Enhancing the DISSFCM Algorithm for Data Stream Classification

Gabriella Casalino[1,2], Giovanna Castellano[1,2(✉)], Anna Maria Fanelli[1], and Corrado Mencar[1,2]

[1] Computer Science Department, University of Bari "Aldo Moro", Bari, Italy
{gabriella.casalino,giovanna.castellano,annamaria.fanelli,
corrado.mencar}@uniba.it
[2] INdAM Research Group GNCS, Rome, Italy

Abstract. Analyzing data streams has become a new challenge to meet the demands of real time analytics. Conventional mining techniques are proving inefficient to cope with challenges associated with data streams, including resources constraints like memory and running time along with single scan of the data. Most existing data stream classification methods require labeled samples that are more difficult and expensive to obtain than unlabeled ones. Semi-supervised learning algorithms can solve this problem by using unlabeled samples together with a few labeled ones to build classification models. Recently we proposed DISSFCM, an algorithm for data stream classification based on incremental semi-supervised fuzzy clustering. To cope with the evolution of data, DISSFCM adapts dynamically the number of clusters by splitting large-scale clusters. While splitting is effective in improving the quality of clusters, a repeated application without counter-balance may induce many small-scale clusters. To solve this problem, in this paper we enhance DISSFCM by introducing a procedure that merges small-scale clusters. Preliminary experimental results on a real-world benchmark dataset show the effectiveness of the method.

Keywords: Data stream classification · Semi-supervised fuzzy clustering · Incremental adaptive clustering

1 Introduction

Data stream mining is a recent methodology that deals with the analysis of large volumes of ordered sequences of data records. Data streams are a manifestation of Big Data, which are characterized by the four 'V' dimensions, namely Volume, Velocity, Variety and Veracity [1]. In particular, data stream mining assumes that the volume of the sequence of data is so large that records can be used few times (or just once) for the analysis. Data streams are produced by sensor networks, e-mails, online transactions, network traffic, weather forecasting, health monitoring, social networks, etc., just to cite the most common applications made available by current technology [2,3].

R. Fullér et al. (Eds.): WILF 2018, LNAI 11291, pp. 109–122, 2019.
https://doi.org/10.1007/978-3-030-12544-8_9

The requirement of using data records few times for extracting useful information involves the development of special-purpose data analysis methods, which should not require to store the whole stream of data in memory [4–6]. An approach to analyze data streams exploits an incremental generation of informational patterns, which represent a synthesized 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 [7,8].

Data stream mining is applied for different tasks, such as classification, clustering and frequent pattern mining. In this paper, we focus on classification of data records in a stream, which is deeply studied in literature [4,9–14]. Differently from most works in literature, which focus on supervised methods [15,16], we specialize into semi-supervised methods as we do not assume that all data records are completely labeled; on the other hand, we recognize that, in many contexts, labeled samples are difficult or expensive to obtain, 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. Semi-supervised learning in the context of data streams is relatively new when compared to supervised and unsupervised learning [17–20]. Despite several semi-supervised learning methods have been developed in the literature [21], only few of them have been applied to classify data streams [22,23]. Moreover, there are few attempts of using fuzzy clustering for data stream mining, despite fuzzy clustering could be particularly useful to capture the continuous changes in the clustering structure [24–28].

Based on the idea of combining the benefits of semi-supervised learning and fuzzy clustering, recently we developed an incremental semi-supervised clustering method for data stream classification [29], which applies the Semi-Supervised Fuzzy C-Means algorithm (SSFCM) [30] to data chunks. The method has been further 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 [31]. In essence, DISSFCM applies SSFCM to data chunks that correspond to a fixed-size collection of contiguous data records coming from a stream. Furthermore, SSFCM is modified in order to allow the incremental evolution of clusters; cluster quality is evaluated by reconstruction error so that, when the quality goes below a threshold, a splitting procedure is applied in order to divide a low-quality cluster into two higher-quality clusters. While splitting is effective in improving the quality of clusters, a repeated application without counter-balance may induce many small-scale clusters that do not represent meaningful patterns.

In this paper we enhance DISSFCM by introducing a merging procedure that merges clusters when there are too many clusters or there are clusters with too few data records. Clusters are merged when they are sufficiently close so as to not hamper the overall quality of the cluster structure.

