtained by adding different constraints and penalty terms to the main objective, the mean squared distance or distortion:

$$D = \frac{1}{n} \sum_{l=1}^{n} \sum_{j=1}^{c} u_{lj} \|x_l - y_j\|^2.\quad (3.3)$$

The definition of centroids obtained from the necessary minimum condition $\nabla D = 0$ turns out to be common to all methods:

$$y_j = \frac{\sum_{l=1}^{n} u_{lj} x_l}{\sum_{l=1}^{n} u_{lj}}.\quad (3.4)$$

One exception to this schema is FCM where $u_{lj}^m$ replaces $u_{lj}$.

Usually constraints are imposed on the sum of all membership for any given observation $x_l$,

$$\zeta_l = \sum_{j=1}^{c} u_{lj}.\quad (3.5)$$

The value $\zeta_l$ can be interpreted as the total membership mass of observation $x_l$. 


3.2.2 Maximum-entropy clustering

The Maximum Entropy (ME) approach \cite{98, 99} imposes $\zeta_l = 1$, so we are in the “probabilistic” case, where memberships are formally equivalent to probabilities.

The objective is the Lagrangian
\[
J_{\text{ME}} = D + H + U, \quad (3.6)
\]
where
\[
H = \sum_{l=1}^{n} \sum_{j=1}^{c} \beta_{l} u_{lj} \log u_{lj}, \quad (3.7)
\]
is an entropic penalty with weight $\beta$ and
\[
U = \sum_{l=1}^{n} \lambda_l (1 - \zeta_l), \quad (3.8)
\]
is the sum of all constraint functions, with Lagrange multipliers $\lambda_l$. The necessary minimum condition $\nabla J_{\text{ME}} = 0$ yields
\[
u_{lj} = e^{-d_{lj}/\beta_{l}}. \quad (3.9)
\]
The associated optimization procedure proposed for this model includes an “annealing” schedule on $\beta$. At each step of this procedure, previously overlapping clusters may separate when a sufficiently low $\beta$ is reached. The value $\beta = 0$ corresponds to crisp clustering.

3.2.3 Possibilistic clustering

The Possibilistic $c$-Means (PCM) \cite{12, 71, 72} does not impose any constraint on $\zeta_l$, so memberships are not formally equivalent to probabilities; they represent degrees of typicality.

The objective is:
\[
J_{\text{PCM}} = D + W. \quad (3.10)
\]
We focus on the version presented in \cite{72}, here labeled PCM-II, where the following penalty is used:
\[
W = \sum_{l=1}^{n} \beta_{j} \sum_{j=1}^{c} (u_{lj} \log u_{lj} - u_{lj}). \quad (3.11)
\]
The condition $\nabla J_{\text{PCM}} = 0$ yields
\[
u_{lj} = e^{-d_{lj}/\beta_{j}}. \quad (3.12)
\]
In spite of the similarity to the previous method, we note that in this case the values of the coefficient $\beta_{j}$ in the exponent are different for each centroid. This is because in this case these
coefficients are model parameters, playing a role in the representation of data as cluster widths, while in the ME approach the unique coefficient $\beta$ is an optimization parameter, which matters mainly during model building (learning).

As a consequence, in [71] two heuristics for setting $\beta_j$ are suggested, starting from the centroids and memberships obtained from a run of fuzzy $c$ means, used as an initialization:

$$\beta_j = K \frac{\sum_{j=1}^{c} u_{ij}^m d_{ij}}{\sum_{j=1}^{c} u_{ij}^m},$$  

(3.13)

where $K = 1$ and $m$ is not very different from 2;

$$\beta_j = \frac{\sum_{j \in (\Pi_l)_a} d_{ij}}{|(\Pi_l)_a|} = \frac{\sum_{u_{lj} > a} d_{lj}}{\sum_{u_{lj} > a} 1},$$  

(3.14)

where $(\Pi_l)_a$ is an $a$-cut of $\Pi_l$, or $(\Pi_l)_a = \{\forall j \ u_{lj} | u_{lj} > a\}$.

Note that equation (3.12) is a special case of equation (3.9) with $\zeta_l$ replaced by 1 $\forall l$, whereas equation (3.7) is a special case of equation (3.11) with $\beta_j = \beta \ \forall j$.

An appropriate initialization is required in PCM-II, since it is extremely prone to falling into local minima. This problem is somewhat alleviated in the next method.

### 3.2.4 Graded possibilistic models

Starting from the last observations, we can generalize the membership function as follows:

$$u_{lj} = \frac{v_{lj}}{Z_l},$$  

(3.15)

where the free membership $v_{lj} = e^{-d_{lj}/\beta_j}$ is normalized with some term $Z_l$, not necessarily equal to $\zeta_l$. This allows us to add a continuum of other, intermediate cases to the two extreme models just described, respectively characterized by $Z_l = \zeta_j = \sum_{j=1}^{c} v_{lj}$ (probabilistic) and $Z_l = 1$ (possibilistic).

This can be done in several ways. In this work we use the following formulation:

$$Z_l = \zeta_l^\alpha = \left( \sum_{j=1}^{c} v_{lj} \right)^\alpha, \quad \alpha \in [0, 1] \subset \mathbb{R}.$$

(3.16)

In [84] another possible formulation was used. The parameter $\alpha$ controls the “possibility level”, from a totally probabilistic ($\alpha = 1$) to a totally possibilistic ($\alpha = 0$) model, with all intermediate cases for $0 < \alpha < 1$. 

43
Figure 3.1: Comparison between membership profiles: (a) possibilistic model, $\alpha = 0$; (b) graded model, $\alpha = 0.8$; (c) probabilistic model, $\alpha = 1$. 
For $\alpha = 1$, the representation properties of the method coincide with those of ME. When $\alpha = 0$, they are equivalent to those of PCM-II. In the intermediate cases, as soon as $\alpha > 0$, there is a degree of competition between clusters, as in probabilistic models, but memberships eventually vanish for points sufficiently far away from the centroids, as in the possibilistic case.

Figure 3.1 illustrates the effect of $\alpha$ in the case of two one-dimensional centroids. The plots show the membership of a point, whose position $x$ varies from $-1$ to $+1$, to each cluster. Three different values of $\alpha$ are shown.

This behavior alleviates the problem of coincident centroids, if so desired [72], and this approach has better convergence than PCM-II for $\alpha$ sufficiently larger than 0. However, at the same time the outlier insensitivity that characterizes PCM and PCM-II is preserved, helping with a more precise centroid positioning [84], and can be modulated by the user. This ability to spot outliers is the key to the proposed measure of outlierness.

### 3.3 Outlierness measurement through graded possibilistic memberships

The nature of membership functions suggests a characterization of outliers similar to Davé’s Noise Clustering model that was used in the context of robust clustering [25]. Given a trained clustering model, i.e., a set of centroids and a set of cluster widths $\beta_j$, we exploit the properties of the possibilistic memberships to evaluate the degree of outlierness. We define outlierness as the membership of an observation to the concept of “being an outlier” with respect to a given clustering model. Differently from other approaches based on analyzing pattern-centroid distances [120], the graded possibilistic model used in this work provides a direct measure of outlierness. We propose to measure the total mass of membership to clusters $\zeta_l$, which, by definition of the graded possibilistic model, does not necessarily equal 1, and measure whether and how much it is less than 1. Quantitatively, we define an index $\Omega$ as follows:

$$\Omega(x_l) = \max \{1 - \zeta_l, 0\}.$$  \hspace{1cm} (3.17)

Outlierness can be modulated by an appropriate choice of $\alpha$. Low values correspond to sharper outlier rejection, while higher values imply wider cluster regions and therefore lower rejection. For $\alpha = 1$ the model becomes probabilistic and loses any ability to identify or reject outliers.

We observe that $\zeta_l = \sum_j u_{lj} \in (0, c)$. However:

- values $\zeta_l > 1$ are typical regions well covered by centroids;
- but $\zeta_l \gg 1$ is very unlikely for good clustering solutions without many overlapping clusters;
finally, $\zeta_l \ll 1$ characterizes regions not covered by centroids, and any observation occurring there is an outlier.

The index $\Omega$ is defined as the complement to one of $\zeta_l$, with negative values clipped out as not interesting. Figure 3.2 shows a level plot of outlierness for the clusters of a dataset. If we set a threshold on $\Omega$, each contour marks the borders between the regions of inliers and outliers for a given threshold value.

The outlierness index is a pointwise measure, but it can be integrated to measure the frequency of outliers. For crisp decision-making, a point could be labeled as outlier when $\Omega$ exceeds some threshold. It is therefore easy to count the proportion (frequency) of outliers over a given set of probe points $S$.

However, we take advantage of the fact that $\Omega$ expresses a fuzzy concept, and rather than simply counting the frequency we can measure an outlier density $\rho \in [0, 1)$ defined in one of the following ways.

$$\rho_M = \frac{1}{|S|} \sum_{l \in S} \Omega(x_l)$$

The density $\rho_M$ accounts for both frequency and intensity, or degree of anomaly, of outliers. A high number of borderline outliers is equivalent to a lower number of stronger outliers, provided their mean value is the same. To give more emphasis to the case where some observation have a
higher outlierness, an alternative definition can be used:

$$\rho_{\text{RMS}} = \frac{1}{|S|} \sqrt{\sum_{l \in S} (\Omega(x_l))^2}.$$  \hspace{1cm} (3.19)

The definition $$\rho = \rho_{\text{RMS}}$$ will be adopted in the experiments.

### 3.4 Learning regimes

We distinguish between three different situations, corresponding to three possible learning regimes.

1. **Concept drift.** The source is stationary or changing smoothly and slowly ($$\Omega = 0$$, density is low). Action to be taken: The model should be incrementally updated to track possible variations. We can call this the *baseline learning regime*.

2. **Outliers.** One or few isolated observations are clearly not explained by the model, which means that they have outlierness ($$\Omega > 0$$, density is low). Action to be taken: Incremental learning should be paused to avoid skewing clusters with atypical observations (*no-learning regime*).

