https://doi.org/10.1142/S0218213019600091

Data stream classification by dynamic incremental semi-supervised fuzzy clustering*

Gabriella Casalino, Giovanna Castellano, Corrado Mencar

CILab - Computational Intelligence Lab
Department of Computer Science
University of Bari Aldo Moro, Italy

Abstract

A data stream classification method called DISSFCM (Dynamic Incremental Semi-Supervised FCM) is presented, which is based on an incremental semi-supervised fuzzy clustering algorithm. The method assumes that partially labeled data belonging to different classes are continuously available during time in form of chunks. Each chunk is processed by semi-supervised fuzzy clustering leading to a cluster-based classification model. The proposed DISSFCM is capable of dynamically adapting the number of clusters to data streams, by splitting low-quality clusters so as to improve classification quality. Experimental results on both synthetic and real-world data show the effectiveness of the proposed method in data stream classification.

Data stream classification; Semi-supervised fuzzy clustering; Incremental adaptive learning

1 Introduction

Data stream mining is a recent methodology that deals with the analysis of large volumes of ordered sequences of data samples. Data streams are a manifestation of Big Data, which are characterized by the four 'V' dimensions, namely Volume, Velocity, Variety and Veracity.[1] They are produced by sensor networks[2, 3], e-mails[4], online transactions[5], network traffic[6, 7], weather forecasting[8], health monitoring[9], social networks[10, 11] , learning analytics[12], etc., just to cite the the most common applications made available by current technology.

In data stream mining, it is commonly assumed that the volume of the sequence of data is so large that samples can be used a few times (or just once) for the analysis. This requirement involves the development of special-purpose data analysis methods, which should not require to store the whole stream of data in memory.[13] An approach to analyze data streams exploits an incremental generation of informational patterns, which represent a synthesized

*The paper is a revised and extended version of: G. Casalino, G. Castellano and C. Mencar, "Incremental adaptive semi-supervised fuzzy clustering for data stream classification," 2018 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), Rhodes, 2018

view of all data records analyzed in past and progressively evolve as new data records are available. Incremental and on-line algorithms are potentially useful to deal with continuous arrival of data in rapid, time-varying, and potentially unbounded streams since they continuously incorporate information into their model.[14, 15]

Main challenges in data stream mining derive from dealing with non-stationary and potentially unbounded data, which require the development of algorithms capable of performing fast and incremental processing of data samples, with limited time and memory constraints.[16] Data stream mining is applied for different tasks, such as classification, clustering, frequent pattern mining and time-series analysis.[17] In this paper, we focus on classification of data samples in a stream[18, 19] which poses specific challenges in the data stream domain, such as concept drift and class imbalance.[20, 21]

In particular, we specialize in semi-supervised methods as we do not assume that all data samples are completely labeled; in fact, labeled samples may be difficult or expensive to obtain in some real-world scenarios, meanwhile unlabeled data are relatively easy to collect. For example it is quite easy to collect new sensor data coming from continuous streams but it may be difficult or even impossible to manually label all such data. Recently, we developed an incremental semi-supervised clustering method for data stream classification,[22] which applies the Semi-Supervised Fuzzy C-Means algorithm (SSFCM)[23] to data chunks. Successively, the method has been refined by enabling the dynamic determination of the number of clusters through an appropriate splitting procedure, leading to the DISSFCM (Dynamic Incremental Semi-Supervised FCM) algorithm.[24]

In this paper we extend the previous work to better analyze the dynamic nature of DISSFCM and its capability to cope with changes in the distribution of data incoming in the stream.

The organization of the rest of the paper is as follows. Section 2 presents the state of the art related to semi-supervised fuzzy stream data analysis. Section 3 presents our method for data stream classification and its extension based on cluster splitting. In Section 4 the effectiveness of DISSFCM is evaluated on synthetic and real-world datasets in order to show its ability to cope with changes in data distribution. The last section draws some concluding remarks and outlines future work.

2 Related Works

A common approach for data stream classification is based on sliding windows, where a subset of data (data chunk) is considered for building and evolving a classifier.[25] The data chunk contains the most recent data that are available from a stream; when new data arrive, chunk change either by replacing oldest data with newest, or replacing all data as soon as there are enough new data to fill the whole chunk. The sliding windows approach is very general as it enables the application of most classifier design methods, which however are limited to use only a part of the stream.[17]

Different approaches can be adopted to design classification models from data streams organized in chunks; among others, ensemble methods[26, 27] and evolving schemes[28] are often used, sometimes in combination.[29] In particu-

lar, evolving schemes for data-driven classifiers can be used to train classifiers on data chunks while taking into account the knowledge acquired in previous chunks. As a consequence, a single model is dynamically adapted to data (in contrast to ensemble methods, which build a multitude of models), thus fostering interpretability and intelligent data analysis.[30]

Many classical machine learning methods have been adapted to evolving schemes. Yang *et al.* adopted naïve Bayesian classifers due to their low construction cost and easiness for incremental maintenance.[31] Ikonomovska *et al.* propose an incremental method to learn regression and model trees from stream data.[32] Zhang & Zhou propose a novel parameter estimation method, called transfer estimation, to dynamically adapt the estimation of the class priors and adjust a classifier accordingly.[33] The method has been applied to the Online Expectation-Minimization algorithm for non-stationary data sequence classification.[34] Pang *et al.* developed an incremental linear discriminant analysis that is able to evolve a discriminant eigenspace over fast and large data streams.[35] Guarracino *et al.* propose Incremental Regularized Generalized Eigenvalue Classification (I-ReGEC), which is an evolving scheme of a supervised learning algorithm for dynamically tuning kernel functions used for classification.[36] Annapoorna *et al.* implemented Random Forests with stratified random sampling and Bloom filtering in order to fast training in the presence of high-velocity data.[37] Gomes *et al.* enhanced Random Forests for the context of data stream mining with resampling and adaptive operators that can cope with different types of concept drifts without complex optimization.[38]

In classification tasks, a common approach is to cluster data so as to group data belonging to the same class.[39, 40] Aggarwal & Yu develop a method for clustering uncertain data streams with the use of a micro-clustering model.[41] Hyde *et al.* also used micro-clusters in a two-stage process, the first sfor producing micro-clusters and the second stage for combining them into macro-clusters.[19] Ghesmoune *et al.* developed a version of the growing neural gas approach, aimed to clustering data streams by making one pass over the data, which allows to discover clusters of arbitrary shapes without any assumptions on the number of clusters.[42] In these cases, however, class information is not directly exploited in the clustering process. Lughofer proposed a clustering-based classification technique that is able to perform supervised clustering in a data stream context, by extending the conventional learning vector quantization scheme so as to evolve new clusters on demand when data of a stream become available.[43]

Fuzzy clustering offers several advantages in the case of data stream analysis, such as a better representation of concept drift through continuous membership functions and robustness of possibilistic models among others.[44] This gave rise to several extensions of fuzzy clustering techniques to data streams, mainly by improving efficiency[45, 46] and effectiveness for highly-dimensional data.[47] On more general terms, the use of fuzzy logic in data stream mining and big data analytics is interesting for the robustness and interpretability of the resulting models, at the expenses of a more complex design process.[48]

Different from most offline classification tasks, in data stream classification it is not realistic to assume that all data are pre-classified. As a consequence, many studies focus on classifying stream data that are partially or completely unlabeled. Bassan & Santos propose a technique for classifying unlabeled streaming data using grammar-based immune programming, where data are labelled

through an active learning technique.[49] Active learning if often used in data stream mining[50], but it requires an external oracle (a human annotator or particular sample selection criteria[18]) to label critical data so as to refine the classification model. As an alternative approach to active learning, semi-supervised clustering exploits data neighborhood to classify unlabeled data. Available class labels provide information to the clustering process, so as to foster grouping of data belonging to the same class while separating data belonging to different classes. Semi-supervised clustering is a mainstream methodology when class labels are not available for all data samples.[51, 52] and its application to data stream mining is currently in progress. Borchani *et al.* proposed a semi-supervised approach for handling concept-drifting data streams containing both labeled and unlabeled instances, which can be applied to different classification models.[53] Castellano *et al.* proposed an incremental method for fuzzy clustering process guided by partial supervision which has been applied to shape annotation in image streams.[54]

In many cases, the number of clusters is either user-defined or determined from data, but it is usually kept fixed throughout the clustering process. The constant number of clusters turns out to be a limitation in data stream analysis, because concept drifts lead to a change in the structure of data which could be better described by adapting the number of clusters to incoming chunks. Silva *et al.* describe a support system for both estimating the number of clusters from data and monitoring the process of the data-stream clustering.[55] Lughofer proposed split and merge operations for evolving cluster models, which are learned incrementally from data streams.[56] Cao *et al.* proposed a new approach for discovering clusters in an evolving data stream, which are dynamically adapted in number through a pruning strategy.[57] In general, the design of clustering algorithms that adapt the cluster number to the structure of data in a stream is one of the main challenges in data stream clustering. Few works focus on the dynamical adaption of the number of fuzzy clusters.[58] In this paper we propose a novel approach for the dynamical adaption of the number of fuzzy clusters in a semi-supervised setting for data stream analysis.