The organization of the rest of the paper is as follows. Section 2 presents our method for data stream classification and its extension proposed in this work. In Sect. 3 the effectiveness of the extended method is evaluated on a benchmark dataset. The last section draws the conclusion and outlines future work.

2 Dynamic Incremental Semi-Supervised FCM

In this section we describe the complete DISSFCM (Dynamic Incremental Semi-Supervised FCM) algorithm [31], including a merging mechanism to avoid small-scale clusters and improve the structure of clusters.

DISSFCM assumes that data belonging to C different classes are continuously available during time and processed as chunks. Namely, a chunk of N_1 data is available at time t_1, a chunk of N_2 data is available at t_2 and so on[1]. We denote by X_t the data chunk available at time t. No assumption is made on the dimension of chunks that may vary from one chunk to another. One key feature of DISSFCM is the possibility to exploit partial supervision when available. Namely, when some pre-labeled data are available in a chunk, their labels can be used to drive the clustering process. The presence of pre-labeled data is not mandatory but it should be assured in the first chunk in order to initialize properly the cluster prototypes.

The core of DISSFCM is the SSFCM (Semi-Supervised FCM) algorithm [30] that is applied incrementally so as to enable continuous update of clusters based on new data chunks. At each time step SSFCM granulates data in the current chunk by producing a set of K clusters represented by K labeled prototypes $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_K\}$ representatives for the local data chunk they model. Each prototype $\mathbf{p}_k$ is a medoid, i.e. it is the datapoint closest to the center $\mathbf{c}_k$. Before starting the clustering process, K labeled data are randomly chosen to initialize the prototypes, so that each cluster prototype is associated to a class label ($K = C$). To take into account the evolution of the data during the incremental clustering process, the cluster prototypes discovered from the previous chunk are used as pre-labeled prototypes for the current chunk.

To better take into account the data evolution, DISSFCM is equipped with a splitting mechanism [31] that is applied to the current clusters in order to divide a low-quality cluster into two higher-quality clusters. The cluster quality is evaluated in terms of the *reconstruction error* [30]:

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

that measures the difference between the original data $\mathbf{x}_j$ and their "reconstructed" counterpart $\hat{\mathbf{x}}_j$ that is derived using the clustering outcome (prototypes and membership degrees) as follows:

[1] Any stream can be turned into a chunked stream by simply waiting for enough data points to arrive.

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

The splitting mechanism is activated when the reconstruction error on the current chunk exceeds a tolerance value ϵ 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.

The cluster having the highest value of the reconstruction error, i.e. the cluster with lowest reconstruction ability, is selected as candidate for splitting. The splitting is performed by means of the *conditional fuzzy clustering* [32] applied to the collection of data belonging to the cluster so as to create two novel prototypes. If we denote by S^* the set of data belonging to the cluster k^* selected for splitting and by $\mathbf{z}_1$ and $\mathbf{z}_2$ the two novel prototypes, the conditional clustering minimizes the following objective function:

$$J = \sum_{k=1}^{2} \sum_{j \in S^*} f_{jk}^m \|\mathbf{x}_j - \mathbf{z}_k\|^2 \tag{3}$$

under the constraint $f_{j1} + f_{j2} = u_{jk^*}$ where f_{jk} is the membership degree of $\mathbf{x}_j$ to the new cluster k. The objective function (3) is minimized by iteratively computing the membership values f_{jk} and the prototypes $\mathbf{z}_k$ according to:

$$f_{jk} = \frac{u_{jk^*}}{\sum_{c=1}^{2} \left(\frac{\|\mathbf{x}_j - \mathbf{z}_k\|}{\|\mathbf{x}_j - \mathbf{z}_c\|} \right)^{1/(m-1)}} \tag{4}$$

and

$$\mathbf{z}_k = \frac{\sum_{j \in S^*} f_{jk}^m \mathbf{x}_j}{\sum_{j \in S^*} f_{jk}^m}, \qquad k = 1, 2; \tag{5}$$

After conditional clustering, the prototype $\mathbf{p}_{k^*}$ is replaced by the two novel prototypes $\mathbf{z}_1$ and $\mathbf{z}_2$ that inherit the class label from $\mathbf{p}_{k^*}$. Then membership values u_{ik} are recomputed as in SSFCM. The splitting can be repeated until the reconstruction error drops below the previous value. A maximum pre-fixed number N_s of splittings is allowed for each chunk.