3. **Concept shift.** Several observations have outlierness ($$\Omega > 0$$, density is high). Action to be taken: The old clustering model should be replaced by a new one. This is the *re-learning regime*.

The learning depends on a parameter $$\theta$$ that balances between stability ($$\theta \approx 0$$, model stays the same) and plasticity ($$\theta \approx 1$$, model changes completely), so it is possible to modulate this parameter as a function of outlier density, so that the three learning regimes (baseline, no learning, and re-learning) can be implemented.

Since learning regimes are yet another fuzzy concept, rather than splitting them into clear-cut regions, in our experiments, we employed a smooth function that is controlled by parameters: The user should interactively select their values to obtain the desired profile. This procedure is similar to defining the membership function for a linguistic variable. The function we used is the following:

$$\theta = 1 + \theta_0 \exp \left( -\frac{\rho}{\tau_1} \right) \exp \left( -\left( \frac{\rho}{\tau_2} \right)^\gamma \right).$$ \hspace{1cm} (3.20)

• $$\theta_0$$ is the baseline value of $$\theta$$, used when new data are well explained by the current model (baseline learning regime).
• \( \tau_1 \) is a scale constant, analogue to a time constant in linear dynamical system eigenfunctions, that determines the range of values for which the baseline learning regime should hold.

• \( \tau_2 \) is a scale constant that determines the range of values for which the re-learning regime should hold.

• \( \gamma \) is an exponential gain that controls how quick the relearning regime should go to saturation, i.e., to \( \theta \approx 1 \).

In the transition between baseline learning and re-learning, this function has a valley that brings the value of \( \theta \) close to zero, implementing the no-learning regime.

Figure 3.3 shows plots of the learning function for various values of its parameters.

Figure 3.3: The learning parameter \( \theta \) as a function of \( \rho \) for various values of its parameters.
3.5 Graded Possibilistic \( c \) Means stream clustering method

In this section we outline the two learning algorithms proposed in this thesis that are a batch and an online versions Graded Possibilistic \( c \) Means stream clustering (GPCM stream) method \(^1\) that is based on the deterministic annealing optimization algorithm. We term the cost function “possibilistic entropy” because actual probabilities are replaced by possibilistic membership.

In the algorithms proposed in this section, the degree of possibility \( \alpha \) is assumed to be fixed. This quantity incorporates the a-priori knowledge about the amount of outlier sensitivity desired by the user.

In batch learning, at each time \( t \) we train the clustering model on a training set (window) \( W_t \) of size \( w \) and evaluate \( \rho \) on the next \( s < w \) observations (set \( S_t \)). Then we compute \( W_{t+1} \) for the next step by removing the \( s \) oldest observations and adding \( S_t \), so that the training set size remains constant. Finally, the amount of learning required is estimated by computing \( \theta(\rho) \) according to equation (3.20).

From an optimization or learning perspective, we are estimating the true (expected) objective function on the basis of a set \( W \) of \( w \) observations, that we use to compute a sample average. The batch process is initialized by taking a first sample \( W_0 \) and performing a complete deterministic annealing optimization on it with an annealing schedule \( B = \{ \beta_1, \ldots, \beta_b \} \). In subsequent optimizations, the annealing schedule is shortened proportionally to the computed value of \( \theta(\rho) \in [0, 1] \): When \( \theta = 1 \) the complete \( B \) is used (\( |B| = b \) optimization steps); when \( \theta = 0 \) no training is performed (0 steps); when \( 0 \leq \theta \leq 1 \) a corresponding fraction \( B_\theta \) of the schedule \( B \) is used, starting from step number \( \lceil b \cdot \theta \rceil \) up to \( \beta \) (that is, \( \lfloor b \cdot (1 - \theta) \rfloor \) steps in total). The updating rule for a generic centroid (non-stationary data streams) is the following:

\[
y_j(t+1) = y_j(t) + \eta \frac{\sum_{l=1}^{w} u_{lj} (x_l - y_j(t))}{\sum_{l=1}^{w} u_{lj}}, \quad (3.21)
\]

\[
= (1 - \eta) y_j(t) + \eta \frac{\sum_{l=1}^{w} u_{lj} x_l}{\sum_{l=1}^{w} u_{lj}}. \quad (3.22)
\]

For stationary data streams the distribution of any sample \( W(t) \) is constant w.r.t. \( t \), and therefore its weighted mean \( \frac{\sum_{l=1}^{w} u_{lj} x_l}{\sum_{l=1}^{w} u_{lj}} \) is also constant. In this case

\[
y_j(t \to \infty) \rightarrow \frac{\sum_{l=1}^{w} u_{lj} x_l}{\sum_{l=1}^{w} u_{lj}}. \quad (3.23)
\]

With fixed \( \eta \), equation (3.21) computes an exponentially discounted moving average.

\(^{1}\)A MATLAB implementation of GPCM stream is available at: https://it.mathworks.com/matlabcentral/fileexchange/64318-onlinegradedpossibilisticclustering. Appendix A of this thesis describes its implementation.
Table 3.4: Batch GPCM stream clustering

| init: Select $\alpha, B = \{\beta_1, \ldots, \beta_b\}, \theta = \theta(\rho)$ |
| Read first $w$ observations into $W_0$ |
| Learn clustering model from $W_0$ using annealing schedule $B$ |

| loop: (for each time $t : 1, \ldots, \infty$) |
| Read next $s$ observations into $S_t$ |
| Use (3.17) to compute $\Omega$ for all observations in $S_t$ |
| Use (3.19) to compute $\rho$ |
| Use (3.20) to compute $\theta$ |
| Discard $s$ oldest observations from $W_{t-1}$ |
| Update $W_t \leftarrow [W_{t-1} \ S_t]$ |
| Learn clustering model from $W_t$ using annealing schedule $B_\theta$ |

Figure 3.4: Batch GPCM stream clustering

In summary, for the batch case we use $\theta$ to modulate the number of annealing steps and, consequently, the scale parameter $\beta$. The optimization is longer, and starts with a higher coverage (higher $\beta$), in the re-learning regime; it is shorter and more localized in the baseline learning regime; it does not occur at all (zero steps) in the no-learning regime. Figure 3.4 outlines the batch learning algorithm.

Now for real-time learning, we provide an online learning method. This case can be modeled as a limit case of the batch method. At each time $t$ we train the clustering model on a training set $W_t$ of size 1, i.e., one observation, and evaluate $\rho$ on the next $s = 1$ observation forming the “set” $S_t$. Then we compute $W_{t+1}$ for the next step by replacing the single observation with that in $S_t$. The amount of learning required is estimated by computing $\theta(\rho)$ according to equation (3.20). In this case the updates are incremental and therefore for $\rho$ as well we propose an incremental computation according to the following discounted average formula:

$$\rho_t = \lambda \Omega_t + (1 - \lambda) \rho_{t-1}. \quad (3.24)$$

In this case, as well, the process is initialized by taking a first sample $W_0$ and performing a complete deterministic annealing optimization on it with an annealing schedule $B = \{\beta_1, \ldots, \beta_b\}$. However, after the initial phase $w = 1$, the estimate of the objective function cannot be obtained by approximating an expectation with an average. So we resort to a stochastic approximation procedure [96]. This results in the following iterative update equations:

$$y_j(t+1) = y_j(t) + \eta_t u_{lj}(x_l - y_j) \quad (3.25)$$

for each centroid, $j = 1, \ldots, c$, with learning step size $\eta_t$. The update equation for the memberships is still given by equations (3.15), and (3.16).
init: Select $\alpha, B = \{\beta_1, \ldots, \beta_b\}, \theta = \theta(\rho), \eta_0$

Read first $w$ observations into $W_0$
Learn clustering model from $W_0$ using annealing schedule $B$

loop: (for each time $t : 1, \ldots, \infty$)
Read next observation
Use (3.17) to compute $\Omega$ for current observation
Use (3.20) to compute $\theta$ with $\rho = \Omega$
Use (3.26) to compute $\beta$
Learn clustering model from current observation using Eq. (3.25) with learning step $\eta_t = \eta_0 \cdot \theta$

Figure 3.5: Online GPCM stream clustering

$\beta_j$ is updated [72] according to:

$$\beta_j = \frac{\sum_{i=1}^{N} u_{ij} d_{ij}}{c \sum_{i=1}^{N} u_{ij}}, \quad j = 1, \ldots, c.$$  (3.26)

Differently from the batch case, after the initialization step a deterministic annealing schedule is not needed; rather, we have a stochastic annealing step size $\eta_t$. There are well-known conditions on $\eta_t$ for convergence in the stationary case [96]:

$$\sum_{t=1}^{\infty} \eta_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty.$$  (3.27)

However, these conditions obviously do not hold in the non-stationary case, and this topic has not been thoroughly studied in the literature because the conclusions depend on the specific problem setting.

The strategy adopted in this work is to have the step size $\eta$ be directly proportional to $\rho = \Omega$, i.e., $\eta = \eta_0 \cdot \rho$ for a user-selected constant $\eta_0$. An averaging effect is obtained through the stochastic iterative updates. In this way, after initialization, the intensity of updates depends on the degree of outlierness of the current observation.

To avoid premature convergence, the possibility degree $\alpha$ is also made dependent on $\rho$, so as to increase centroid coverage when outliers are detected. The formula used is:

$$\alpha = \alpha_{\min} + \rho (1 - \alpha_{\min}).$$  (3.28)

Figure 3.5 sketches the online learning algorithm.
3.6 Experimental results

3.6.1 Synthetic dataset

Synthetic datasets containing concept drift (we select the Gaussian and electricity datasets) were generated using the Matlab program ConceptDriftData.m\(^2\) [26]. We have also integrated our model into a traffic flow management system [4]. The proposed work was used as a generative model to assess and improve the accuracy of a short term traffic flow forecasting model.