3 Dynamic Incremental Semi-Supervised FCM

We assume the availability of a data stream X consisting of samples $\mathbf{x}$ characterized by numerical features, i.e. $\mathbf{x} \in \mathbb{R}^n$. Each sample belongs to a class in $\mathcal{C} = \{1, \ldots, C\}$ through a classification function $f : X \mapsto \mathcal{C}$. However, according to the semi-supervised hypothesis, the classification function is generally unknown but is some samples only; we formalize this hypothesis through a function $b : X \mapsto \{0, 1\}$ such that $b(\mathbf{x}) = 1$ iff $\mathbf{x}$ is *pre-labeled*, i.e. its class value $f(\mathbf{x})$ is known. The structure of X as a data stream is formalized as a sequence of *chunks*, i.e. $X = X_1, X_2, \ldots, X_t, \ldots$ being $X_t \subset \mathbb{R}^n$ a chunk of samples.

Within each chunk, it is possible to apply a clustering procedure in order to find a structure in data. We base our study on Fuzzy C-Means (FCM)[59] clustering because it is particularly suited for numerical data. FCM is a fast and simple fuzzy partitional clustering method that is widely used in literature; its process is based on the minimization of an objective function so as to optimally distribute samples into a pre-defined number of fuzzy clusters. Clusters are represented with their centers $\mathbf{c} \in \mathbb{R}^n$ while data clustering is represented by

a partition matrix U which reports the membership degree of each sample to each cluster. (The membership degrees of a sample to all clusters must sum up to unity.) FCM is a purely unsupervised clustering method and requires an *a-priori* specification of the number of clusters; it is therefore necessary to extend FCM so as to make it suitable for semi-supervised data stream mining.

The clustering process can be guided by the class information available in data, so that the discovered structure is conditioned by the class distribution; however, we must take into account the partial availability of class labels among samples. To this end, we resort to Semi-Supervised Fuzzy C-Means (SSFCM)[23], which embeds partial supervision in the classical FCM clustering algorithm by adapting the objective function as follows:

$$J = \sum_{k=1}^{K} \sum_{j=1}^{N_t} u_{jk}^2 d_{jk}^2 + \alpha \sum_{k=1}^{K} \sum_{j=1}^{N_t} \left(u_{jk} - b_j f_{jk}\right)^2 d_{jk}^2 \tag{1}$$

where $K \geq C$ is the number of clusters, $N_t = |X_t|$ is the cardinality of the t-th chunk in the data stream, $u_{jk} \in [0, 1]$ is the membership degree of a sample $\mathbf{x}_j$ in the k-th cluster, $d_{j,k}$ is the Euclidean distance between sample $\mathbf{x}_j$ and center $\mathbf{c}_k$ of the k-th cluster, $\alpha \geq 0$ is a regularization parameter for the second part of the objective function that exploits class information, $b_j = b(\mathbf{x}_j)$ and $f_{jk} = 1$ iff the j-th sample belongs has the same class label of the k-th cluster. As stated in [23] the choice of α depends on the relative number of labeled and unlabeled data. Usually the number of the latter is much higher than the number of labeled data. To ensure that the impact of the labeled data is not ignored, the value of α should produce approximately equal weighting of the two additive components of J; this suggests that α be proportional to the rate N/M where N is the number of data and M denotes the number of labeled data. In this work we have fixed $\alpha = N/M$.

It is required that each cluster is associated to a class; this is accomplished as follows: (i) at the beginning of clustering, K pre-labelled samples are randomly selected, which will define the initial values of the cluster centers (class distribution is preserved); (ii) being these samples pre-labelled, each cluster is tagged with the class label of the corresponding sample; (iii) the class label is preserved throughout the optimization process, so that it is always possible to compute the value of f_{jk}. As a consequence, the second term of the objective function is minimized if all samples are close to prototypes of the same class.

The objective function (1) must be minimized subject to the constraint

$$\forall j : \sum_{k=1}^{K} u_{jk} = 1$$

which leads to the following formulas to be applied in an alternating optimization scheme[23]:

$$u_{jk} = \frac{1}{1+\alpha} \left[\frac{1 + \alpha(1 - b_j \sum_{h=1}^{K} f_{jh})}{\sum_{h=1}^{K} d_{jk}^2 / d_{jh}^2} \right] + \alpha b_j f_{jk} \tag{2}$$

and

$$\mathbf{c}_k = \frac{\sum_{j=1}^{N_t} u_{jk}^2 \mathbf{x}_j}{\sum_{j=1}^{N_t} u_{jk}^2} \tag{3}$$

Eqs. (2) and (3) are iteratively applied to update both centers and the partition matrix, as in [60].

We used SSFCM to data stream clustering by applying the algorithm to each chunk.[22] For each chunk, cluster prototypes are computed, which can be seen as representatives of local data. In particular, for each cluster a medoid $\mathbf{p}_k$ is computed as the closest data sample to the center $\mathbf{c}_k$ and is tagged with the class of the corresponding cluster. In order to take into account the evolution of data in successive chunks, the calculated prototypes for a chunk are used as pre-labeled prototypes in the next chunk. The labeled prototypes obtained at each step of the incremental clustering are used for classification, namely all data samples maximally belonging[1] to the k-th cluster are associated with the class label assigned to prototype $\mathbf{p}_k$.

The main limitation of SSFCM stands in the choice of the number of cluster, which is fixed throughout the data chunks. This has a twofold disadvantage: (i) the shape of clusters may not follow the distribution of classes, resulting in model under-fitting; (ii) the distribution of classes my evolve throughout chunks, thus requiring reshaping the data clusters. We therefore extended SSFCM by introducing the ability of evolving the clustering structure also in terms of number of clusters.[24]

The resulting method, called Dynamical Incremental SSFCM (DISSFCM) adjusts the number of clusters by splitting low-quality clusters. The quality of clusters is measured by the *reconstruction error*[61], which evaluates the ability of a clustering model in reconstructing the original data samples. Formally, the reconstruction error for cluster C_k is defined as:

$$V_k = \frac{1}{q} \sum_{\mathbf{x}_j \in C_k} \|\mathbf{x}_j - \hat{\mathbf{x}}_j\|^2 \tag{4}$$

where $q = \sqrt{\sum_{i=1}^{n} \sum_{l=1}^{N_t} |x_{il}|^2}$ and the reconstructed data $\hat{\mathbf{x}}_j$ is computed as:

$$\hat{\mathbf{x}}_j = \frac{\sum_{k=1}^{K} u_{jk}^2 \mathbf{p}_k}{\sum_{k=1}^{K} u_{jk}^2} \tag{5}$$

The reconstruction error finds motivation from the granulation–degranulation strategy [62] that uses the prototypes to represent each data point by computing the corresponding membership degrees (granulation step) and then estimates the data point on a basis of the prototypes and the membership degrees (degranulation step). Ideally, for a good granulation of data we would expect that the result of the de-granulation step should return the original data points. In practice we can consider the distance between the original data point $\mathbf{x}_j$ and its de-granulated (reconstructed) version $\hat{\mathbf{x}}_j$ as a good measure of quality of the obtained granulation.

As an overall measure of the cluster quality we use the maximum reconstruction error, given by:

$$V_{max} = \max_k V_k \tag{6}$$

If the prototypes derived from a data chunk do not meet the reconstruction criterion, i.e. if the reconstruction error V_{max} exceeds the value computed on

[1] A data sample maximally belongs to a cluster if the prototype of this cluster is the closest to the sample among all prototypes. Ties are solved randomly.