Since a repeated application of the splitting without counter-balance may induce many small-scale clusters that do not represent meaningful patterns, in this work we enhance DISSFCM by introducing a merging procedure that merges clusters when there are too many clusters or there are clusters with too few data records in a chunk. Clusters are merged when their prototypes are close so as to not hamper the overall quality of the cluster structure. The merging mechanism is activated when one of the following conditions is met:

1. the number of clusters exceeds a predefined threshold θ;
2. the number of data belonging to a cluster is below a predefined threshold λ.

In case 1. we select the nearest prototypes having the same class label as candidates for merging. We denote by $\mathbf{p}_s$ and $\mathbf{p}_l$ the nearest prototypes among all the current cluster prototypes sharing the same label. The new prototype $\mathbf{p}$ obtained by merging $\mathbf{p}_s$ and $\mathbf{p}_t$ is given by the following formula:

$$\mathbf{p} = \frac{\sum_{i=1}^{N}(u_{is} + u_{it})^m \mathbf{x}_i}{\sum_{i=1}^{N}(u_{is} + u_{it})^m} \tag{6}$$

where u_{is} and u_{it} are the membership values of $\mathbf{x}_i$ to cluster s and cluster t. In case 2. the prototype of the cluster with low number of data is merged with the closest cluster prototype, using Eq. (6). In each case, the merging reduces the number of clusters by one. The merging is repeated until there are no small clusters nor too many clusters. However, a maximum pre-fixed number N_m of merges is allowed for each chunk.

Algorithm 1. DISSFCM

Require: Data stream of chunks $X_1, X_2, ...$ containing few 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; K: number of prototypes

$t \leftarrow 1$
$K \leftarrow |P_0|$
$P \leftarrow P_0$
while $\exists$ nonempty chunk X_t **do**
  $X_t \leftarrow X_t \cup P$ */* *Add previous prototypes to the current chunk* */*
  $P, U \leftarrow SSFCM(X_t, K, P)$
  $n_s \leftarrow 0$ */* *Number of splits* */*
  $V_{max}^{(t)} \leftarrow$ reconstruction_error(X_t, P, U)
  while $(V_{max}^{(t)} - V_{max}^{(t-1)} > \epsilon)$ **and** $(n_s < MAX_s)$ **do**
    $P, U \leftarrow split(X_t, P, U)$
    $V_{max}^{(t)} \leftarrow$ reconstruction_error(X_t, P, U)
    $n_s \leftarrow n_s + 1$
  end while
  $n_m \leftarrow 0$ */* *Number of merges* */*
  while $(|P| > \theta$ **or** $\exists k : \sum_{j=1}^{N_t} u_{jk} < \lambda)$ **and** $(n_m < MAX_m)$ **do**
    $P, U \leftarrow merge(X_t, P, U)$
    $n_m \leftarrow n_m + 1$
  end while
  $K \leftarrow |P|$
  Classify data in X_t using labeled prototypes in P
  $t \leftarrow t + 1$
end while
return P

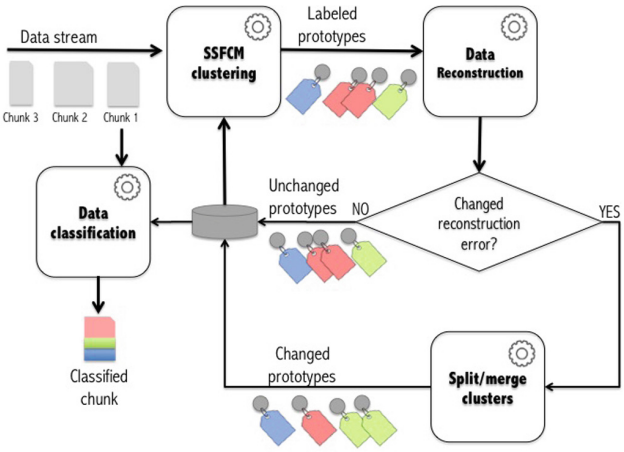


Fig. 1. Outline of DISSFCM. (Color figure online)

The overall scheme of DISSFCM enhanced with merging is shown in Fig. 1 and described in Algorithm 1. The algorithm requires the data stream as a sequence of 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 6) 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 20). 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 against all prototypes and assigned to the class label of the best-matching prototype. The matching mechanism is based on the standard Euclidean distance. 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.