The Gaussian dataset already include concept drift, so outliers and concept shift were added by removing a number of observations in two parts of the data sequence. Discontinuities in the sequence are therefore introduced at 50% and 75% of the streams. In addition, the final 25% was shifted by adding an offset to all the data. Results are here shown for the Gaussian dataset. This includes four two-dimensional, evolving Gaussian with equal and known centers and spreads. After introducing discontinuities and shift, the data were remapped into \([0,1] \times [0,1]\). This procedure ensures that a ground truth is available at all times.

The dataset contains 2,500 observations. The parameters used for the experiments are listed in table 3.1.

Figure 3.6 shows the graphs of outlier density \(\rho\) in the batch (upper plot) and online (middle plot) cases. For reference, plots of the data taken at various stages are displayed in the bottom. The upper (batch) plot shows \(\rho = \rho_{\text{RMS}}\) computed on the probe set \(S\) at each iteration. The middle (online) plot shows \(\rho = \Omega\). During training this is evaluated at each pattern. For clearer display, average over the past few iterations, rather than instantaneous value, is shown in the figure. In both plots, vertical lines indicate discontinuities (concept shift).

The graphs show the effectiveness of the proposed indexes in indicating the conditions occurring in the stream at each point in time. From the bottom plots, it is evident that the first discontinuity occurs between similar configurations, so there is no actual concept shift; in fact, the graph in the batch case highlights that after the discontinuity clusters are closer to each other, so the concentration of data is higher, the clustering task is easier, and the outlier density decreases.

The second discontinuity produces instead a relevant change in centroid configuration, and this is evident in both graphs. However, the online version is much quicker to respond to the variation, as indicated by the more steeply increasing plot around the second discontinuity.

A comparative study was performed with two other methods. The first, used as a baseline, is a simple non-tracking k-means, trained once and used without updates. The second comparison is with a method with similar goals, called TRAC-STREAMS [88], which performs a weighted PCM-like clustering while updating by pattern.

\(^2\) Available under GPL at https://github.com/gditzler/ConceptDriftData.
Table 3.1: Parameters used in the experiments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Batch</th>
<th>Online</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training window size</td>
<td>$w$</td>
<td>200</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Probe window size</td>
<td>$s$</td>
<td>30</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Possibility degree</td>
<td>$\alpha$</td>
<td>0.7</td>
<td>0.7</td>
<td>(1)</td>
</tr>
<tr>
<td>Num. annealing steps</td>
<td>$b$</td>
<td>20</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>$\beta$ schedule</td>
<td></td>
<td>linear</td>
<td>linear</td>
<td>(2)</td>
</tr>
<tr>
<td>Starting value for annealing</td>
<td>$\beta_1$</td>
<td>0.05</td>
<td>0.05</td>
<td>(2)</td>
</tr>
<tr>
<td>Ending value for annealing</td>
<td>$\beta_b$</td>
<td>0.002</td>
<td>0.002</td>
<td>(2)</td>
</tr>
<tr>
<td>Density estimation function</td>
<td>$\rho(\Omega)$</td>
<td>$\rho_{\text{RMS}}$</td>
<td>$\Omega$</td>
<td></td>
</tr>
<tr>
<td>Coefficient for discounted avg.</td>
<td>$\lambda$</td>
<td>—</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\theta_0$</td>
<td>0.3</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>Parameters for computing $\theta$</td>
<td>$\tau_1$</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\tau_2$</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\gamma$</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

(1) For online: minimum value, maximum is 1.

(2) For online: only in the batch initialization phase.
To perform a comparative analysis, for each method we measured tracking performance by computing the distortion (mean squared error) with respect to the learned centroids and to the true centroids, that we have available since the dataset is synthetic. Figure 3.7 shows the tracking performance, obtained by plotting the absolute difference of distortions computed for the “true” and learned models. Apart from the abrupt discontinuity, quickly recovered in both the batch and online cases, the difference is extremely limited, and superior to the other two methods considered.

### 3.6.2 KDD99

The dataset provided for the 1999 KDD Cup was originally prepared by MIT Lincoln labs for the 1998 Defense Advanced Research Projects Agency Intrusion Evaluation Program, with the objective of evaluating research in intrusion detection, and it has become a benchmark dataset for evaluating intrusion detection algorithms.

The dataset contains several types of attacks and they fall into four main categories:
Figure 3.7: Tracking error. Absolute difference between distortion w.r.t. true centroids and distortion w.r.t. learned centroids. Top left: batch. Top right: online. Bottom left: TRACSTREAMS. Bottom right: $k$ means, statically trained.
• Denial of Service Attack (DoS): a type of attacks on a network that flood it with useless traffics by the consumption of resources and memories;

• Remote to Local (R2L): guessing password;

• User to root: the attacker log-in a normal user account on the system with intent to get to root access to the system;

• Probing: where an attacker scans a network to gather information or find known vulnerabilities.

The dataset used in our experiment is a smaller subset (10% of the original training set\(^3\)).

<table>
<thead>
<tr>
<th>Classes</th>
<th>Number of training samples</th>
<th>Number of test samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>97278</td>
<td>60588</td>
</tr>
<tr>
<td>Error</td>
<td>396743</td>
<td>250441</td>
</tr>
<tr>
<td>Total</td>
<td>494021</td>
<td>311029</td>
</tr>
</tbody>
</table>

Table 3.2: KDDCup99 dataset

Table 3.2 shows the number of errors and normal samples in both training and test set. We have treated the intrusion detection problem as a binary classification problem to distinguishing between attacks and normal samples. The binary classification case is considered. The original KDD Cup 1999 contains 41 attributes, however, as suggested by [21, 90] they are reduced to to 14 Attributes. Table 3.3 shows the the selected Attributes and table 3.4 shows their description.

In our experiment for initializing the centroids we have used training data with chunk size \(s = 1000\), then the chunk size was reduced to \(s = 1\) for online learning. The \(\text{mean}(\zeta_i)\) computed during training stage is used as a threshold to distinguish between attacks and normal samples. After the training stage, we start the outlier detection test, where dataset samples come as stream of chunks, where each chunk size is equal to 1. For each upcoming input chunk \(i\), \(\zeta_i\) is computed and compared to a threshold. In our experiment, we use the \(\Theta = \text{mean}(\zeta_i)\) as a threshold. However other choices, more or less restrictive, are possible based on the quantity and reliability of the training data. When a new chunk is presented to the model, if \(\zeta_i < \Theta\) it is considered as an attack. The sensitivity of detecting the attacks depend on the value of \(\alpha\). A high value of \(\alpha\)-means less detection to the attack. We have also to note that low value of \(\alpha\) may lead to high false positive rate (detect normal samples as attacks) so the value of \(\alpha\) must be carefully chosen. To evaluate the classification results we used the precision, recall, Receiver Operating Characteristic (ROC), and accuracy measure:

ROC curves are commonly used to present results for binary classification problems. Different from the precision-recall curve, the ROC curve represents a relation between sensitivity (RECALL) and specificity.

\(^3\)https://kdd.ics.uci.edu/databases/kddcup99/
<table>
<thead>
<tr>
<th>Attribute column ID</th>
<th>Attribute name</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Service</td>
</tr>
<tr>
<td>5</td>
<td>Source bytes</td>
</tr>
<tr>
<td>6</td>
<td>Destination bytes</td>
</tr>
<tr>
<td>7</td>
<td>Land</td>
</tr>
<tr>
<td>8</td>
<td>Wrong fragment</td>
</tr>
<tr>
<td>11</td>
<td>Failed login</td>
</tr>
<tr>
<td>14</td>
<td>Root shell</td>
</tr>
<tr>
<td>23</td>
<td>Count</td>
</tr>
<tr>
<td>24</td>
<td>Srv count</td>
</tr>
<tr>
<td>28</td>
<td>Srv rerror rate</td>
</tr>
<tr>
<td>29</td>
<td>Same srv error rate</td>
</tr>
<tr>
<td>30</td>
<td>Diff srv rate</td>
</tr>
<tr>
<td>36</td>
<td>Dst host same src port rate</td>
</tr>
<tr>
<td>39</td>
<td>Dst host srv serror rate</td>
</tr>
</tbody>
</table>

Table 3.3: KDDCup99 selected attributes.

\[
\text{Precision} = \frac{\text{True positive}(TP)}{\text{True positive}(TP) + \text{False positive}(FP)} \tag{3.29}
\]

\[
\text{Recall} = \frac{\text{True positive}(TP)}{\text{True positive}(TP) + \text{False negative}(FN)} \tag{3.30}
\]

\[
\text{Specificity} = \frac{\text{True negative}(TN)}{\text{True negative}(TN) + \text{False positive}(FP)} \tag{3.31}
\]

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.32}
\]

Figure 3.8 shows the Precision-recall and ROC curve of the results.