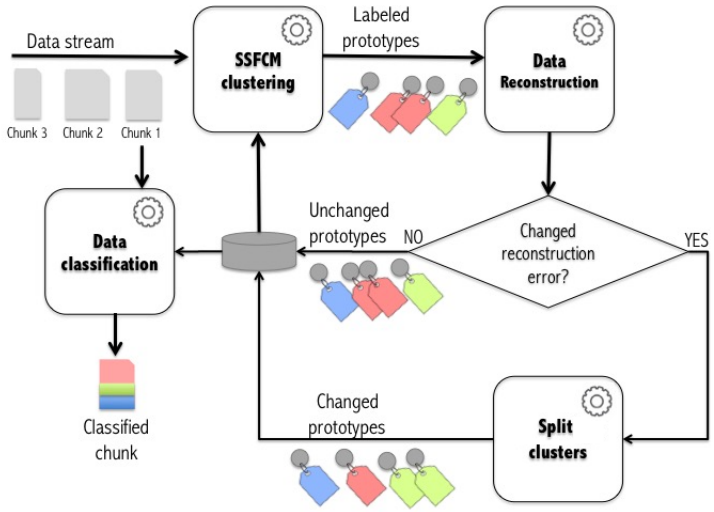


Figure 1: Scheme of DISSFCM

the previous chunk, then some clusters need to be split so as to increase the number of clusters. This is done by splitting the cluster with the highest value of reconstruction error.

Summarizing, the proposed adaptive incremental approach has the following workflow (fig. 1). When a new chunk is available, the SSFCM clustering is applied to generate a number of labeled prototypes. Prototypes are used to reconstruct the data and the reconstruction error is evaluated. If the reconstruction error increases with respect to the previous chunk, then a splitting is applied and the resulting prototypes are stored. The prototypes are then used to classify data in the incoming chunk.

The steps of DISSFCM are detailed in algorithm 1. The algorithm requires the data stream granulated into chunks and an initial collection of labeled prototypes such that each class label is represented by at least one prototype. After application of SSFCM clustering (Step 4) the resulting prototypes are labeled automatically due to the semi-supervised nature of SSFCM. The derived prototypes are the basis for the classification process (Step 10). Indeed, the derived labeled prototypes are used to classify all the data in the current chunk via a matching mechanism. Namely, each data sample is matched[2] against all prototypes and assigned to the class label of the best-matching prototype. At the end, the algorithm returns the most recent collection of the prototypes, reflecting the data structure of the last data chunk. Notice that the returned collection can be used as input for a new run of the algorithm as long as new data are available from the data stream. The split function used in step 7 of the algorithm is detailed below.

[2]The matching mechanism is based on the standard Euclidean distance.

Algorithm 1 DISSFCM

Require: Data stream of chunks $X_1, \ldots, X_t, \ldots$ containing some labeled data belonging to C classes

Require: Initial set P_0 of K labeled prototypes containing at least one prototype per class;

Ensure: P: labeled prototypes;

```
t ← 1
P ← P_0
while ∃ nonempty chunk X_t do
    P ← SSFCM(X_t, P)
    Compute the reconstruction error V_max^(t)
    while V_max^(t) > V_max^(t−1) do
        P ← split(P, X_t)
        Compute the reconstruction error V_max^(t)
    end while
    Classify data in X_t using labeled prototypes in P
    t ← t + 1
end while
```

3.1 Splitting

The splitting mechanism is activated when the reconstruction error on the current chunk exceeds the reconstruction error computed on the previous chunk. This means that the current number of clusters is not enough to effectively represent the data, hence the number of clusters should be augmented. This is done by splitting one cluster in two parts, so as to form two new clusters.

The cluster having the highest value of the reconstruction error, i.e. the cluster with lowest reconstruction ability, is selected as candidate for splitting. We denote by k^* the selected cluster. Its splitting is performed by means of the *conditional fuzzy clustering*[63] applied to the collection of data samples belonging to the cluster so as to create two novel clusters. Given the set C_{k^*} of data samples belonging to the selected cluster k^*, the splitting process is performed by the minimization of the following objective function

$$J_c = \sum_{k=1}^{2} \sum_{\mathbf{x}_j \in C_{k^*}} v_{jk}^2 \|\mathbf{x}_j - \mathbf{z}_k\|^2 \tag{7}$$

under the constraint

$$\sum_{k=1}^{2} v_{jk} = u_{jk^*}$$

where v_{jk} is the membership degree of $\mathbf{x}_j$ to the cluster k, $k = 1, 2$.

The objective function (7) is minimized by iteratively computing the membership values v_{jk} and the prototypes $\mathbf{z}_k$ according to:

$$v_{jk} = \frac{u_{jk^*}}{\sum_{c=1}^{2} \left(\frac{\|\mathbf{x}_j - \mathbf{z}_k\|}{\|\mathbf{x}_j - \mathbf{z}_c\|} \right)^2} \tag{8}$$

and

$$\mathbf{z}_k = \frac{\sum_{\mathbf{x}_j \in C_{k*}} v_{jk}^2 \mathbf{x}_j}{\sum_{\mathbf{x}_j \in C_{k*}} v_{jk}^2}, k = 1, 2 \tag{9}$$

Iterative application of (8) and (9) stops when there is no significant decrease of objective function J_c. Once the new clusters are generated, they inherit the class label of the original cluster (which is no more used) and membership degrees are re-computed according to (2). The splitting procedure is repeated until the reconstruction error drops below the reconstruction error of the previous chunk.

4 Experimental results

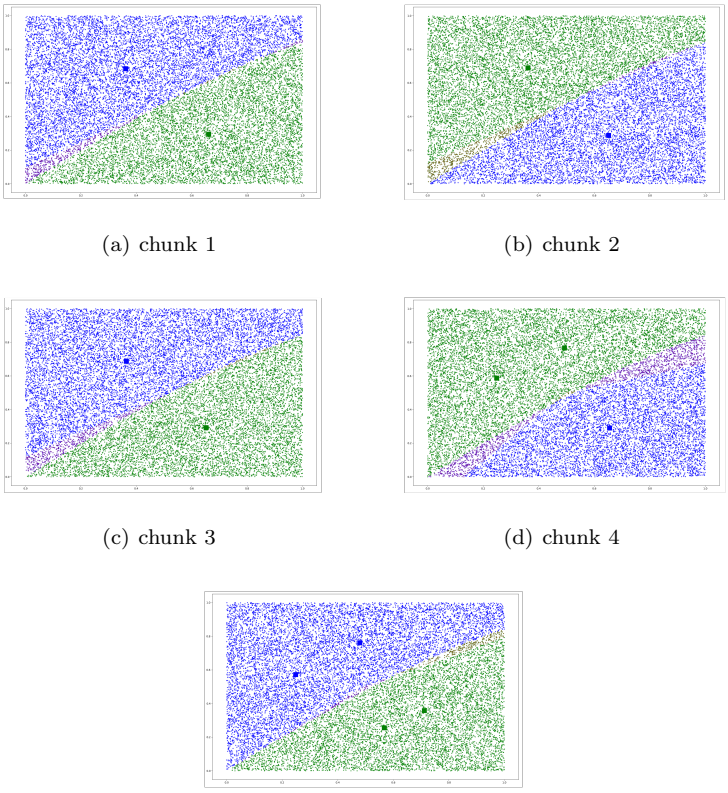
Numerical experiments were conducted to evaluate the effectiveness of the proposed algorithm in data stream classification. Synthetic data were firstly analyzed to better understand how the algorithm behaves when a change in the class distribution occurs. Then real data were considered to analyze the classification ability of DISSFCM.

4.1 Synthetic data

The *Sine1* dataset[3] has been considered to analyze the behavior of DISSFCM in presence of abrupt concept drift, i.e. sudden changes happen in data class distribution. It is a two dimensional synthetic dataset of uniformly distributed points in the interval $[0, 1]$. The sine function $y = \sin(x)$ is used to classify the instances as positive if they are under the sine curve, negative otherwise. The data are divided in five chunks in order to have a drift point (i.e. the reversion of the class labels) in each chunk.