3 Experimental Results

Numerical experiments were conducted to evaluate the effectiveness of the proposed algorithm in data stream classification. The Optical Recognition of Handwritten Digits dataset[2] has been considered. It contains $5,620$ images of handwritten digits belonging to 10 classes (namely, $0, 1, 2, \ldots, 9$). We used 10% of the samples as test set, and we partitioned the remaining 90% in a fixed number of chunks in order to simulate a data stream. The class distribution was preserved both in the chunks and in the test set.

[2] https://archive.ics.uci.edu/ml/datasets/optical+recognition+of+handwritten+digits.

Table 1. Parameters of the enhanced DISSFCM algorithm.

Parameter	values
MaxSplits	10
MaxMerge	2
%Labeling	75%
#Chunk	5 10 15 20
ϵ	25 50 100

The accuracy measure has been used to evaluate the classification results:

$$Acc = \frac{|\{\mathbf{x}_j | y_j = a_j\}|}{N_t}$$

where $\mathbf{x}_j$ is the j-th data point, y_j is the true class label and a_j is the predicted class label, N_t is the number of data points. After the t-th chunk has been processed, accuracy is computed not only on the test set, but also on the t-th chunk and on the previous processed chunks.

The purity external clustering measure has been used to evaluate the extent to which clusters contain a single class, after each chunk arrival. To compute purity, each cluster C_k is assigned to the class of a_k of its prototype, and then the accuracy of this assignment is measured by counting the number of correctly assigned data points and dividing by the cardinality of the cluster:

$$Pur(k) = \frac{|\{\mathbf{x}_j | \mathbf{x}_j \in C_k \cap y_j = a_k\}|}{|C_k|}$$

Then an average purity is computed on all the clusters.

We carried out some preliminary experiments by varying the parameters of the DISSFCM algorithm. Table 1 summarizes the experimental settings. A first evaluation was done by observing the reconstruction error. As an example, Fig. 2 shows the trend of the reconstruction error with $\#Chunk = 15$ and $\epsilon = 50$. Green dots correspond to the error after processing the current chunk, the blue dots indicate the error after a split and the yellow ones the error after a merge. Numbers on the dots indicate the number of prototypes (clusters). It can be seen that every time the reconstruction error exceeds the previous value plus the threshold ϵ, a split is activated and a new cluster is created (the number of clusters upon the blue dot is increased by one). When a cluster with a small number of samples occurs, a merge is activated and the number of clusters is reduced. It can be seen that most peaks occur when a new chunk arrives. This means that DISSFCM is still learning the correct model to fit the data and it improves the model as soon as a new chunk arrives (i.e. more training data). We observe that the split and merge steps help the model to fit the data. This could be better observed from Fig. 3, where the average purity values obtained

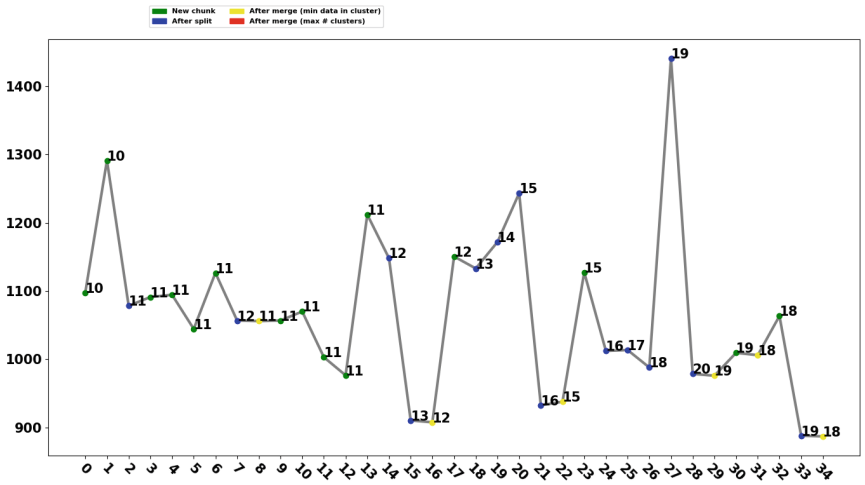


Fig. 2. Trend of the reconstruction error V_{max} with #Chunk = 15 and $\epsilon = 50$. (Color figure online)

Table 2. Number of cluster prototypes for each class at the end of the incremental process with #Chunk = 15, $\epsilon = 50$.

