EPiC Series in Computing

Volume 81, 2022, Pages 65–76

Proceedings of 11th International Congress on Advanced Applied Informatics

A Possibilistic Fuzzy Approach for Novelty Detection with Automatically Adjusted Number of Clusters*

Shakhnaz Akhmedova, Vladimir Stanovov, Eugene Semenkin and Sophia Vishnevskaya

Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russian Federation

shahnaz@inbox.ru, vladimirstanovov@yandex.ru, eugenesemenkin@yandex.ru, vichnevskaya@mail.sibsau.ru

Abstract

This paper introduces a new modification of the Possibilistic Fuzzy multiclass Novelty Detector for data streams (PFuzzND). Mentioned modification is based on the implementation of the automated adjustment of the number of clusters for each class (determined beforehand or during the novelty detection procedure) to improve algorithm's ability to divide objects into small groups. As result, the proposed approach generates models with flexible class boundaries, which are capable to identify new classes or extensions of the ones that are already known as well as the outliers. Proposed possibilistic fuzzy algorithm for novelty detection was used to solve various benchmark problems with synthetically generated datasets. In order to show the workability and efficiency of the introduced modification its results were also compared with the results obtained by the original PFuzzND algorithm. Thus, it was established that the PFuzzND technique with automatically adjusted number of clusters allows achieving better results in regards to the accuracy, the Macro *F*-score metric and the unknown rate measure. Comparison to the original algorithm showed that the proposed modification outperforms it but is sensitive to the parameter settings, which can be also said about the PFuzzND method. Therefore, the MPFuzzND approach can be used instead of the original PFuzzND algorithm for other classification problems.

1 Introduction

Data stream mining is the process of the knowledge discovery in large amounts of continuously generated data (Bifet A., 2018). It usually involves appearance of new concepts, their evolution and

* The reported study was funded by RFBR and FWF according to the research project № 21-51-14003\21.

even disappearance, thus, novelty detection (Pimentel, 2014) becomes one of the most important tasks for data stream learning. Novelty detection is the process of identifying new or unknown situations not experienced before (or new objects not seen before). Novelty detection tasks can be considered as classification problems with aim to distinguish new concepts, which can be extensions of already existing and known classes or something completely new and, therefore, these concepts may form new groups (classes). Besides, novelty detection also allows determining noise in data streams.

Novelty detection is related to the outlier detection (Chandola, 2009): both mechanisms are interested in detecting abnormal or unusual observations in data streams also called anomalies. Outliers (or anomalies) in data are extreme values that deviate from other observations in a given dataset, they may indicate noise, errors in measurement or a novelty. Thus, novelty detection is a semi-supervised learning approach used for data streams where these outliers may form new patterns in data.

There are two possible outcomes of the novelty detection algorithms application: concept evolution and concept drift. Concept evolution in data streams refers to the appearance of novel classes while streams evolve (Gama, 2010). Traditional classifiers are unable to detect novel classes, which leads to misclassifying all instances representing the novel concepts. Novel concepts should be determined as soon as they appear in data streams so that they could also be assimilated into the underlying concept for further detection of recurring novel class instances. The term concept drift refers to changes in the conditional distribution of the output or in other words target variable given the input features, while the distribution of the input features may stay unchanged (Cejnek, 2018). Concept drift indicates that there are extensions of the previously established classes that should be considered for new incoming data instances.

Nowadays, novelty detection problems receive increasing interest from researchers due to their applicability and significance in real-world practices, thus, more works have been proposed to solve them in the last years, for example (Ouafae, 2020) or (Silva, 2018). Most of the novelty detection approaches have two phases, namely offline and online phases (Faria, 2018): that makes it possible to use already known information about the part of the instances to find new patterns among the new data. Additionally, in order to generate more flexible and efficient models researches started frequently implementing fuzzy logic while developing the novelty detection algorithms (Škrjanc, 2019).

In this study, a new modification of the Possibilistic Fuzzy multiclass Novelty Detector for data streams (PFuzzND) (Silva, 2020) is introduced. The original PFuzzND approach is based on the MINAS algorithm (de Faria, 2016): it has both offline and online phases and uses the fuzzy set theory. It is applicable to the multiclass data streams and divides each class into the pre-determined number of clusters. Thus, in the proposed modification (denoted as MPFuzzND hereinafter) the number of clusters for all classes is automatically adjusted on each step of the algorithm's work.

In order to demonstrate the advantages of the proposed MPFuzzND method it was evaluated on a set of benchmark problems with synthetically generated datasets (Data stream repository). Obtained results were compared with results of the original PFuzzND algorithm. It should be noted that the experimental results were evaluated by the accuracy, the Macro *F*-score metric and the unknown rate measure (de Faria, 2016). Finally, it was established that the proposed method outperformed its alternative PFuzzND, thus, its usefulness was proved. However, it is dependable on its parameters that should be accurately chosen for each problem. Therefore, it is shown that the main advantage of the MPFuzzND is that it allows finding better description of the data streams in terms of accuracy, Macro *F*-score and unknown rate measure thanks to clusters merging procedure.

Therefore, in this paper, firstly, the original possibilistic fuzzy algorithm PFuzzND is described, and then the description of its modification is presented. In the next section the experimental results, namely the experimental setup and numerical results obtained by the proposed approach as well as results obtained by the original PFuzzND algorithm are discussed and demonstrated. Finally, some conclusions are given in the last section.