Figure 2 shows the five chunks and the prototypes produced by DISSFCM when each chunk arrives. Two colors (blue and green) are used for the two output classes. Data points are represented by dots and prototypes by squares. The color red is used to highlight misclassifications. When the process starts, one cluster prototype is associated to each class. At the end of the computation, DISSFCM separates the data in two classes, being each of them described by the prototypes and the membership values. We can observe few misclassified data near the class separation border (fig. 2(a)). When the next chunk arrives, the class positions are reversed as the green class moves to the upper part of the graph whilst the blue one goes to the bottom. It can be seen that DISSFCM is able to recognize the changed class distribution and to switch the prototypes according to the classes they belong to (fig. 2(b)). Moreover, since the number of misclassified data after processing the third chunk has increased (fig. 2(c)), when the next chunk arrives the green cluster (with the highest reconstruction error) is split and two new prototypes are generated (fig. 2(d)). Finally, when the last chunk arrives, the blue cluster is split and two new blue prototypes (with corresponding clusters) are generated. The new model better fits the data structure and the error is consequently reduced (few red dots on the last graph fig. 2(e)).

[3] https://github.com/alipsgh/data_streams



(a) chunk 1 (b) chunk 2

(c) chunk 3 (d) chunk 4

(e) chunk 5

Figure 2: Sine1 synthetic dataset processed by DISSFCM.

In summary, DISSFCM is able to adapt the cluster-based model according to the structure of incoming data, even when an abrupt concept drift occurs. Moreover, when the current clusters do not sufficiently represent new data chunks, the granularity of the model is refined as new clusters are generated.

4.2 Real-world data

The Occupancy Detection Data Set[4] by Candanedo *et al.*, [64] was used to test the effectiveness of DISSCM for classifying real-world temporal data.

The goal of this dataset is to detect the absence of people in the room, in order to switch off lights, ventilation and all the tools that require energy. Since the presence/absence of humans can be used for saving energy and safety purposes, accurate automatic occupancy detection in buildings has recently grown attention[64] focusing on the assessment of smart spaces that are aware of their state and can act accordingly. Sensors are commonly used to measure the state of a room in terms of light, temperature, humidity and CO_2. Indeed, they allow to automatically infer the presence/absence of people in the room.

[4]https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+

Data set	Observations	Class Distribution	
		0 (non-occupied)	1 (occupied)
Training	8134	79%	21%
Testing 1	2665	64%	36%
Testing 2	9752	79%	21%

Table 1: Data set description.

	Time slots	#chunks	chunk size
S1	[00.00, 23.58]	7	1440
S2	[00.00, 11.58] [12.00, 23.58]	12	720
S3	[00.00, 07.59] [08.00, 15.59] [16.00, 23.58]	17	479

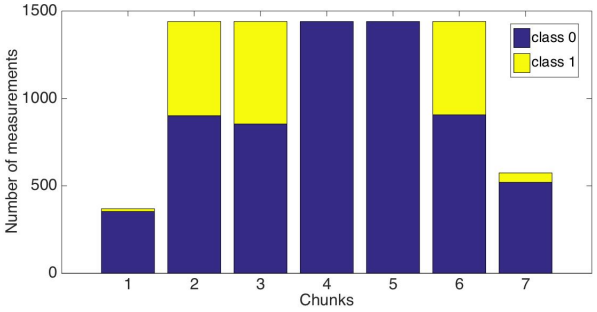
Table 2: Different structures of the data stream considered for experiments.

In the considered dataset, each data sample is defined by five features: Temperature (Celsius), Relative Humidity (%), Light (Lux), CO2 (ppm), Humidity Ratio (kg of water-vapor/kg of air), which have been measured every minute in a room in the period from Feb. 3rd, 2015 to Feb. 10th, 2015. Each sample is classified in one out of two possible classes: 0 (non-occupied room) and 1 (occupied room). As a consequence, the dataset is structured as a stream of 20,551 chronologically sorted data samples, which have been divided in a training set and two test sets differing by the room conditions (in the first one, as in the training set, the measurements have been taken mostly with the door closed during occupied status, whilst in the second one the measurements have been taken mostly with the door open during occupied status). Table 1 reports some basic statistics on data.

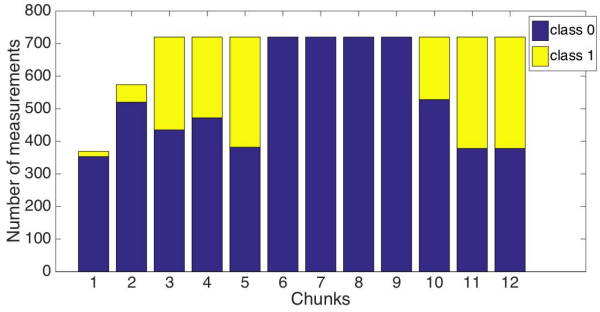
We considered temporal slots to partition the training set in chunks, in order to simulate a data stream. As reported in Table 2, we considered three different structures of the data stream (S1, S2, S3) by defining chunks (time slots) of three different granularities. This lead to the emergence of different class distributions in the chunks. Specifically, we considered daily time slots (S1), half-day slots (S2) and one-third-of-day slots (S3) so as to roughly separate working hours from the rest of the day. Figure 3 shows the class distribution for each chunk in the three experiments. It can be observed that class distributions change gradually over time, which can be referred as a gradual concept drift. There are few abrupt concept drifts in correspondence of days when there was no occupancy at all. Moreover, abrupt concept drifts are more apparent in S3 where the chunks are made of fewer data, leading to higher variability in class distributions.

Standard classification measures have been used to evaluate the classification performance of DISSFCM, namely accuracy, precision, recall and F-measure. Moreover the trend of the reconstruction error during the computation of the chunks has been considered to evaluate how the model fits with the observed data over each chunk.

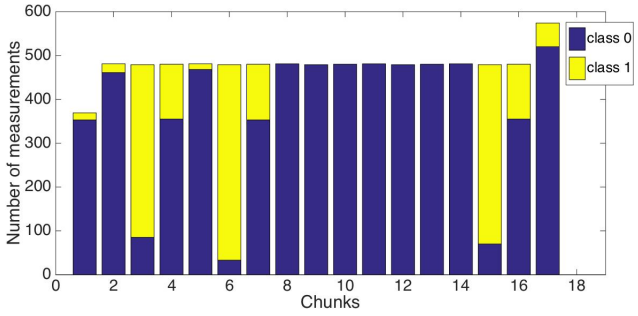
In the following, we first show the behavior of DISSFCM when dealing with different stream structures and concept drifts. Then, we report an extensive experiment to show the effectiveness of the proposed method under different percentage of labeled data. We finally compare DISSFCM results with other state-of-art classification algorithms.



(a)



(b)



(c)

Figure 3: Class distribution through chunks in different structures of the data stream: S1 (a), S2 (b) and S3 (c).

4.2.1 First experiment

Preliminary simulations were devoted to analyze the behavior of DISSFCM in presence of gradual and abrupt concept drifts. In these simulations we considered daily chunks (stream S1) and the percentage of labeled data was fixed to 25%. After processing a single data chunk we computed the classification accuracy on the current chunk, as well as on the previous chunks. Of course the algorithm does not require access to previously used data during subsequent incremental learning sessions, but the accuracy on previous chunks was evaluated to show how the method is able to retain the previously acquired knowledge. Table 3 reports the accuracy values achieved by DISSFCM on the training set. It can be seen that when a new chunk is processed a good accuracy on the current chunk is achieved and the accuracy on the previous chunks is preserved as well. This means that the model is successfully adapted to new data without forgetting the knowledge acquired from previous data. The lowest accuracy values at times t_6 and t_7 are observed for the third chunk X_3 when a big change in data distribution is occurred (chunks X_4 and X_5), nevertheless the model well describes the previous chunks. A connection between the changes in data distribution and the algorithm effectiveness could be observed in the trend of the reconstruction error in figure 4.