											Tot
Class	0	1	2	3	4	5	6	7	8	9	10
#Cluster	1	1	6	1	2	2	1	1	2	1	18

on single chunks during the learning process are reported. It can be seen that after processing the fifth chunk, the average value of purity decreases. When the sixth chunk arrives one split and one merge are applied (Fig. 2) rising the purity value. The same behavior could be observed after chunks 14-th and 15-th are processed. The processing of all the chunks ends with 18 cluster prototypes that are used to represent the 10 original classes. The number of cluster prototypes for each class is reported in Table 2.

Table 3 reports the accuracy computed on the chunks at each step t_i, during the incremental process with #Chunk = 15 and $\epsilon = 50$. Bold terms represent accuracy values on the current chunk. We observe that the model is properly adapted to the new arrived chunk. At each time step we also evaluated the classification accuracy of the current model on the previously seen chunks to verify if the model still fits the old data.

To assess the effectiveness of DISSFCM, we evaluated the classification accuracy of the final models for each configuration of parameters (#Chunk, ϵ). Results are summarized in Table 4. Both on the test and the training sets we can observe that the impact of the tolerance ϵ is higher when the number of chunks

Table 3. Accuracy obtained on single chunks during the incremental process, with #Chunk = 15, $\epsilon = 50$.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}
K	10	10	11	11	11	11	11	11	11	11	11	12	15	19	18
X_1	**0.84**	0.88	0.87	0.85	0.83	0.84	0.84	0.84	0.84	0.84	0.83	0.85	0.86	0.85	0.88
X_2	-	**0.86**	0.88	0.88	0.85	0.82	0.82	0.81	0.82	0.79	0.80	0.79	0.82	0.82	0.82
X_3	-	-	**0.82**	0.82	0.79	0.81	0.81	0.81	0.81	0.79	0.79	0.76	0.79	0.79	0.81
X_4	-	-	-	**0.82**	0.81	0.82	0.82	0.79	0.80	0.80	0.83	0.82	0.83	0.83	0.89
X_5	-	-	-	-	**0.81**	0.81	0.78	0.77	0.79	0.79	0.81	0.76	0.80	0.81	0.80
X_6	-	-	-	-	-	**0.84**	0.83	0.82	0.83	0.84	0.85	0.81	0.82	0.82	0.82
X_7	-	-	-	-	-	-	**0.86**	0.86	0.86	0.86	0.84	0.83	0.87	0.86	0.88
X_8	-	-	-	-	-	-	-	**0.85**	0.85	0.85	0.86	0.83	0.86	0.83	0.88
X_9	-	-	-	-	-	-	-	-	**0.87**	0.86	0.87	0.85	0.89	0.86	0.89
X_{10}	-	-	-	-	-	-	-	-	-	**0.82**	0.83	0.77	0.79	0.80	0.85
X_{11}	-	-	-	-	-	-	-	-	-	-	**0.84**	0.81	0.82	0.80	0.87
X_{12}	-	-	-	-	-	-	-	-	-	-	-	**0.81**	0.84	0.84	0.87
X_{13}	-	-	-	-	-	-	-	-	-	-	-	-	**0.83**	0.84	0.86
X_{14}	-	-	-	-	-	-	-	-	-	-	-	-	-	**0.87**	0.87
X_{15}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**0.91**

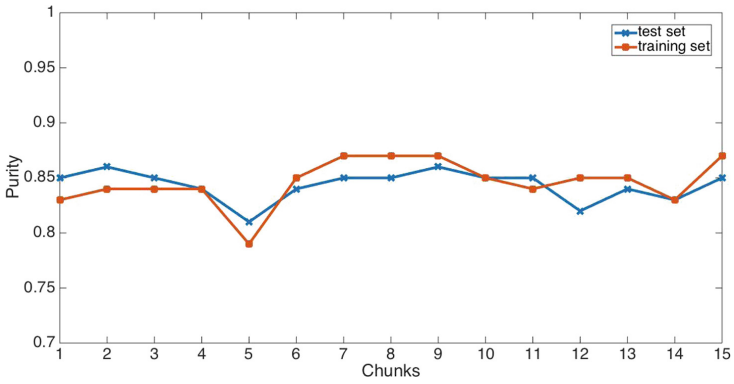


Fig. 3. Average purity obtained on single chunks during the incremental process, with #Chunk = 15, $\epsilon = 50$ on training and test sets. (Color figure online)

grows (i.e. the data samples in each sample decreases). Indeed the accuracy values with 5 and 10 chunks are stable when varying the values of ϵ. With 15 and 20 chunks the accuracy is more sensitive to the value of ϵ. This behavior can be better observed in the plots of Fig. 4 that show the trend of the accuracy on the test set during the processing of the chunks, varying the ϵ tolerance.