Unlike semi-supervised approach provided by [21,114], our model is a fully unsupervised model that is able to detect attacks in the KDD99 dataset without providing the labels. The average accuracy of our model is 92.94\% (no labels). This results is comparable with the semi-supervised algorithm SUN [114] that achieves 94\% (50\% pre-labeled data), and with [21] that achieves 83.76\% (30\% pre-labeled data), and 95.30\% (50\% pre-labeled data). The parameters used in our experiment are shown in Table 3.5.
<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>network service on the destination, e.g., http, telnet, etc.</td>
</tr>
<tr>
<td>Source bytes</td>
<td>number of data bytes from source to destination</td>
</tr>
<tr>
<td>Destination bytes</td>
<td>number of data bytes from destination to source</td>
</tr>
<tr>
<td>Land</td>
<td>1 if connection is from/to the same host/port; 0 otherwise</td>
</tr>
<tr>
<td>Wrong fragment</td>
<td>number of “wrong” fragments</td>
</tr>
<tr>
<td>Failed login</td>
<td>number of failed login attempts</td>
</tr>
<tr>
<td>Root sheel</td>
<td>1 if root shell is obtained; 0 otherwise</td>
</tr>
<tr>
<td>Count</td>
<td>number of connections to the same host as the current connection in the past two seconds</td>
</tr>
<tr>
<td>Srv count</td>
<td>number of connections to the same service as the current connection in the past two seconds</td>
</tr>
<tr>
<td>Srv rerror rate</td>
<td>% of connections that have “REJ” errors</td>
</tr>
<tr>
<td>Same srv error rate</td>
<td>% of connections to the same service</td>
</tr>
<tr>
<td>Diff srv rate</td>
<td>% of connections to different services</td>
</tr>
<tr>
<td>Dst host same src port rate</td>
<td>% of connections whose source port is the same to that of the current connection in Dst host count feature</td>
</tr>
<tr>
<td>Dst host srv error rate</td>
<td>% of connections that “SYN” errors in Dst host srv count feature</td>
</tr>
</tbody>
</table>

Table 3.4: KDDCup99 attributes description.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial chunk size</td>
<td>1,000</td>
</tr>
<tr>
<td>Test chunks size</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>0.96</td>
</tr>
<tr>
<td>Number of clusters</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 3.5: KDDCup99 dataset.
Figure 3.8: Precision-recall and ROC curve.
Chapter 4

Tracking time evolving data streams for short-term traffic forecasting

This chapter is based on the papers [1] and [4]:


It summarizes the state of the art in streaming data clustering and urban traffic modelling and then proposes the Robust Layered Ensemble Model (RLEM) for short-term traffic flow forecasting. The RLEM combines the GPCM clustering method and ensembles of Time Delayed Neural Networks. The experimental validation show that the RLEM gives an accurate forecasting of the traffic flow rates with outlier detection and shows a good adaptation to non-stationary traffic regimes. Given its characteristics of outlier detection, accuracy, and robustness, RLEM can be fruitfully integrated into real-time traffic flow management systems.

4.1 Introduction

Traffic forecasting is one of the most relevant problems related to data stream mining. It can be cast either in the long term, where forecasts are used to configure and validate road management
plans, or in the short term, for real-time decision-making. Short-term forecasting is the subject of this work.

Forecasting can be done with a system identification approach, often with macroscopic models [42]. Although it gives the most reliable results in the long-term forecasting problem, this approach is often not feasible for short-term forecasting, due to the inherent complexity of an accurate, first-principles model. The computation time required is often not compatible with the response time required.

The usual practice, in this case, is to use methods that forecast based on observations. This approach has developed out of the growing availability of data and, in parallel, of methods from data science, machine learning, and computational intelligence [109].

Methods presented in the recent literature can be categorized into parametric models [34,41,107] and non-parametric or hybrid models [104, 108, 123].

Many traffic forecasting approaches focus on the problem of freeway/motorway traffic forecasting in which the state of the road traffic is quite stable. In contrast, traffic forecasting in urban and network-scale areas is more complex because of the rapid change of traffic behaviour and of the limited availability of sensors that can cover the whole network.

Many approaches based on non-parametric models to tackle this problem have been proposed, such as multilayer perceptron with a learning rule based on a Kalman filter [110], wavelet-based neural network [38], fuzzy-neural model [118], ARIMA models [46], graphical-lasso neural network [106], multi-task neural network [39]; multi-task ensemble neural network [105], k-nearest neighbour non-parametric regression [119].

Most of these approaches are not meant to track changes in traffic behavior [109]. This is the main motivation for our proposal, which is described in the next section.

Since our method is centered around data stream clustering, we also survey some related work on this topic. Most algorithms in this area [5–7,52] focus on two aspects: detecting outliers without taking concept drift tracking into consideration; and clustering irregularly distributed data, which is a challenging direction of research in the field.

Data stream clustering methods can be of the batch type, collecting a number of instances and then performing clustering on these accumulated data [63, 89]. Other methods are single-pass, storing summaries of past data as they are scanned [49]. The strategies of these algorithms can be incremental [17] or divide-and-conquer [6]. Yet other algorithms alter the structure of the data themselves so that they can be more effectively accessed [122].

Some popular stream clustering methods are density-based. They aim to find clusters of arbitrary shape by modelling them as dense regions separated by sparse regions [8, 30].

While this class of algorithms is popular and effective, they all produce only crisp partitions with no direct way to evaluate the outlierness of incoming data. An alternative strategy is to use fuzzy
Several incremental fuzzy clustering algorithms based on Fuzzy c-means (FCM) [15] to track non stationarity in data streams have been developed. Under the fuzzy modelling paradigm, each data point belongs to a cluster to a degree specified by a membership value. In general, as no membership is exactly null, a data point belongs to all clusters with different degrees.

Popular incremental fuzzy clustering algorithms for data streams include Single-Pass FCM [54] and Online FCM (OFCM) [55]. Both process data chunk by chunk (by-pattern) and estimate centroids for entire dataset by extracting summary information from each chunk, but the ways they handle chunks are different.

In [76] two algorithms based on fuzzy c medoids (FCMD) [70], called Online Fuzzy C-Medoids (OFCM) and History based Online Fuzzy C Medoids (HOFCMD), are developed for clustering large relational datasets. In [85], it is shown that one medoid may not be sufficient to capture the underlying structure of a cluster. As a solution, in [112] an incremental fuzzy clustering approach called Incremental Multiple Medoids based Fuzzy Clustering (IMMFC) was proposed, which is based on the idea of OFCM and HOFCMD and includes a mechanism to select multiple medoids instead of a single one to represent each of the clusters in each chunk.

4.2 Methodology

Our choice has fallen on an auto-regressive approach which forecasts one step in the future after observing a suitable interval of past observations.

4.2.1 Data pre-processing

The observed data are samples of traffic flow on a road network. At any given time, each arc of the network graph contains a given number of vehicles. Flow is defined as the number of vehicles per unit time. An arc is characterized by a maximum number of vehicles, its capacity. When flow approaches this value, the traffic slows down and enters a stop-and-go regime. Once the capacity is reached, traffic is entirely congested. We will be mainly concerned with relative flow, the ratio of flow to the arc capacity. Flow is sampled at discrete time intervals of the order of some minutes.

As already mentioned, data are organized in chunks of observations corresponding to a time-lag vector. To forecast $f_t^a$, the traffic flow on arc $a$ at time $t$, a vector of length $T$ (the lag period) is used to represent a given chunk:

$$\mathbf{x} = [f_{t-T}^a, f_{t-T+1}^a, \ldots, f_{t-1}^a].$$  \hspace{1cm} (4.1)
4.2.2 Forecasting model issues

The design of auto-regressive methods requires solving the following issues.

**Lag period** – Proper selection of the lag period $T$, the size of the chunks, is crucial because it affects the correct representation of the data stream source. If the lag period is chosen too small, then we will not be able to distinguish between the time-lag vectors in the vector space [20]; hence, the prediction process will be practically impossible because data does not carry enough valuable information. If the lag period is chosen too large, measurement will refer to times which are too far to have a significant correlation with the present, and therefore they will be irrelevant and act as noise [61].

In this thesis we adopt the minimum of the time delayed mutual information as an estimation of the time-lag [35]:

$$S(\tau) = -\sum_{ij} p_{ij}(\tau) \ln \frac{p_{ij}(\tau)}{p_i p_j}, \quad (4.2)$$
where for some partition of the real line into intervals:

- $p_i$ is the probability to find a time series value in the $i$-th interval,
- $p_{ij}(\tau)$ is the joint probability that an observation at any time $t$ falls into the $i$-th interval and the observation at time $t + \tau$ falls into the $j$-th one.

Unlike the autocorrelation function, the mutual information takes into account also nonlinear correlations. If the time delayed mutual information exhibits a marked minimum, then $T = \arg\min_\tau S(\tau)$ is a good candidate for a suitable time delay. The obtained values are then confirmed by checking them against domain knowledge.

Note that if the minimum is not sufficiently prominent, then another method should be used. In the case of road traffic, the “memory” of the system is limited and this problem did not occur in our experiments.

**Training set size** – This refers to the number of observation patterns that will be used to train the forecasters. This is usually not under the control of the designer, but in the problem, at hand, the availability of data has been found to be sufficient.

**Outliers handling** – Learning patterns with a different behavior using the same model tends to reduce the model’s performance. This can occur for both diversity in the operating conditions, in the presence of a stationary source, and changes in the underlying source itself (concept drift and shift), in the non-stationary case.

Accordingly, the proposed model focuses on two strategies: learning an ensemble of locally specialized models, and explicitly measuring outlieriness.

### 4.2.3 Robust Layered Ensemble Model

The proposed *Robust Layered Ensemble Model* (RLEM) for short-term traffic forecasting consists of two layers as shown in figure 4.1 and is able to track the changes in data streams such as traffic flows, and to use this information to improve the forecasting accuracy.

The first layer of RLEM consists in a fuzzy clustering process having as its goals to cluster traffic flow chunks into $c$ fuzzy clusters, where chunks with high membership to the same cluster represent similar temporal patterns, and at the same time to measure the outlieriness degree of each chunk, and consequently to measure the density of outliers.

To this aim we employ an incremental clustering process based on the Graded Possibilistic $c$-Means (GPCM) \cite{84} that is able to adapt to the changes in the traffic flow, by implementing a continuous learning that exploits the input chunks as they arrive. Intrinsic to this clustering method is a measure of outlieriness that provides information about the goodness of fit of each input chunk to the clustering model.
Figure 4.2: Diagram of the forecasting stage in the RLEM. See text for details on the quantities and on the operational blocks mentioned in the diagram.
In the second layer, an ensemble of a number of base learners acting as forecasters equal to the number $c$ of clusters is used, each of them specialized on a homogeneous region of the data space. This approach follows the mixture of local experts model proposed in [57].

To obtain the $c$ homogeneous regions of the data space needed for base learner training, we defuzzify the fuzzy segmentation performed by the first layer by assigning each chunk to the cluster where it has the highest membership (nearest-neighbour criterion). To implement the base forecasters, we employ Time-Delayed Neural Networks (TDNN) [31] trained with the error back-propagation algorithm. Other choices may be implemented as well. The TDNN model is simply a multi-layer perceptron neural network whose input is a time-lag vector. In this work, one-hidden-layer networks are used for this purpose. We will indicate the network topology by specifying just the number of input, hidden, and output units as a triplet, ni-nh-no, with the understanding that each layer is fully connected to the following and that hidden units are sigmoidal while output units are linear.