As reported in figure 3(a), there is a gradual concept drift between the first and the second chunk. Indeed few data belong to class 1 in chunk 1, while almost half of the data belong to the each class in the second chunk. This change produces an increment of the reconstruction error when the second chunk arrives, because the model is no more adequate to describe the current data. For this reason the splitting is activated, so the number of prototypes increases by one ($K = 3$) and the maximum reconstruction error decreases (requiring no more splits). When the third chunk is presented the class distribution does not change, the model fits the new data, and the error decreases. We then observe an abrupt concept drift when the fourth chunk comes (an unexpected change in the class distribution: the first class is no more represented). The old model does not describe the new data, so the error rises again. A novel splitting is applied and four prototypes are used to describe this new configuration. When the fifth chunk comes, it has the same distribution of the previous one, the model is able to describe these new data and so no error increase is observed. On the contrary, with the seventh chunk another class distribution change occurs, the error raises, a splitting is required ($K = 5$). Finally, even if the class distribution in the last chunk is quite different from that in the previous one, since both classes are represented, the algorithm is able to move the prototypes in order to describe the new data.

4.2.2 Second experiment

The second set of simulations was devoted to evaluate the accuracy of DISSFCM when changing the chunk composition (S1, S2, S3) and the percentage of labeled data (25%, 50%, 75%, 100%). The main goal of this study was to observe the influence of the labeling percentage on the classification process. In table 4 we show the average accuracy of all the cluster-based models derived during chunk processing on the two test sets. Figure 5 shows the trend of the accuracy. As it can be expected, once fixed the number of chunks, the more the labeled

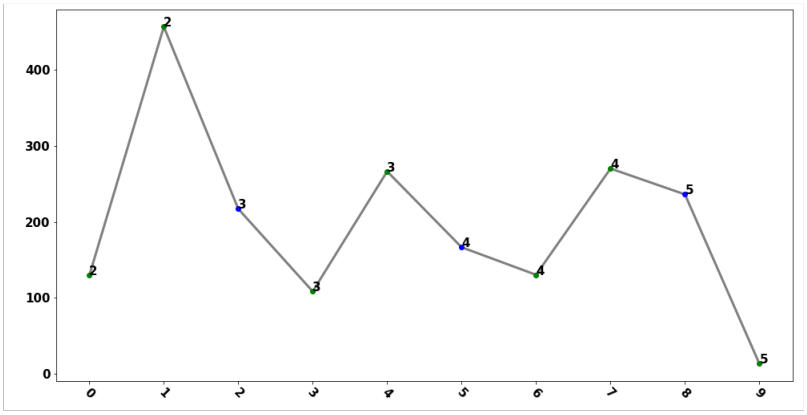


Figure 4: Trend of the reconstruction error V_{max} (S1, %Labeling=25%). Green dots correspond to the error upon arrival of new chunks, the blue ones indicate the error after a split. Numbers on the dots indicate the number of cluster prototypes.

data are, the higher the accuracy is. Moreover, it can be observed that the number of chunks (hence their cardinality and class distribution) emphasizes the influence of the percentage of labeled data. In fact, when the class distribution does not change across the chunks, the same results are obtained regardless the percentage of labeled data. When a significant change in the distribution occurs (chunk 4 in S1, chunks $6-9$ in S2 and chunks $8-14$ in S3) then the results differ significantly depending on the labeling percentage.

We also computed precision, recall and F-measure of the target class (i.e. no occupancy) in order to assess the usefulness of the classification models in a real-world scenario. Tables 5-7 and figures 6-8 report these metrics. We observe high values of precision ((≥ 0.90) for streams S1 and S2. This is a good result from the application point of view. Actually, high precision values mean that in

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
K	2	2	3	3	4	4	5
X_1	1	1	1	0.957	1	0.957	1
X_2	-	0.95	0.95	0.84	0.95	0.85	0.90
X_3	-	-	0.99	0.99	0.94	0.59	0.59
X_4	-	-	-	0.81	1	1	1
X_5	-	-	-	-	0.93	1	1
X_6	-	-	-	-	-	0.97	0.98
X_7	-	-	-	-	-	-	1

Table 3: Accuracy obtained on single chunks during the incremental process (S1, %Labeling=25%).

most cases the classifier correctly recognizes that there are no people in the room when it is really empty. As a consequence, a sensor control system embodying the classification model may correctly activate energy saving without causing annoyance to users. In case S3 the performance of the classifier decreases when data are partially labeled, while it is quite high when all data are labeled. On the overall, we found that large data chunks make the classification models less sensitive to the percentage of labeled data as well as to concept drifts, both abrupt and gradual. Similar behavior is observed for recall: almost every time there are actually no people in the room the system is able to recognize the correct state. This allows sensible energy saving.

	Test set 1			Test set 2		
%labeling	S1	S2	S3	S1	S2	S3
25	0.93(0.06)	0.91(0.08)	0.79(0.23)	0.91(0.05)	0.91(0.06)	0.73(0.26)
50	0.93(0.06)	0.90(0.15)	0.74(0.32)	0.91(0.05)	0.90(0.07)	0.76(0.25)
75	0.93(0.06)	0.90(0.12)	0.70(0.34)	0.91(0.05)	0.90(0.05)	0.82(0.14)
100	0.95(0.03)	0.94(0.04)	0.95(0.07)	0.94(0.04)	0.91(0.05)	0.90(0.10)

Table 4: Average classification accuracy (and standard deviation) on the test sets.

	Test set 1			Test set 2		
%labeling	S1	S2	S3	S1	S2	S3
25	0.94(0.08)	0.91(0.11)	0.91(0.19)	0.96(0.07)	0.94(0.08)	0.95(0.09)
50	0.94(0.09)	0.90(0.14)	0.85(0.27)	0.96(0.07)	0.93(0.09)	0.97(0.07)
75	0.94(0.09)	0.91(0.11)	0.77(0.31)	0.96(0.07)	0.94(0.08)	0.91(0.10)
100	0.97(0.06)	0.96(0.07)	0.96(0.09)	0.97(0.04)	0.96(0.06)	0.96(0.06)

Table 5: Average precision (and standard deviation) for the target class 0 (no occupacy) on the test sets.

	Test set 1			Test set 2		
%labeling	S1	S2	S3	S1	S2	S3
25	0.96(0.02)	0.97(0.01)	0.73(0.38)	0.93(0.06)	0.95(0.05)	0.69(0.35)
50	0.96(0.02)	0.94(0.12)	0.75(0.36)	0.93(0.06)	0.95(0.06)	0.75(0.29)
75	0.96(0.02)	0.95(0.06)	0.76(0.33)	0.93(0.06)	0.95(0.06)	0.87(0.14)
100	0.97(0.02)	0.96(0.02)	0.96(0.02)	0.95(0.06)	0.91(0.07)	0.90(0.12)

Table 6: Average recall (and standard deviation) on the target class 0 (no occupacy) on the test sets.

	Test set 1			Test set 2		
%labeling	S1	S2	S3	S1	S2	S3
25	0.95(0.04)	0.94(0.06)	0.74(0.34)	0.94(0.03)	0.94(0.03)	0.74(0.30)
50	0.95(0.04)	0.91(0.12)	0.77(0.34)	0.94(0.03)	0.93(0.04)	0.81(0.22)
75	0.95(0.04)	0.93(0.08)	0.76(0.32)	0.94(0.03)	0.94(0.03)	0.88(0.10)
100	0.96(0.02)	0.96(0.03)	0.96(0.05)	0.96(0.03)	0.93(0.03)	0.92(0.07)

Table 7: Average F-measure (and standard deviation) for the target class 0 (no occupancy) on the test sets.