This is explained by observing that the higher the number of chunks, the less the number of samples in each chunk; therefore the algorithm has fewer

samples to learn from. Thus the number of the samples in each chunk affects the stability of the algorithm. With 5 and 10 chunks (high number of data) the algorithm keeps the same behavior as new chunks arrive (Fig. 4(a) and (b)). As the number of chunks increases (and hence the number of data in each chunk decreases), the algorithm is more unstable and needs more time to converge to an accurate model (Fig. 4(c) and (d)).

Table 4. Classification accuracy on the whole training set (a) and the test set (b), varying the number of chunks and the tolerance ϵ.

(a) Training set

	# chunks			
ϵ	5	10	15	20
25	0.84	0.85	0.88	0.93
50	0.84	0.85	0.91	0.83
100	0.84	0.85	0.85	0.85

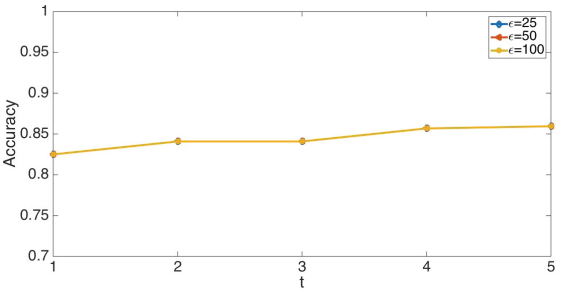
(b) Test set

	# chunks			
ϵ	5	10	15	20
25	0.86	0.84	0.85	0.89
50	0.86	0.85	0.87	0.78
100	0.86	0.84	0.81	0.79

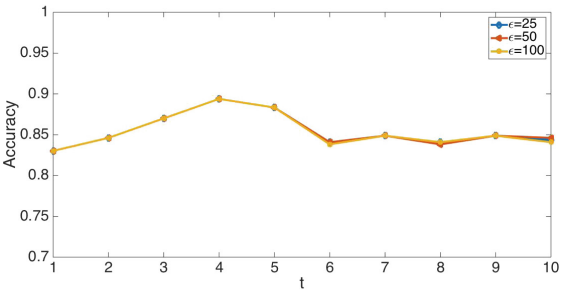
Finally, DISSFCM enhanced with merge was compared with its previous version [31] and with ILFM (Incremental Learning Fuzzy Measures) [33], which is a supervised incremental method based on Choquet integrals to classify data streams. Comparative results with #chunks = 15, $\epsilon = 50$ and labeling = 75% are plotted in Fig. 5.

It can be seen that the introduction of the merging mechanism in DISSFCM slightly deteriorates the classification results with respect to the previous version which only applies splits. However, it should be noted that the final classification model provided by the novel version of DISSFCM is very simple (18 clusters) in comparison to the final model obtained by the previous version of DISSFCM which was based on 70 clusters.

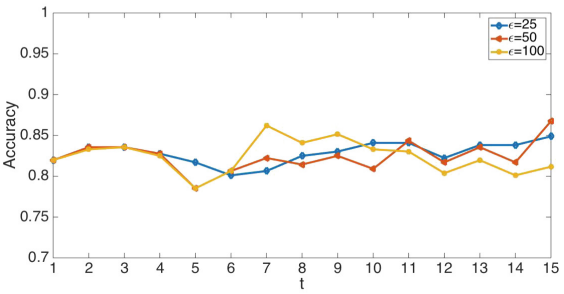
The models obtained by DISSFCM were also compared to the model built by ILFM. It can be seen that the classification accuracy of ILFM is slightly better. However it should be noted that ILFM is a supervised method, thus it requires completely labeled data, that are difficult to find in real applications. Conversely, DISSFCM works with partially labeled data. Moreover the model produced by ILFM is an ensemble of classifiers, hence it is far more complex than our model. On the overall, DISSFCM achieves a good balance between accuracy and complexity of the classification model, while taking into account the evolution of data.