The measure of outlierness evaluated by the first layer is used in the second layer to assess and improve the forecasting accuracy.

In the following, we describe the specific clustering technique used.

### 4.2.4 The Graded Possibilistic c-Means

In central clustering, we have a training set of $n$ instances (random vectors) and $c$ clusters represented by means of their central points or centroids $y_j$. Many central clustering methods perform the minimization of an objective function [15], that usually is the expectation of the distortion:

$$D = \frac{1}{n} \sum_{l=1}^{n} \sum_{j=1}^{c} u_{lj} d_{lj},$$  \hspace{1cm} (4.3)

$$l = 1, \ldots, n,$$

$$j = 1, \ldots, c,$$

$$d_{lj} = \| x_l - y_j \|^2,$$  \hspace{1cm} (4.4)

optimized with respect to centroids $y_j$ and memberships $u_{lj}$, with some constraints placed on the total mass of membership to clusters

$$\xi_l \equiv \sum_{j=1}^{c} u_{lj}.$$  \hspace{1cm} (4.5)

In equations (4.3), and (4.5), $n$ is the cardinality of the dataset, $c$ is the number of clusters, while
The first model we present is the Maximum Entropy (ME) or Deterministic Annealing approach [98] that imposes $\zeta_l = 1$. In this case, we are in the probabilistic case, where memberships are formally equivalent to probabilities.

In addition to the expectation of the distortion equation (4.3), the objective function $J_{ME}$ of ME includes the probabilistic constraint. The necessary conditions for the minimum of $J_{ME}$ are $\nabla J_{ME} = 0$ that yields:

$$u_{lj} = \frac{e^{-d_{lj}/\beta}}{\zeta_l}, \quad (4.6)$$

and

$$y_j = \frac{\sum_{l=1}^n u_{lj}x_l}{\sum_{l=1}^n u_{lj}}, \quad (4.7)$$

equation (4.6) and equation (4.7) can be interpreted as the basis of a Picard iteration that leads to the minimum of a free energy at different levels of temperature (or fuzziness) that is regulated by the value of $\beta$ (Deterministic Annealing procedure). When $\beta$ is large the free energy is equivalent to the unconstrained optimization of the expectation of the distortion equation (4.3).

The objective function of ME is formally equivalent to the one of Fuzzy C-Means [15], and both of them show the problem of low outlier rejection: The memberships of outliers can be very large, not different from those of inliers.

In contrast to ME, the Possibilistic c-means (PCM) [72] does not impose any constraint on $\zeta_l$, so memberships are not formally equivalent to probabilities but represent degrees of typicality to clusters.

The objective of PCM, $J_{PCM}$ includes an individual parameter $\beta_j$ for each cluster, and $\nabla J_{PCM} = 0$ yields

$$u_{lj} = e^{-d_{lj}/\beta_j}, \quad (4.8)$$

for membership of instances to clusters and equation (4.7) for cluster centers. Again, equations (4.8), and (4.7) can be interpreted as the basis of a Picard iteration for the minimization of $J_{PCM}$.

As discussed in [72], while the PCM produces membership functions that can be interpreted as measures of typicality of instances to clusters and shows a strong outlier rejection, the Picard iterations may fail to converge due to the lack of competitive terms in equation (4.8).

The Graded Possibilistic c-Means (GPCM) clustering model proposed by our group [84] exploits the similarities of equation (4.6) and equation (4.8) to obtain both the nice properties of memberships with the meaning of typicality and strong outlier rejection of the PCM and the convergence ability of the ME.
In this chapter, we present a new simpler version of the GPCM. To this aim, we propose the following formula that unifies the equation (4.6) and equation (4.8):

\[ u_{lj} = \frac{v_{lj}}{Z_l}, \quad (4.9) \]

where

\[ v_{lj} = e^{-d_{lj}/\beta_j}, \quad (4.10) \]

is called the free membership and \( Z_l \) is the generalized partition function that is a function of the membership mass \( \zeta_l \).

This allows us to add a continuum of other, intermediate cases to the two limit case models just described, respectively characterized by \( Z_l = \zeta_j \) (probabilistic) and \( Z_l = 1 \) (possibilistic). Here we use the following formulation:

\[ Z_l \equiv \zeta_l^\alpha = \left( \sum_{j=1}^{c} v_{lj} \right)^\alpha, \quad \alpha \in [0,1], \quad (4.11) \]

where the parameter \( \alpha \) controls the possibility level, from a totally probabilistic (\( \alpha = 1 \)) to a totally possibilistic (\( \alpha = 0 \)) model, with all intermediate cases for \( 0 < \alpha < 1 \). The Picard iteration implementing the GPCM iterates the membership evaluation equation (4.9) and the cluster centers evaluation equation (4.7) until convergence.

In the GPCM model at each iteration of the Picard procedure \( \beta_j \) is updated [72] according to:

\[ \beta_j = \frac{\sum_{i=1}^{N} u_{ij} d_{ij}}{c \sum_{i=1}^{N} u_{ij}}, \quad j = 1,\ldots,c. \quad (4.12) \]

Note that in the GPCM after training \( \zeta_l \in (0,c) \) and depends on the value of \( \alpha \). More specifically:

- values \( \zeta_l \approx 1 \) are typical of regions well covered by centroids;
- but \( \zeta_l \gg 1 \) is very unlikely for good clustering solutions, since it implies many overlapping clusters;
- finally, \( \zeta_l \ll 1 \) characterizes regions not covered by centroids, and any observation occurring there is an outlier.

In order to reject outliers, let we define the degree of outlierness index \( \Omega \), corresponding to the concept of being an outlier, as follows:

\[ \Omega(x_j) = \max\{1 - \zeta_l, 0\}. \quad (4.13) \]
For each threshold on $\Omega$ we set, we obtain a region of inlier in the data space and we define as outliers the data outside this region.

Differently from other approaches based on analysing instance-centroid distances [120], the GPCM provides a direct measure of outlierness that is not referred to local density or to individual clusters, but is defined with respect to a whole clustering model.

Outlierness can be modulated by an appropriate choice of $\alpha$. Low values correspond to sharper outlier rejection, while higher values imply wider cluster regions and therefore lower rejection. For $\alpha = 1$ the GPCM becomes probabilistic and loses its ability to identify or reject outliers.

We can define the initial outlier density $\rho_0 \in [0, 1]$ as the average degree of outlierness:

$$\rho_0 = \frac{1}{|W_0|} \sum_{l \in W_0} \Omega(x_l),$$

(4.14)

where $W_0$ is an initial window of data to “bootstrap” GPCM and provide an initial clustering.

The average density $\rho_0$ accounts for both frequency and intensity, or degree of anomaly, of outliers. This is a mean, so quantity and intensity can compensate each other, so that the effect of few strong outliers is the same as that of many moderate outliers.

During execution, outlier intensity at step $l > |W_0|$ is computed as follows:

$$\rho_l = 0.01 \Omega(x_{l-1}) + 0.99 \rho_{l-1},$$

(4.15)

where $\Omega_l$ is the measure of outlierness at step $l$. Note that the density is a function of the past values, being a convex combination of current outlierness and past density (exponentially weighted moving average). The exponential time constant is $-\ln 100 \approx 4.6$, similar to the typical lag periods $T$ used in this study.

The updating formula can also be rewritten as

$$\rho_l = \rho_{l-1} + 0.01 \left( \Omega(x_{l-1}) - \rho_{l-1} \right),$$

(4.16)

to make it evident that it is a Robbins-Monro-type [96] formula for approximating $\Omega$, with step size of 0.01 kept fixed to enable continuous tracking, and with $\Omega(x_{l-1}) - \rho_{l-1}$ acting as the stochastic gradient estimate at step $t - 1$.

The GPCM parameters are updated during the execution as follows. To avoid premature convergence, the possibility degree $\alpha$ is made dependent on $\rho$, so as to increase centroid coverage when outliers are detected:

$$\alpha_l = \alpha_0 + \rho_l (1 - \alpha_0).$$

(4.17)

Note that $\alpha_l$ is a function of the current density and of $\alpha_0$, its baseline value, so this formula is not a moving average.
The spread parameter for each centroid, $\beta_j$, is similarly updated during the execution as follow:

$$\beta_{j,l+1} = \beta_{j,l} + \rho_l (\beta_{j,0} - \beta_{j,l}) ,$$  \hspace{1cm} (4.18)

which provides the ability to roll back closer to the initial values of $\beta$ when the model is not adequate any more, as indicated by the value of $\rho$.

### 4.2.5 Ensemble forecast model

As shown in figure 4.1, for each cluster a forecaster with the architecture shown in table 4.1 is trained and $\zeta_l$ is obtained, which is the quantity computed for each chunk in the training dataset. After the training stage, we start the forecasting stage as shown in figure 4.2 where chunks come as a stream. For each upcoming input chunk $i$, $\zeta_i$ is computed and compared to a threshold. In the proposed model the threshold is selected as the minimum of $\zeta_l$ observed on the training set:

$$\Theta \equiv \min_l \zeta_l .$$  \hspace{1cm} (4.19)

However other choices, more or less restrictive, are possible based on the quantity and reliability of the training data.

When a new chunk is presented to RLEM, if $\zeta < \Theta$ it is considered an extreme outlier and will be dropped.

Technically, this is implemented as follows. We compute the **binarized membership mass** of the input chunk, defined as:

$$\zeta^B = \begin{cases} 0 & \text{for } \zeta < \Theta \\ 1 & \text{otherwise} \end{cases} .$$  \hspace{1cm} (4.20)

The upcoming chunk is considered as an extreme outlier and is dropped (rejected) if $\zeta^B = 0$.