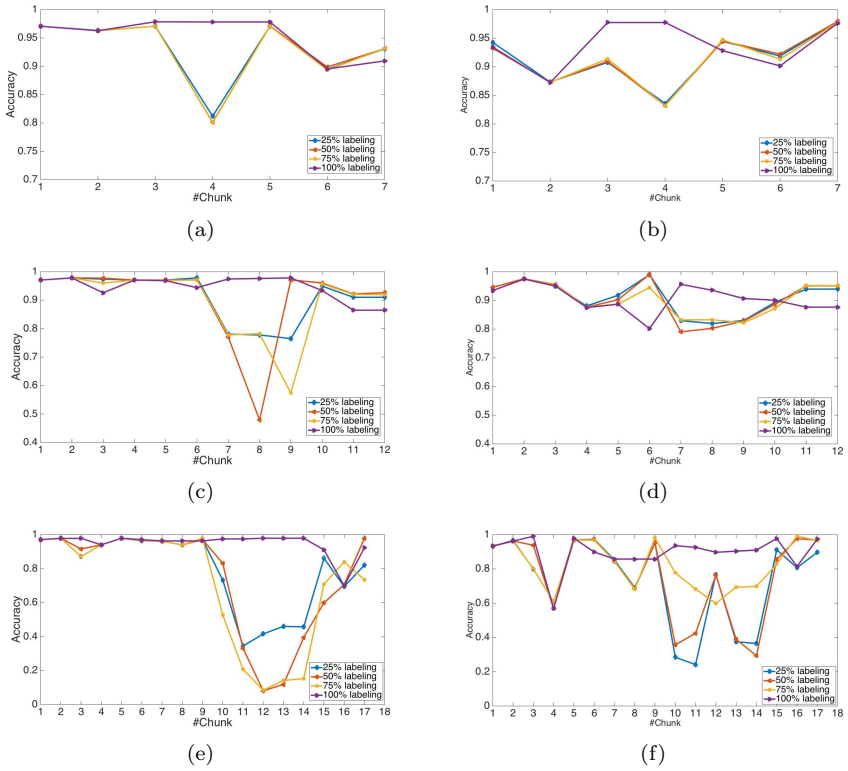


Figure 5: Accuracy on test set 1 and test set 2 during the processing of streams S1 (a) and (b), S2 (c) and (d), S3 (e) and (f).

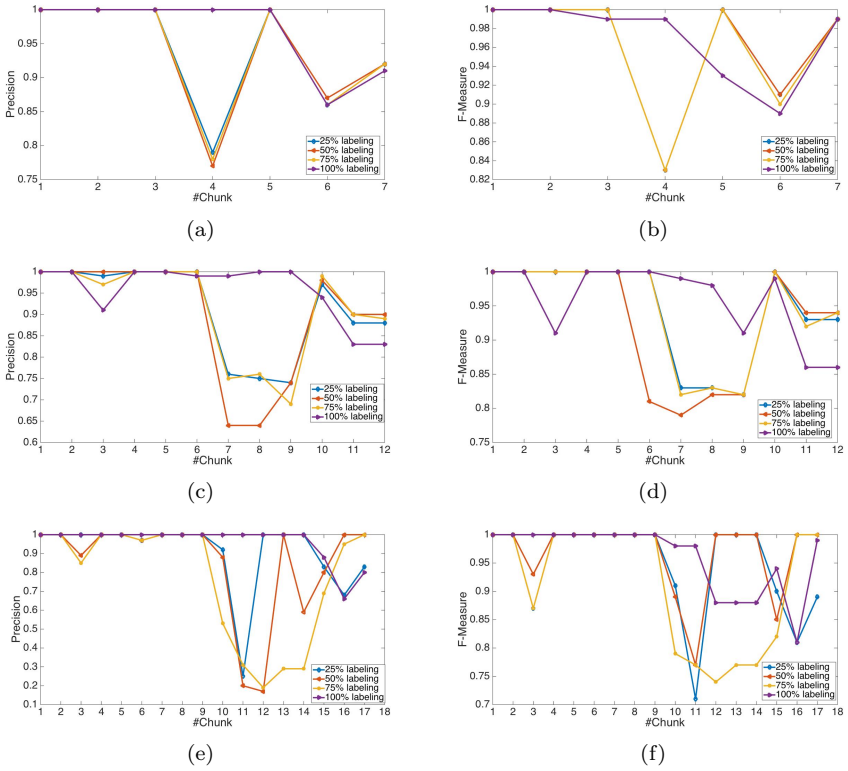


Figure 6: Precision on test set 1 and 2 during the processing of streams S1 (a) and (b), S2 (c) and (d) and S3 (e) and (f).

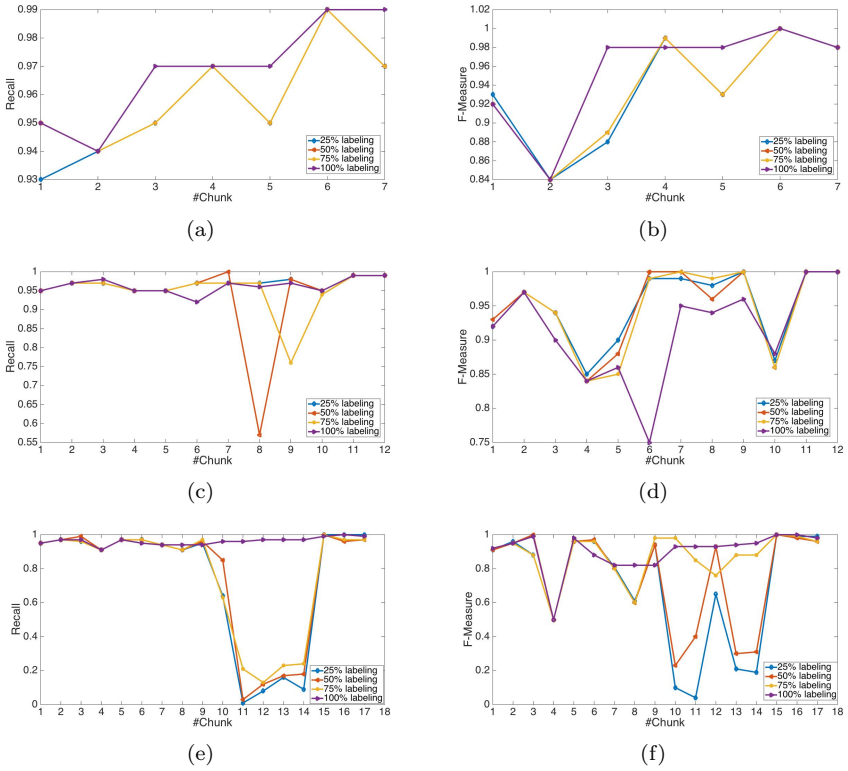


Figure 7: Recall on test set 1 and 2 during the processing of streams S1 (a) and (b), S2 (c) and (d) and S3 (e) and (f).

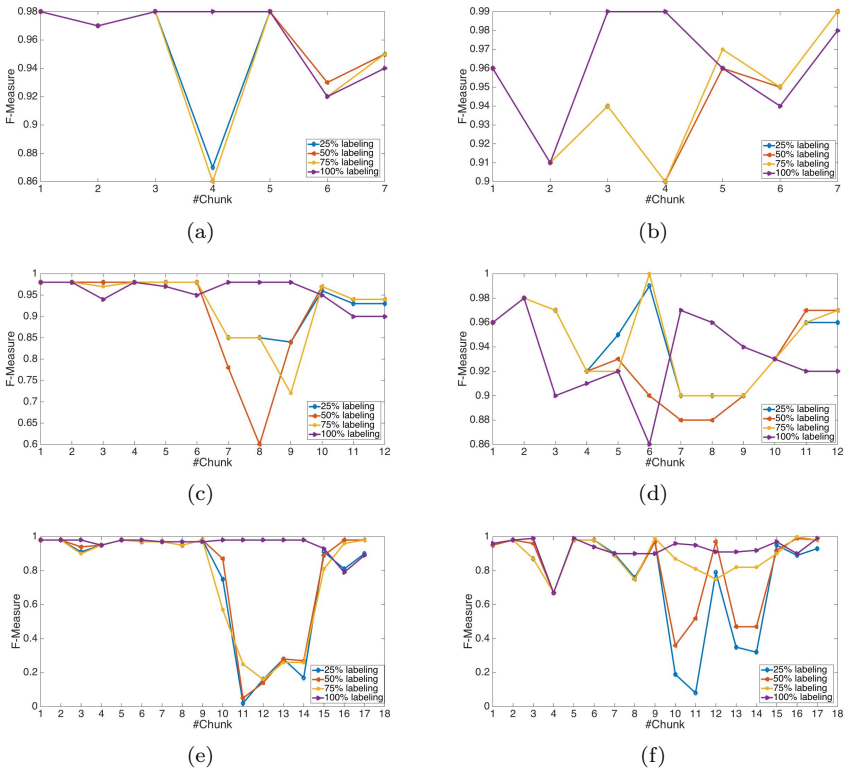


Figure 8: F-measure on test set 1 and 2 during the processing of streams S1 (a) and (b), S2 (c) and (d) and S3 (e) and (f).