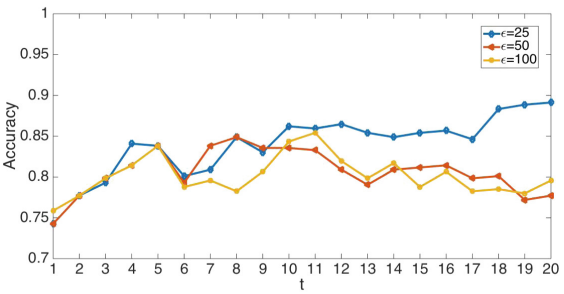
(a)



(b)



(c)



(d)

Fig. 4. Accuracy on the test set varying ϵ for #Chunk equal to 5 (a), 10 (b), 15 (c) and 20 (d). (Color figure online)

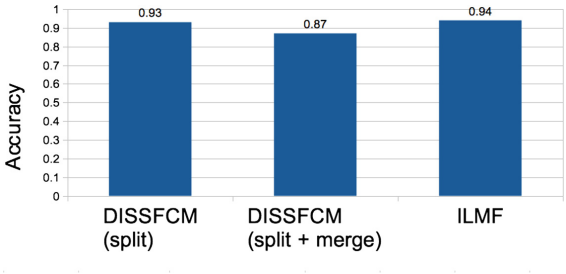


Fig. 5. Comparing the enhanced DISSFCM against its previous version (no merge), and ILMF. (Color figure online)

4 Conclusions

In this work we have described DISSFCM, a dynamic incremental semi-supervised version of the standard FCM clustering that is suitable for data stream classification. DISSFCM enables the structure of clusters to change dynamically: 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. Moreover, when few samples are grouped in a cluster, a merge step is activated for reducing the number of groups. Numerical preliminary analysis has shown that the split tolerance ϵ influences the accuracy results when the chunks dimension is small. Finally, it has been observed that the merge mechanism has a small negative impact on the accuracy of the model, when compared with DISSFCM without merge. However, in the face of such accuracy reduction we observe a significant simplification of the final model (18 cluster for DISSFCM with split and merge, against 70 for DISSFCM with split only). Similar considerations can be derived by comparing DISSFCM (with merge) and ILMF.

Further work is devoted to analyze the influence of the chunk 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. Moreover further research is going on along the direction of introducing a mechanism to detect outliers, concept drift and the emergence of new classes.

References

1. Eaton, C., Zikopoulos, P.: Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, 1st edn. McGraw-Hill Osborne Media, New York (2011)
2. Casalino, G., Castiello, C., Del Buono, N., Mencar, C.: Intelligent Twitter data analysis based on nonnegative matrix factorizations. In: Gervasi, O., et al. (eds.) ICCSA 2017. LNCS, vol. 10404, pp. 188–202. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62392-4_14