The drop rate of the input chunks depends on the value of $\alpha$ which controls the sensitivity of the model to outliers. A high value of $\alpha$-means less sensitivity to outliers and a lower drop rate.

For detecting concept shift in traffic flows, we use average density $\rho$ as an indicator of the reliability of the forecasting model.

The final output of the RLEM is computed as a weighted sum of the individual base learner forecasts [57], as follows:

$$f = \sum_{j=1}^{c} f_j u_j / \zeta .$$  \hspace{1cm} (4.21)

In equation (4.21) we see that the output $f_j$ of each forecaster is weighted by $u_j$, which is the membership degree of each chunk to each cluster, so that $u_j$ will have a high value for the most suitable forecaster(s) and low to the others.
Note that, despite the possibilistic nature of the GPCM method, this weighting is convex ($\sum_j u_j/\zeta = 1$) because of the use of $\zeta$ as a partition function, since outliers and concept drift/shift have been taken into account in the previous layer.

4.2.6 Retraining

During operation, the system collects a sliding window of a fixed number of past observations from the input stream. When the outlier density $\rho$ is over a certain threshold $\rho_t$, the model is considered inadequate and a retraining step is triggered.

In the retraining step, the centroids and forecasters are trained on the current data window, so as to make them up to date.

4.3 Experiments and results

The experimental validation of proposed Robust Layered Ensemble Model included the test of the clustering procedure based on the Graded Possibilistic $c$-Means on an artificial dataset with built-in concept drift and shift. Then we applied RLEM to the short-term forecasting of three traffic flow datasets.

4.3.1 Datasets

The datasets employed in our experimental analysis are:

- **Gaussian dataset** that is a synthetic dataset with four evolving two-dimensional Gaussian distributions [26]. Along time, one new data point is added and one removed randomly so that the total number stays constant. However, the underlying data source (centroid positions) is slowly changed, leading to concept drift. Concept shift is obtained by removing a whole segment of the sequence at time 4,000 where the stream changes abruptly. The dataset was generated using the Matlab program ConceptDriftData.m available at https://github.com/gditzler/ConceptDriftData.

- **PeMS** that is a dataset by the Caltrans Performance Measurement System (PeMS) available at http://pems.dot.ca.gov. The traffic flow data are collected every 30 seconds from over 15,000 detectors deployed across California. The collected data are aggregated in 5-minutes periods. In [80] a deep learning model was developed using these data.

- **UK road network** that contains multiple datasets obtained from different road links in the United Kingdom (UK) available at http://www.highways.gov.uk. This data
Figure 4.3: The five snapshots taken during the clustering process of the Gaussian dataset (see figure 4.4). In each snapshot red stars are centroids. Dots are the 100 previous data points, with the 30 most recent in darker colour.
Figure 4.4: Degree of outlierness during the tracking of the Gaussian dataset. The numbers under the curve correspond to the five snapshots in figure 4.3.
Figure 4.5: The road network for the short-term traffic forecasting study and the corresponding graph.
4.3.2 Implemented models

The learning task associated with the Gaussian dataset is non-stationary data streaming tracking and outlier detecting. We approach this problem using the GPCM clustering model.

Table 4.1 shows the values of parameters of the RLEM implementations for the short-term traffic forecasting for PeMS, UK, and Genoa datasets. Each data corresponds to the average traffic flow measured in the observation period.

The size of the data chunk is the time lag estimated as the minimum of the time-delayed mutual information, as noted in 4.2.2. The estimated time lags for PeMS, UK, and Genoa datasets correspond, respectively, to 35 minutes, one day, and 20 minutes.

---


2 Appendix B of this thesis gives additional details on the SUMO package, the Genoa road network implementation, and the simulated data extraction.
For the first stage of RLEM, that implements a GPCM model for chunk clustering, we set five clusters for all datasets.

The base learners of the second layer of the RLEM are Time-Delayed Neural Networks (TDNN) using Multi-Layer Perceptrons with three layers. For the PeMS, and UK datasets, the dimension of the input layer of Multilayer Perceptrons is identical to the size of the chunk, while in the Genoa dataset the dimension of the input layer of Multilayer Perceptrons is equal to the (size of the chunk * the number of selected adjacent links), where the number of adjacent links are equal to 2. The hidden layers are set to 10 units for the three cases, while the output is unidimensional and corresponds to the estimation of the traffic flow.

4.3.3 Choice of parameters

As most adaptive methods, the RLEM model includes three types of parameters: Model parameters, optimization parameters, and evaluation parameters.

Model parameters directly influence the operation of the system in the inference (forecasting) phase. Although the model just described includes several parameters, the only actual, user-selected model parameters are the number of forecasters $c$ and the topology of the individual forecasters. When the number of forecasters is increased, it has been observed that the performance of the system increases accordingly, although not proportionally. Additional model parameters influencing the trade-off between stability and reactivity of the system are the adaptation gain for the moving-average update of $\rho$ and the lag period $T$. For both the user can choose an arbitrary value, but reasonable, objective selection criteria have been previously discussed (equation (4.16) and (4.2)). The membership threshold $\Theta$ should operate on extreme observations. Even if criteria other than equation (4.19) are employed to set its value, it should not impact normal operation.

Most parameters described are optimization parameters. These have an indirect influence on the system’s behavior, being related to the evolution of the system in time. These include initial values for $\alpha$ and $\beta$, the size of the initial window $W_0$, and the optimization parameters for the individual forecasters which depend on the training strategy adopted (in this study we used the error back-propagation algorithm) but do not have a strong influence on the result due to the use of an ensemble. As for the actual numerical values of these parameters, $\alpha$ has an absolute interpretation and values in $[0.85, 1)$ can be used. However, $\beta$ strongly depends on the magnitude, distribution, and dimensionality of the data and on the location of clusters, so a general indication cannot be given, although empirical methods like analyzing the histogram of pairwise distances between samples can be attempted.

Finally, evaluation parameters include the metrics employed to measure performance and the relative size of training set and test set. These don’t have a strong influence on the results provided that the metrics are reasonably related to actual performance on the field, that they are
used consistently in comparisons, and that the absolute size of training and test set are sufficient. Repeated experiments have shown that this latter point was not an issue with the datasets used in this study.

4.3.4 Experimental results and discussion

Gaussian dataset

Figure 4.4 shows the outlierness index $\rho$ equation (4.14) during the tracking of the Gaussian dataset. Five snapshots, taken at different times, are shown in figure 4.3 and labeled with numbers corresponding to those in figure 4.4. Dots represent the 100 most recent data points of the evolving dataset. Stars are the current centroids.

The outlierness index is high when the clustering model does not fit well the data, indicating an inadequacy situation. Observing the snapshots in figure 4.3 and referring to the outlierness indicator in figure 4.4 shows that the model can quickly adapt to the novelty:

1. After recovering from a moderate drift with respect to initial configuration;
2. After some outliers have appeared (note the fast recovery of the outlierness indicator in figure 4.4);
3. Clusters are changing their relative position but the data support stays approximately the same. Outlierness slightly increased;
4. Concept shift. The data support changes abruptly from the South-West to the North-East part of the graph. Outlierness peaks;
5. Recovery from concept shift. Incoming data points are no longer considered as outliers.
Figure 4.6: PeMS dataset: Forecasting results of RLEM (measured values are in blue; forecasted values are in red).
Table 4.2: Performance comparison on PeMS dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Index</th>
<th>RMSE</th>
<th>Drop rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAE</td>
<td></td>
<td>50.0</td>
<td>0</td>
</tr>
<tr>
<td>BP-NN</td>
<td></td>
<td>90.2</td>
<td>0</td>
</tr>
<tr>
<td>RBF-NN</td>
<td></td>
<td>56.1</td>
<td>0</td>
</tr>
<tr>
<td>RLEM</td>
<td></td>
<td>20.8</td>
<td>0.0044</td>
</tr>
</tbody>
</table>

PeMS

In figure 4.6 a forecasting experiment on the traffic flow data that were used in [80] for comparing the forecasting capabilities of the Stacked AutoEncoder (SAE), the Back-Propagation Neural Network (BP-NN), and the Radial Basis Function Neural Network (RBF-NN) using 3 days data for training and the upcoming 7 days data for testing. The figure shows the forecasting results obtained by the RLEM using the same training and test sets.

In table 4.2 we compare forecasting performances of the models studied in [80] with the RLEM. The performance indexes used in the table are:

- The Mean Squared Error (MSE) measuring the average error of the forecasting results:

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (t_i - \hat{t}_i)^2}, \quad i = 1, \ldots, N, \tag{4.22}
\]

where \(t_i\) is the observed traffic value, \(\hat{t}_i\) is its forecasted value and \(N\) is the size of the test set.

- The Drop Rate (DR) defined as follow:

\[
DR = 1 - \frac{\sum_{i=1}^{N} \chi_B}{N}. \tag{4.23}
\]

With a drop of 9 outliers corresponding to a DR=0.0044, the RLEM shows the best Root Mean Squared Error.

**UK and Genoa datasets**

Figure 4.9 shows data from the UK dataset as a continuous line, with forecast output superimposed as round dots, with a similar representation as in figure 4.6.

Figure 4.7 shows the effect of \(\alpha\) on the accuracy (Mean square error) of the RLEM model for the UK road network dataset. The selected range of \(\alpha\) values are \(0.93 \leq \alpha \leq 1\). An appropriate value of \(\alpha\) allows us to control the degree of outlierness, drop unwanted outliers, and improve the
accuracy rate. The values of the RMS are very small, but this magnitude depends on the range of the data. What carries useful information is actually the change in these values, i.e., the relative differences between values.

Figure 4.8 shows the scatter plots of the traffic flow forecasting using the UK road network dataset (a), and the one for Genoa dataset (b), both with zero drop rate. The correlation coefficients are, respectively, 0.98 and 0.99.