4.2.3 Comparison

Finally, the proposed DISSFCM was compared with some standard classification algorithms. We considered the comparative study made in [64], where Random Forest (RF), Gradient Boosting Machines (GBM), Linear Discriminant Analysis (LDA) and Classification and Regression Trees (CART) are compared on the Occupancy dataset. Since all the considered algorithms are based on supervised learning, we applied DISSFCM with 100% of labeled data for a fair comparison. Table 8 reports the accuracy on test set 1 and test set 2, for the standard classifiers and for the classifiers built by DISSFCM on the three streams (S1, S2, S3). It can be observed that the accuracy of DISSFCM-based classifiers compares well with standard classification methods. In addition, we point out that DISSFCM is able to create incrementally the classification model using portions of data, while standard methods construct the model using the whole dataset. In real scenarios, where the whole data are not always available, this ability of DISSFCM turns out to be very useful. It can enable the development of a control system that performs real-time occupancy detection to assure energy saving.

Model	Test set	
	1	2
RF	95.05	97.16
GBM	93.06	95.14
CART	95.57	96.47
LDA	97.90	98.76
DISSFCM (S1)	95.34	93.83
DISSFCM (S2)	94.59	90.63
DISSFCM (S3)	94.86	89.68

Table 8: Accuracy comparison

5 Conclusions

In this work we have presented DISSFCM, a dynamic incremental semi-supervised version of the classical fuzzy c-means (FCM) clustering algorithm. DISSFCM extends FCM in a number of ways. First, it enables semi-supervision as a powerful means to inject knowledge in the clustering process, by driving the grouping of data through the pre-classification of a subset of data. Second, clustering becomes incremental, i.e. it does not require the whole dataset beforehand: chunks of data are enough to proceed with clustering and prototypes are eventually adapted to follow the evolution of structure in data. Third, the structure of clustering is dynamic: when the reconstruction error of data given a clustering structure becomes inadequate, the most troublesome clusters are split into finer grained clusters that better represent data.

Preliminary results on a real-world dataset show the potentialities of DISSFCM in data stream analytics: data chunks are processed just once, therefore there is no need to store the entire dataset in memory. Notwithstanding, the final results are comparable to batch algorithms in terms of classification accuracy. Moreover, results on both synthetic and real-world data show the ability of DISSFCM to cope with gradual and abrupt concept drift, especially when labeled data are available. All the above features make DISSFCM a good candidate for classification tasks in real-world applications where very large amounts of data are continuously available as a stream. In particular, we have shown the effectiveness of DISSFCM in a real-world problem of occupancy detection based on sensor data.

Further work is devoted to analyze deeper the influence of the chunk size/composition on DISSFCM, so as to better take into account real data stream scenarios, where the incoming chunks may have different sizes and may contain data with inhomogeneous class distributions. Future research is going on along the direction of merging small-scale clusters, which may hamper the quality of clustering in terms of structure and interpretability. Also, automated techniques for detecting concept evolution (i.e. the emergence of new classes or the disappearance of existing ones) are a subject of further studies.

Acknowledgments

The research is partially supported by Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) under grant PON ARS01_01116 "TALIsMAn". The

authors are members of the INdAM Research group GNCS.

References

[1] Paul Zikopoulos, Chris Eaton, Dirk DeRoos, Thomas Deutsch, and George Lapis. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 1st edition, 2011.

[2] João Gama and Mohamed Medhat Gaber. *Learning from data streams: processing techniques in sensor networks*. Springer, 2007.

[3] G. Piccinni, G. Avitabile, and G. Coviello. A novel distance measurement technique for indoor positioning systems based on zadoff-chu sequences. In *2017 15th IEEE International New Circuits and Systems Conference (NEWCAS)*, pages 337–340, 2017.

[4] Georg Krempl, Myra Spiliopoulou, Jerzy Stefanowski, Indre Žliobaite, Dariusz Brzeziński, Eyke Hüllermeier, Mark Last, Vincent Lemaire, Tino Noack, Ammar Shaker, and Sonja Sievi. Open challenges for data stream mining research. *ACM SIGKDD Explorations Newsletter*, 16(1):1–10, 9 2014.

[5] Ee-Peng Lim, Hsinchun Chen, and Guoqing Chen. Business intelligence and analytics: Research directions. *ACM Transactions on Management Information Systems (TMIS)*, 3(4):17, 2013.

[6] Xu Tian, Qiong Sun, Xiaohong Huang, and Yan Ma. A dynamic online traffic classification methodology based on data stream mining. In *Computer Science and Information Engineering, 2009 WRI World Congress on*, volume 1, pages 298–302. IEEE, 2009.

[7] Alessandro Bianchi, Sebastiano Pizzutilo, and Gennaro Vessio. Intercepting blackhole attacks in manets: An asm-based model. In *International Conference on Software Engineering and Formal Methods*, pages 137–152. Springer, 2017.

[8] Beth Plale, Dennis Gannon, Jerry Brotzge, Kelvin Droegemeier, Jim Kurose, David McLaughlin, Robert Wilhelmson, Sara Graves, Mmohan Ramamurthy, Richard D. Clark, Sepi Yalda, Daniel A. Reed, Everette Joseph, and V Chandrasekar. CASA and LEAD: adaptive cyberinfrastructure for real-time multiscale weather forecasting. *Computer*, 39(11):56–64, 11 2006.

[9] Yanxin Ji, Chengwei Mi, Feng Gao, Fang Deng, and Chao Zheng. Wearable Human Health Monitoring System. In *2018 37th Chinese Control Conference (CCC)*, pages 7256–7261. IEEE, 2018.

[10] Svitlana Petrasova, Nina Khairova, and Wlodzimierz Lewoniewski. Building the Semantic Similarity Model for Social Network Data Streams. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, pages 21–24. IEEE, 2018.

[11] G Casalino, C Castiello, N Del Buono, and C Mencar. A Framework for Intelligent Twitter Data Analysis with Nonnegative Matrix Factorization. *International Journal of Web Information Systems*, 14(3), 2018.

[12] P. Ardimento, M. L. Bernardi, M. Cimitile, and G. De Ruvo. Mining developer's behavior from web-based ide logs. In *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 277–282, 2019.

[13] Mohamed M. Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining Data Streams: A Review. *SIGMOD Rec.*, 34(2):18–26, 6 2005.

[14] Manoj B. Chandak. Role of big-data in classification and novel class detection in data streams. *Journal of Big Data*, 3(1):5, 12 2016.

[15] Min Chen, Shiwen Mao, and Yunhao Liu. Big Data: A Survey. *Mobile Networks and Applications*, 19(2):171–209, 1 2014.

[16] Jonathan de Adrade Silva, Elaine Ribeiro de Faria, Rodrigo C Barros, Eduardo R Hruschka, André Carlos Ponce de Leon Ferreira De Carvalho, and João Gama. Data Stream Clustering: A Survey. {*ACM*} *Comput. Surv.*, 46(1):13:1–13:31, 2013.

[17] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Data Stream Mining. In *Data Mining and Knowledge Discovery Handbook*, volume 8, pages 759–787. Springer US, Boston, MA, 2009.

[18] Edwin Lughofer and Mahardhika Pratama. Online Active Learning in Data Stream Regression Using Uncertainty Sampling Based on Evolving Generalized Fuzzy Models. *IEEE Transactions on Fuzzy Systems*, 26(1):292–309, 2 2018.

[19] Richard Hyde, Plamen Angelov, and Angus Robert MacKenzie. Fully online clustering of evolving data streams into arbitrarily shaped clusters. *Inf. Sci.*, 382-383:96–114, 2017.

[20] Eleftherios Spyromitros Xioufis, Myra Spiliopoulou, Grigorios Tsoumakas, and Ioannis P Vlahavas. Dealing with Concept Drift and Class Imbalance in Multi-Label Stream Classification. In *IJCAI*, 2011.

[21] Mohamed Medhat Gaber, Arkady B Zaslavsky, and Shonali Krishnaswamy. A Survey of Classification Methods in Data Streams. In *Data Streams - Models and Algorithms*, 2007.