3. Casalino, G., Castiello, C., Del Buono, N., Mencar, C.: A framework for intelligent Twitter data analysis with nonnegative matrix factorization. Int. J. Web Inf. Syst. **14**(3), 334–356 (2018)
4. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2002, pp. 1–16. ACM, New York (2002)
5. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams: a review. SIGMOD Rec. **34**(2), 18–26 (2005)
6. Gama, J.: Knowledge Discovery from Data Streams, 1st edn. Chapman & Hall/CRC, Boca Raton (2010)
7. Chandak, M.B.: Role of big-data in classification and novel class detection in data streams. J. Big Data **3**(1), 5 (2016)
8. Chen, M., Mao, S., Liu, Y.: Big data: a survey. Mob. Netw. Appl. **19**(2), 171–209 (2014)
9. Ferranti, A., Marcelloni, F., Segatori, A., Antonelli, M., Ducange, P.: A distributed approach to multi-objective evolutionary generation of fuzzy rule-based classifiers from big data. Inf. Sci. **415**, 319–340 (2017)
10. Ducange, P., Pecori, R., Mezzina, P.: A glimpse on big data analytics in the framework of marketing strategies. Soft Comput. **22**(1), 325–342 (2018)
11. Lughofer, E., Pratama, M.: Online active learning in data stream regression using uncertainty sampling based on evolving generalized fuzzy models. IEEE Trans. Fuzzy Syst. **26**(1), 292–309 (2018)
12. Hyde, R., Angelov, P., MacKenzie, A.R.: Fully online clustering of evolving data streams into arbitrarily shaped clusters. Inf. Sci. **382–383**, 96–114 (2017)
13. Lughofer, E.: A dynamic split-and-merge approach for evolving cluster models. Evol. Syst. **3**(3), 135–151 (2012)
14. Olorunnimbe, M.K., Viktor, H.L., Paquet, E.: Dynamic adaptation of online ensembles for drifting data streams. J. Intell. Inf. Syst. **50**(2), 291–313 (2018)
15. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2000, pp. 71–80. ACM (2000)
16. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2001, pp. 97–106. ACM (2001)
17. Nguyen, H.-L., Woon, Y.-K., Ng, W.-K.: A survey on data stream clustering and classification. Knowl. Inf. Syst. **45**(3), 535–569 (2015)
18. Mousavi, M., Bakar, A.A., Vakilian, M.: Data stream clustering algorithms: a review. Int. J. Adv. Soft Comput. Appl. **7**(3), 13 (2015)
19. Toshniwal, D.: Clustering techniques for streaming data - a survey. In: 2013 3rd IEEE International Advance Computing Conference, IACC, pp. 951–956, February 2013
20. Ghesmoune, M., Lebbah, M., Azzag, H.: Micro-batching growing neural gas for clustering data streams using spark streaming. Proc. Comput. Sci. **53**, 158–166 (2015). INNS Conference on Big Data 2015 Program, San Francisco, CA, USA, 8–10 August 2015
21. Zhu, X.: Semi-supervised learning literature survey. Technical report 1530, Computer Sciences. University of Wisconsin-Madison (2005)
22. Blum, A., Mitchell, T.M.: Combining labeled and unlabeled data with co-training. In: Bartlett, P.L., Mansour, Y. (eds.) COLT, pp. 92–100. ACM (1998)

23. Zhou, Z.-H., Li, M.: Tri-training: exploiting unlabeled data using three classifiers. IEEE Trans. Knowl. Data Eng. **17**(11), 1529–1541 (2005)
24. Beringer, J., Hüllermeier, E.: Fuzzy clustering of parallel data streams. In: Advances in Fuzzy Clustering and Its Application, pp. 333–352 (2007)
25. Abdullatif, A., Masulli, F., Rovetta, S.: Clustering of nonstationary data streams: a survey of fuzzy partitional methods. Wiley Interdisc. Rev.: Data Min. Knowl. Discov. **8**(4), e1258 (2018)
26. Mostafavi, S., Amiri, A.: Extending fuzzy C-means to clustering data streams. In: 20th Iranian Conference on Electrical Engineering, ICEE 2012, pp. 726–729, May 2012
27. Upadhyay, D., Jain, S., Jain, A.: A fuzzy clustering algorithm for high dimensional streaming data. J. Inf. Eng. Appl. **3**(10), 1–9 (2013)
28. Geweniger, T., Fischer, L., Kaden, M., Lange, M., Villmann, T.: Clustering by fuzzy neural gas and evaluation of fuzzy clusters. Comput. Intell. Neurosci. **2013**, 9 (2013)
29. Castellano, G., Fanelli, A.M.: Classification of data streams by incremental semi-supervised fuzzy clustering. In: Petrosino, A., Loia, V., Pedrycz, W. (eds.) WILF 2016. LNCS, vol. 10147, pp. 185–194. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52962-2_16
30. Pedrycz, W.: Algorithms of fuzzy clustering with partial supervision. Pattern Recogn. Lett. **3**(1), 13–20 (1985)
31. Casalino, C., Castellano, G., Mencar, C.: Incremental adaptive semi-supervised fuzzy clustering for data stream classification. In: Proceedings of the 2018 IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS 2018, Rhodes, Greece, 25–27 May 2018, pp. 1–7 (2018)
32. Li, P., Wu, X., Hu, X., Wang, H.: Learning concept-drifting data streams with random ensemble decision trees. Neurocomputing **166**(C), 68–83 (2015)
33. Xuefei, L., Huimin, F., Hongbo, S.: Incremental learning fuzzy measures with Choquet integrals in fusion system. J. Chem. Pharm. Res. **6**, 102–112 (2014)