Figure 4.10 shows the binarized mass of membership $\zeta^B_i$ for the chunks of the test set. Where the value of $\zeta^B_i$ drops the forecasting performance decreases, because the data are not well explained by the model. This makes $\zeta^B_i$ a good indicator of model reliability and forecasting performance even during the inference phase, i.e., when targets are not available.
Figure 4.8: Results of the two forecasting problems. (a) Scatter plot between the target and the output (UK dataset). (b) Forecast output and the target (Genoa dataset). The regression curves are in blue.
Figure 4.9: Forecast output and the target on the UK dataset with 0 drop rate.

Figure 4.10: Binarized sum of membership of each chunk to all clusters during a run on the UK dataset.
Chapter 5

Conclusion and future work

clustering is one of the most relevant data mining tasks. Its goal is to group similar objects in one cluster while dissimilar objects should belong to different clusters. Different clustering approaches have been developed and widely applied in data mining applications. Clustering data streams is a big challenge as their behavior may change overtime (non-stationarity). We distinguish between two types of change: for evolutionary, smooth changes we use the term concept drift, while a radical, sudden change is labeled concept shift. Outliers are isolated observations which are clearly not explained by the clustering model, while the majority of observations are well fitted by the current clustering model. The clustering model should be insensitive to these episodic anomalies.

In this thesis, we focused our study on possibilistic clustering algorithms as means to perform clustering of non-stationary data streams. We specifically exploited the ability, provided by an appropriate possibilistic clustering method, to cluster iteratively using both batch (sliding-window) and online (by-pattern) strategies that can detect outliers and track and adapt to concept drift and shift in a natural way. Measures of fuzzy outlierness and fuzzy outlier density are obtained as immediate by-products of the possibilistic clustering technique adopted. These measures are used to modulate the amount of incremental learning according to the different learning regimes required by non-stationary data stream clustering. Compared with other fuzzy clustering approaches, we have shown that our possibilistic model possesses intrinsic robustness properties which make it a very effective basis for tracking model changes while being insensitive to outliers. The proposed method is currently being deployed in several applications, ranging from urban traffic forecasting to Intrusion detection. In the thesis, we have also proposed a new robust layered ensemble model for traffic forecasting. The proposed Robust Layered Ensemble Model (RLEM) consists of two layers. In the first layer, the chunks (time-lag vectors from the time series) are clustered into homogeneous regions. To perform this segmentation of the input space we employ our clustering model (GPCM) that is able to adapt to the changes in the traffic.
flow, by implementing a continuous learning that exploits the input chunks as they arrive. Intrinsic to this clustering method is a measure of outlierness that provides information about the goodness of fit of each input chunk to the clustering model. In the second layer, an ensemble of a number of base learners acting as forecasters equal to the number of clusters is used, each of them specialized on a homogeneous region of the data space. We evaluated the performance of our RLEM on several real traffic datasets. The results show that the proposed model gives an accurate forecasting of the traffic flow rates with outlier detection and shows a good adaptation to non-stationary traffic regimes. Given its characteristics of outlier detection, accuracy, and robustness, RLEM can be fruitfully integrated into real-time traffic flow management systems.

We have evaluated the performance of the proposed clustering model on synthetic dataset, for which the ground truth is available, and several real datasets. Then we evaluated the performance of RLEM model on 3 datasets. For the PeMS dataset we compared our results with SAE, BP NN, and RBF NN model, and the results show that the proposed method gives an accurate forecasting of the traffic flow rates with outlier detection and shows a good adaptation to non-stationary traffic regimes. For the UK dataset, we show that the proper selection of $\alpha$ improves the forecasting accuracy.

Future work will include improvements in automatic setting of some model parameters, as well as in the optimization process.
Appendix A

GPCM-stream Implementation

A MATLAB implementation of the online GPCM-stream is available at: https://it.mathworks.com/matlabcentral/fileexchange/64318-onlinegradedpossibilisticclustering (see QR-code of figure A.1). A Python version is under implementation.

Figure A.1: QR-code linking the web site location of the online GPCM-stream implementation

The online GPCM-stream algorithm is able to detect outliers and adapt to concept shift. The implemented version can work on raw data and on normalized data. In the first case one will have to change the axis limits of the dynamic plot axis to visualize the clusters centroids. The following section includes the MATLAB codes.
% % Copyright (c) 2017, DIBRIS-University of Genoa (www.dibris.unige.it) % All rights reserved. Please read the "license.txt" for license terms . % Project Code: AmrDIBRIS101 % Project Title: Implementation of Online Graded Possibilistic Clustering "OGPC" Clustering in MATLAB % Read more about the online algorithm % https://www.springerprofessional.de/graded-possibilistic-clustering-of-non-stationary-data-streams/12047532 % Note: This a modified version of the online algorithm % (Algorithm able to detect outliers and adapt to concept shift). % Publisher: DIBRIS (www.dibris.unige.it) % Developer: Amr Abdullatif (DIBRIS-University of Genoa)

function [rhovals, summembership, U, Youtn, Y, normvals, bend, Uval, Yval] = ogpc(X, Y, fb, K, maxiter, eta0, alphamin, plt)

% FUNCTION INPUTS

% X: Input data (Normalized)
% Y: Initial centroids (Please read cm2.m), default: Y = 0 (computed
% from the code)
% fb: First batch size (Used to get initial centroids)
% K: Number of clusters
% eta0: Default initial value is 1 (Update centroids)
% alphamin: Controls the "possibility level", from a totally probabilistic (alphamin = 1) to a totally possibilistic (alphamin = 0).
% Outliers can be modulated by an appropriate choice of alphamin.
% alphamin default value is 0.99 .
% plt: 2D list used to plot data

% Function outputs:

% rhovals: List contains degree of outlierness of incoming data streams.
% summembership : Sum of membership of each pattern to all clusters.
% U: Membership of a pattern to all clusters.
% Youtn, Y: Centroids (Normalized and not normalized).
% normvals: Values used for normalization.
% bend: Final clusters width.
% Uval: Memberships values of all samples to all clusters

absc = plt(1); % For plotting data
ord = plt(2); % For plotting data
a0 = 0.1; % Default 0.2 (set low) Control the learning rate in region 1 "concept drift".
t1 = .01; % Default .1 (set low) Control the learning rate in region 3 "concept shift"
k = 50; % Default 50 (set high) Control the no learning interval in region 2 "outliers"
g = .7; % Default .5(set high)
et = etal; % Default 1
firstbatch = fb;
maxitr = maxiter; % Maximum number of iterations
nout = 50; % Number of iterations for fprintf
close;
normvals=[zeros(1,size(X,2)); ones(1,size(X,2))];
Uval={};
Yval={};

% Uncomment the following line to check parameters
% plot(0:.01:1, 1+a0*exp(-((0:.01:1)./t1))-exp(-(((0:.01:1)./(k*t1)).^((g*log(1/t1))))));pause

%% TRAINING
% From first batch we obtain initial centroids, memberships, and sum of
% memberships (used to obtain the outlierness degree in the first batch)
tinterval=1:firstbatch;
[Ui, Y, Youtn, bend] = cm2(X(tinterval,:),K,maxitr,1,0,Y,eta,alphamin);
summembership = sum(Ui,2);
summembership=summembership(:)';
omega = 1-summembership;
omega(omega<0)=0;
rho=mean(omega(:));
bi0=bend;

% Plot initial centroids and data
plot(Y(:,absc),Y(:,ord),’r*’,X(tinterval,absc),X(tinterval,ord),’b .’)
axis([0 1 0 1]);
grid;
drawnow;
count=0;
U = zeros(1,size(Y,1));
rhovals = [];
numiter = size(X,1)-firstbatch;

% ONLINE TEST
% Omega: Outlierness degree
for j=firstbatch+1:size(X,1)
    count = count + 1;
    omega = 1-summembership;
    omega(omega<0)=0;
    rho = .01*mean(omega)+.99*rho; % Degree of outlierness
    rhovals = [rhovals rho]; % List of outlierness degree
    theta=1+a0*exp(-(rho./t1)) -exp(-((rho./(k*t1)).ˆ(g*log(1/t1))))
    eta = theta*eta0;
    alpha = alphamin+rho*(1-alphamin);
    U = membership2(X(j,:),Y,bend,alpha); % description in
    membership2.m
    Y = Y+((eta*(repmat(X(j,:),size(Y,1),1)-Y).*repmat(U,size(X,2)
    ,1)’) /sum(U)); % added /sum(U)
    %dist=dist2(X(j,:),Y);
    bend = betafcm(X(j,:),Y,U)/K;
    %bend= sum(dist(1:size(U,2)).*U,2)./sum(U,2) ;
    bend=bend+rho*(bi0-bend);
    bi0 = bend;
    U = removedoubles(U,Y,bend);
    summembership = sum(U);
    Uval{j} = U;
    Yval{j} = Y;

    % Dynamic plot of centroids and data
    if ~mod(count,nout)
fprintf('iter %i of %i - rho = %f , theta = %f , eta = %f , alpha = %f
', count, numiter, rho, theta, eta, alpha);

end

tinterval=max(1,j-30):j;

tintervalpast=max(1,j-100):j;

plot(X(tintervalpast,absc),X(tintervalpast,ord),'c.',X(tinterval,absc),X(tinterval,ord),'b.',Y(:,absc),Y(:,ord),'r*');

axis([0 1 0 1]);
grid;
drawnow;
end

end

%%%% plot the final outlirness degree of all patterns
% Density of outliers are used as indicator to outliers and concept shift
% in data streams.
plot(rhovals);

%%%% removedoubles function

function U= removedoubles(U,Y,bend)

dd=dist2(Y);
remove=zeros(size(dd));
for i=1:(size(dd,2)-1)
    for j=i+1:size(dd,2)
        if dd(i,j)<bend
            remove(i,j) = true;
        end
    end
end
keep=(prod(~remove)>0);
U = U(:,keep);
end

%Examples:
% Example 1:
% Gaussian dataset (Download from the tool weblink): Synthetic data sets containing concept drift, so
% outliers and concept shift was added by removing a number of
observations in two parts of the data sequence.