[22] Giovanna Castellano and Anna Maria Fanelli. Classification of Data Streams by Incremental Semi-supervised Fuzzy Clustering. In *Int. Workshop on Fuzzy Logic and Applications*, volume 10147 of *Lecture Notes in Computer Science*, pages 185–194, 2016.

[23] Witold Pedrycz. Algorithms of Fuzzy Clustering with Partial Supervision. *Pattern Recogn. Lett.*, 3(1):13–20, 1 1985.

[24] Gabriella Casalino, Giovanna Castellano, and Corrado Mencar. Incremental adaptive semi-supervised fuzzy clustering for data stream classification. In *Proc. of the 2018 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS 2018)*, pages 1–7, Rhodes, Greece, 5 2018.

[25] Beian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and Issues in Data Stream Systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 1–16, New York, NY, USA, 2002. ACM.

[26] Ludmila I. Kuncheva. Classifier Ensembles for Changing Environments. In F. Roli, J. Kittler, and T. Windeatt, editors, *International Workshop on Multiple Classifier Systems. MCS 2004*, volume 3077 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2004.

[27] M. Kehinde Olorunnimbe, Herna L. Viktor, and Eric Paquet. Dynamic adaptation of online ensembles for drifting data streams. *J. Intell. Inf. Syst.*, 50(2):291–313, 2018.

[28] Plamen Angelov, Edwin D. Lughofer, and Xiaowei Zhou. Evolving fuzzy classifiers using different model architectures. *Fuzzy Sets and Systems*, 159(23):3160–3182, 12 2008.

[29] Mahardhika Pratama, Witold Pedrycz, and Edwin Lughofer. Evolving Ensemble Fuzzy Classifier. *IEEE Transactions on Fuzzy Systems*, 26:2552–2567, 2018.

[30] Gabriella Casalino, Nicoletta Del Buono, and Corrado Mencar. Nonnegative Matrix Factorizations for Intelligent Data Analysis. In G.R. Naik, editor, *Non-negative Matrix Factorization Techniques: Advances in Theory and Applications*, pages 49–75. Springer-Verlag Berlin Heidelberg, 1 edition, 2016.

[31] Jiong Yang, Xifeng Yan, Jiawei Han, and Wei Wang. Discovering an Evolutionary Classifier over a High-speed Nonstatic Stream. In *Advanced Methods for Knowledge Discovery from Complex Data*, pages 337–363. Springer-Verlag, New York, 2003.

[32] Elena Ikonomovska, João Gama, and Saso Dzeroski. Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, 23:128–168, 2010.

[33] Zhihao Zhang and Jie Zhou. Transfer estimation of evolving class priors in data stream classification. *Pattern Recognition*, 43:3151–3161, 2010.

[34] Chunyu Yang and Jie Zhou. Non-stationary data sequence classification using online class priors estimation. *Pattern Recognition*, 41(8):2656–2664, 8 2008.

[35] Shaoning Pang, Seiichi Ozawa, and Nikola K Kasabov. Incremental linear discriminant analysis for classification of data streams. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35:905–914, 2005.

[36] Mario R Guarracino, Salvatore Cuciniello, and Davide Feminiano. Incremental generalized eigenvalue classification on data streams. In *International workshop on data stream management and mining*, pages 1–12, 2009.

[37] P V Srilakshmi Annapoorna and T T Mirnalinee. Streaming data classification. *2016 International Conference on Recent Trends in Information Technology (ICRTIT)*, pages 1–7, 2016.

[38] Heitor Murilo Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106:1469–1495, 2017.

[39] Hai-Long Nguyen, Yew-Kwong Woon, and Wee-Keong Ng Ng. A survey on data stream clustering and classification. *Knowledge and information systems*, 45(3):535–569, 2015.

[40] Maryam Mousavi, Azuraliza Abu Bakar, and Mohammadmahdi Vakilian. Data stream clustering algorithms: A review. *International Journal of Advances in Soft Computing and its Applications*, 7(Specialissue3):1–15, 2015.

[41] Charu C Aggarwal and Philip S Yu. A Framework for Clustering Uncertain Data Streams. *2008 IEEE 24th International Conference on Data Engineering*, pages 150–159, 2008.

[42] Mohammed Ghesmoune, Mustapha Lebbah, and Hanene Azzag. Micro-Batching Growing Neural Gas for Clustering Data Streams Using Spark Streaming. *Procedia Computer Science*, 53:158–166, 2015.

[43] Edwin Lughofer. Evolving Vector Quantization for Classification of On-Line Data Streams. *2008 International Conference on Computational Intelligence for Modelling Control & Automation*, pages 779–784, 2008.

[44] Amr Abdullatif, Francesco Masulli, and Stefano Rovetta. Clustering of nonstationary data streams: A survey of fuzzy partitional methods. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1258, 2018.

[45] Jurgen Beringer and Eyke Hüllermeier. Fuzzy clustering of parallel data streams. *Advances in Fuzzy Clustering and Its Application*, pages 333–352, 2007.

[46] Sara Mostafavi and Ali Amiri. Extending fuzzy c-means to clustering data streams. In *20th Iranian Conference on Electrical Engineering (ICEE2012)*, pages 726–729, 5 2012.

[47] Diksha Upadhyay, Susheel Jain, and Anurag Jain. A Fuzzy Clustering Algorithm for High Dimensional Streaming Data. *Journal of Information Engineering and Applications*, 3(10):1–10, 2013.

[48] Andrea Ferranti, Francesco Marcelloni, Armando Segatori, Michela Antonelli, and Pietro Ducange. A distributed approach to multi-objective evolutionary generation of fuzzy rule-based classifiers from big data. *Information Sciences*, 415-416:319–340, 2017.

[49] Jaspreet Bassan and Marcus Santos. Classifying streaming data using grammar-based immune programming. *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2016.

[50] Indre Zliobaite, Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. MOA concept drift active learning strategies for streaming data. In *Proceedings of the Second Workshop on Applications of Pattern Analysis*, pages 48–55, 2011.

[51] Eric Bair. Semi-supervised clustering methods. *Wiley interdisciplinary reviews. Computational statistics*, 5 5:349–361, 2013.

[52] Xiaojin Zhu. Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[53] Hanen Borchani, Pedro Larrañaga, and Concha Bielza. Classifying evolving data streams with partially labeled data. *Intell. Data Anal.*, 15:655–670, 2011.

[54] Giovanna Castellano, Anna Maria Fanelli, and M A Torsello. Incremental Semi-Supervised Fuzzy Clustering for Shape Annotation. In *Proceedings of the 2014 IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing (SSCI-CIMSIVP 2014)*, pages 190–194, Orlando, Florida, USA, 12 2014.

[55] Jonathan de Andrade Silva and Eduardo Raul Hruschka. A Support System for Clustering Data Streams with a Variable Number of Clusters. *ACM Trans. Auton. Adapt. Syst.*, 11(2):11:1–11:26, 7 2016.

[56] Edvin Lughofer. A dynamic split-and-merge approach for evolving cluster models. *Evolving Systems*, 3(3):135–151, 9 2012.

[57] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. Density-Based Clustering over an Evolving Data Stream with Noise. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 328–339, Philadelphia, PA, 4 2006. Society for Industrial and Applied Mathematics.

[58] Plamen Angelov and Xiaowei Zhou. Evolving Fuzzy Systems from Data Streams in Real-Time. *2006 International Symposium on Evolving Fuzzy Systems*, pages 29–35, 2006.

[59] James C. Bezdek, Robert Ehrlich, and William Full. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[60] Witold Pedrycz and James Waletzky. Fuzzy clustering with partial supervision. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 27(5):787–95, 1997.

[61] Witold Pedrycz. A Dynamic Data Granulation Through Adjustable Fuzzy Clustering. *Pattern Recogn. Lett.*, 29(16):2059–2066, 12 2008.

[62] Witold Pedrycz and Kaoru Hirota. Fuzzy vector quantization with the particle swarm optimization: A study in fuzzy granulation–degranulation information processing. *Signal Processing*, 87(9):2061–2074, 2007.

[63] Peipei Li, Xindong Wu, Xuegang Hu, and Hao Wang. Learning Concept-drifting Data Streams with Random Ensemble Decision Trees. *Neurocomput.*, 166(C):68–83, 10 2015.

[64] Luis M. Candanedo and Véronique Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. *Energy and Buildings*, 112:28–39, 2016.