% load('gaussshift.mat')
% Normalized input is included in 'gaussshift.mat'.
% Then you can run the following code:
% [rhovals, summembership, U, Youtn, Y, Normvals, bend] = ogpc(
  gaussdata_shift_norm, 0, 1000, 4, 100, 1, .99, [1, 2])
%
function [Uout, Yout, Youtn, b] = cm2(X, k, maxiter, replicates, epsilon, Y0, eta, a)
  % clustering skeleton

  % checking mandatory arguments
  if nargin < 2
    disp('At least X and k must be provided')
    Y = [];
    U = [];
    return
  end

  % checking and setting optional arguments
  if exist('maxiter')==0
    maxiter = 10;
  end
  if exist('epsilon')==0
    epsilon = 0;
  end
  if exist('replicates')==0 || replicates<1
    replicates = 1;
  end
  if exist('Y0')
    replicates = 1;
  end

  % initializing run
  Jopt = sum(min(dist2(min(X),X)))/size(X,1);
  for cnt = 1:replicates
    % initializing replicate
    if Y0==0
      [Y, Ufcm] = fcm(X, k);
      b = betafcm(X, Y, Ufcm);  % beta for every cluster.
      b = b/k;
    end
  end
else
    Y = Y0;
    b = betafcm(X,Y,Ufcm);
    b=b/k;
end
for h = 1:maxiter
    % computing memberships
    U = membership2(X,Y,b,a);
    % computing centroids
    YN = U'*X./repmat(sum(U)',1,size(X,2));
    % checking mean squared variation
    if sum(sum((Y-YN).^2,2)) <= epsilon
        break
    else
        Y = eta*YN+(1-eta)*Y;
    end
end
% cost at the end of iteration
J = sum(min(dist2(Y,X)))/size(X,1);
if exist('Uout') == 0 | J<Jopt
    Uout = U;
    Yout = Y;
    % Ul=Ulog;
end
end
% remapping centroids into original data space (de-normalization)
Youtn=Y;
% Yout = Yout. *repmat(Normvals(2,:)-Normvals(1,:),k,1)+repmat(
%    Normvals(1,:),k,1);
end

function bfcm = betafcm(x,y,uf)
% Compute the clusters width
if nargin==0
    d=[];
    bfcm=[];
    return;
end
if nargin ==1
function [U,Z] = membership2(X,Y,b,a)
% Compute the possibilistic memebships
U = dist2(X,Y);
if(length(b)==1)
    U = exp(-U./b);
elseif size(b(:),1)== size(U,2)
    b = b(:)';
    U = exp(-U./repmat(b,size(U,1),1));
else
    disp('b should either be a scalar or contain one value per cluster')
    U = [];
    return
end
U = U + 10^-100;
Z = repmat(sum(U,2),1,size(U,2));
if exist('a','var')
    Z=Z.^a;
end
U = U./Z;
end

function d = dist2(x,y)
if nargin==0
    d=[];
    return;
end
if nargin ==1
    y=x;
end
d=repmat(sum(x'.^2)',1,size(y,1));
d=d+repmat(sum(y'.^2), size(x,1),1);
d=abs(d-2*x*y');  \%x^2+y^2-2xy
bfcm= sum(d.*uf',1)./sum(uf',1) ;
end
Appendix B

Urban traffic data simulator

The Genoa Dataset containing traffic data of Genoa town, described in 4, was obtained via simulation as follows as a part of the contribution to the PLUG-IN project. This Appendix illustrates the SUMO package, the Genoa road network implementation, and the simulated data extraction process.

B.1 SUMO Traffic Simulator

The Simulation of Urban MObility (SUMO) (http://SUMO.dlr.de/wiki/Simulation_of_Urban_MObility__Wiki) is an open-source microscopic traffic simulator created by the German Aerospace Center (DLR) in 2001, with the goal to provide a tool that would be fast in run-time and portable, as well able to support new algorithms developed by its users. It is currently in the version 0.31 for Windows and it has been widely used among traffic simulation community. An important feature is the fact that SUMO is a multi-modal simulator, being able to not only depict the cars’ movements but also the other kind of vehicles that usually circulate within a city, like motorcycles, bicycles or pedestrians, as well public transport vehicles: bus, train, tram or taxi. The simulator is available through command-line or through a GUI (Graphical User Interface) for those who need a graphical presentation. The microscopic algorithm used is the car-following model extension of Gipps algorithm [68], which is a collision-free model since a vehicle adapts its velocity at each simulation step according to the speed of the leading vehicle (the vehicle that is ahead). The vehicle’s speed is defined using two auxiliary equations. The first one is the safe velocity, which is the maximum speed the vehicle can go in order to not collide with the vehicle ahead:

\[ v_s = \frac{v_l}{\gamma} \]

where \( v_s \) is the safe velocity, \( v_l \) is the speed of the leading vehicle, and \( \gamma \) is a parameter that can be adjusted to change the safety distance. The second equation defines the acceleration of the vehicle:

\[ a = \frac{v_s - v}{\delta} \]

where \( a \) is the acceleration, \( v \) is the current velocity of the vehicle, and \( \delta \) is another parameter that controls the deceleration rate. This algorithm ensures that vehicles maintain a safe distance from the one in front of them and adjust their speed to match the speed of the leading vehicle, thus preventing collisions.

Figure B.1: SUMO Traffic Simulator

\[ v_{safe}(t) = v_{l}(t) + \frac{g(t) - v_{l}(t)\tau}{b(v)} \]  \hspace{1cm} (B.1)

\( v_{l}(t) \): speed of the leading vehicle in time \( t \);
\( g(t) \): gap to the leading vehicle in time \( t \);
\( \tau \): the driver’s reaction time (usually 1s);
\( b \): the deceleration function.

Then there is the desired velocity, which is the speed the driver wants to have:

\[ v_{dest}(t) = \min[v_{safe}, v(t) + a, v_{max}] \]  \hspace{1cm} (B.2)

The final velocity is calculated through a third equation that binds a random factor, in order to simulate the fact that the driver might not be able to achieve the desired speed.

\[ v(t) = \max[0, \text{rand}[v_{dest}(t) - a, v_{dest}(t)]] \]  \hspace{1cm} (B.3)

The format of the files used by SUMO is XML-based, varying on the extension used according to the type of file. For instance, the configuration file for the simulation is saved with a .SUMO.cfg extension; the network is defined into a file using .net.xml extension; or even the routes file which uses the .rou.xml extension. Besides the simulator, SUMO has also a set of small applications.
included in its package that work as auxiliary tools for tasks such as: generating the network to be used (netconvert, netgenerate), defining and computing the routes of vehicles (darouter, jtrrouter, dfrouter), inputting the demand of vehicles in the simulation (od2trips), as well for other uses (eg: polyconvert, edgesinDistrict).

The network to be used in simulation can be generate randomly with netgenerate or by using an existing map either previously defined "by hand" or from a maps service like OpenStreetMaps. In Our project we build Marassi Network area from scratch.

To define the traffic demand there are three different generators: the first, duarouter (the one used in the project), is the simplest since it uses the shortest path computation as the way to generate the routes, being the Dijkstra model the algorithm used. There are also many options to control traffic for example, the application jtrouter uses the turning percentages at junctions to compute the routes, being suitable to use in a urban road network scenario. There is also the dfrouter which computes the routes by using actual data collected from induction loop detectors and which is inserted into the network through a few "source points". Those "source points" represent the actual existing detectors. We can also closed roads for specified period, create accident, and create signalized network.

B.2 Network Implementation

The network used as base for implementing both simulation and data-driven models was the Foce-Marassi network area in Genova city, Italy. The network contain 6 main origin/destination
nodes described from/to [BISAGNO, GENOVA EST, MARASSI, CENTER, PONENTE, LEVANTE]. There are 104 loop detectors that cover all road links in the network and it measures traffic parameters (Speed, Average Traffic flow, Halting period, Jam period, Maximum number if vehicle, ..., etc). The network contain 104 road links and they are all covered by loop detectors that measures traffic flow from all the links in the network.

### B.3 Simulation and data extraction

When the road network and traffic descriptions are ready the simulation can be run. For the simulation to be useful the user needs some kind of output. In SUMO we define all the network in XML files (Nodes, Edges, Flow, Routing, Connection, ..., etc). To run a large scale simulations it would be necessary to distribute the simulation across several computational nodes. However SUMO can’t run in a distributed environment on its own. Extracting loop detector measurements can be done with the Traffic Control Interface (TraCI). TraCI uses a TCP based client/server architecture to access SUMO. In this case SUMO acts as a server and e.g a Python script acts as a client. The client can then use TraCI to ask for or change the state of the running simulation. There are many attributes that can be queried and changed but for the current purpose, being able to extract detectors measurements (Average traffic flow every 5 min) and from all the links are most important as shown in figure B.4.

![Figure B.4: SUMO generated data from detectors](image)

We can also create accidents or close road link and define route probability to vehicles to pass a specific road in the network as shown in figure B.5.
Figure B.5: Close a particular link for 1 hour
References


[37] J. H. Friedman and L. C. Rafsky. Multivariate generalizations of the wald-wolfowitz and

[38] J. Gao, Z. Leng, Y. Qin, Z. Ma, and X. Liu. Short-term traffic flow forecasting model based
on wavelet neural network. In 25th *Chinese Control and Decision Conference (CCDC)*,


[41] B. Ghosh, B. Basu, and M. O’Mahony. Multivariate short-term traffic flow forecast-
ing using time-series analysis. *IEEE Transactions on Intelligent Transportation Systems*,

[42] D. Giglio. A medium-scale network model for short-term traffic prediction at neighbour-
hood level. In 2015 *IEEE 18th International Conference on Intelligent Transportation

streams: Theory and practice. *IEEE Trans. on Knowl. and Data Eng.*, 15(3):515–528,


[45] P. Hall and N. Tajvidi. Permutation tests for equality of distributions in high-dimensional


1994.


106