2 Proposed Approach

In this section a brief description of the original possibilistic fuzzy approach PFuzzND, proposed in (Silva, 2020), is given. It is followed by the description of the developed modification MPFuzzND based on the automated selection of the number of clusters for each class (known or found during algorithm's work). It should be noted that the original PFuzzND algorithm as well as its modification MPFuzzND are both using the well-known fuzzy clustering method, namely the Possibilistic Fuzzy C-Means (PFCM) method (Pal, 2005). In its turn, the PFCM method is based on ideas introduced for the fuzzy clustering method FCM (Bezdek, 1981) and the possibilistic clustering method PCM (Krishnapuram, 1993) and, therefore, both the membership values and typicalities are used to generate models, which are more successful during the novelty detection procedure and are less sensitive to parameter choices.

2.1 Baseline Possibilistic Novelty Detection Approach

As was mentioned above the PFuzzND approach has two phases, offline and online, whereas the online phase also can be divided into two steps, namely the main step and the novelty detection step. Let us assume that there are N instances $x_1, x_2, \ldots, x_N$ in a given data stream, and each instance is presented as a vector in the D-dimensional space.

The offline phase is performed one time at the beginning of the algorithm's work, at this phase the timestamp is $t = 1$. During that phase firstly the portion *p_offline* of instances from the data stream are randomly chosen with the only condition, to be more specific, these instances should be already labelled. Let us say that there are *NC* known classes during the offline phase. The end-user has to choose into how many clusters each class will be divided (that number will be denoted as *k_class* hereinafter). Then for each class the PFCM method is applied and each known class is described by two $k\_class \times N$ matrices by the end of the offline phase

1. the membership matrix U (each column of matrix U consists of a given instance's membership values to all *k_class* clusters);
2. the typicality matrix T (each column of matrix T consists of a given instance's typicality values for all *k_class* clusters).

So the decision model is defined as the set of $NC \cdot k\_class$ clusters found for all *NC* different known classes at the timestamp $t = 1$. It should be noted that the PFCM algorithm has its own parameters, which should be accurately chosen (Pal, 2005).

An empty external set, called short memory, is created before the main step of the online phase starts. Examples labelled as unknown are stored in short memory for a time-period *ts*, after that time limit those instances are removed from the mentioned set. Moreover, there are four additional parameters:

1. the minimum number of instances (denotes as *SM*) in the short memory to start the novelty detection procedure;
2. the initial threshold θ_{init} for classifying and processing instances from the data stream;
3. two adaptation thresholds θ_{class} and θ_{adapt} used during the classification step.

Thus, during the online step firstly for each new instance x_j from the data stream its membership values and typicalities related to all existing (known) at moment clusters are calculated. Typicality values are used to determine whether the instance x_j will be labelled by one of the existing classes or will be marked as unknown. For this purpose the highest typicality value of the j-th instance and the corresponding existing class C_i ($i = 1, \ldots, NC$) are determined. Then this value is compared with the

difference between the mean of maximum typicalities of all instances considered before x_j (and belonging to the class C_i) and the adaptation threshold θ_{adapt}. If the highest typicality is greater than the mentioned difference then the instance x_j will be labelled as belonging to the class C_i (Silva, 2020). It should be noted that the highest typicality of the first instance processed after the offline phase is compared to the initial threshold θ_{init}.

If the instance x_j is labelled then the mean of maximum typicalities as well as the clusters of the class C_i are updated (the latter is done by using the PFCM method). Otherwise, the highest typicality of that instance is compared to the difference between the mean of maximum typicalities of all instances considered before x_j (and belonging to the class C_i) and the adaptation threshold θ_{class}. If the highest typicality is greater than the last difference then the instance x_j will be labelled as belonging to the class C_i and new cluster will be created for it with x_j as its centroid (Silva, 2020).

If neither of conditions is met then the instance x_j will be marked as unknown and stored in the short memory until the novelty detection step will be executed. The latter happens if the number of instances marked as unknown reached *SM*. In this case, firstly, the PFCM approach is applied to all the instances in the short memory and the pre-determined number *k_short* of clusters is generated. After that for each generated cluster its fuzzy silhouette (Campello, 2006) is calculated and if the obtained value is greater than 0 and the considered cluster is not empty then this cluster is evaluated as the valid one.

All validated clusters of the short memory represent new patterns. Next step of the novelty detection procedure consists in calculating how similar these validated clusters are to the ones already existing in the model, which is done by using the fuzzy clustering similarity metric introduced in (Xiong, 2004). Finally, if the value of the mentioned metric between one of the known clusters and the examined valid cluster of short memory is greater than φ (which is another parameter of the PFuzzND approach) then all instances from the examined cluster are labelled the same way as instances of the considered known cluster. Consequently, clusters of the corresponding class are updated by the PFCM algorithm (Silva, 2020). Otherwise new class C_{NC+1} is created, instances from the examined cluster are labelled as belonging to the class C_{NC+1}, *NC* increases by one.

If one of the short memory clusters is not validated then it is discarded and its instances remain in the short memory until the model executes the novelty detection procedure again or decides to remove them at all, which can happen if these instances were in the short memory for *ts* iterations.

2.2 Modified Possibilistic Fuzzy Clustering Method

The modification MPFuzzND of the original approach consists in the automated selection of the number of clusters for each class known beforehand or determined during the novelty detection procedure. To be more specific the original PFuzzND algorithm allows increasing the number of clusters belonging to each class, but not the other way around. Experiments showed that in some cases the instances belonging to the same class might be divided into the large number of clusters and that can lead to bad classification results. Thus, in this study it is proposed to merge clusters belonging to the same class if they are similar to each other, which allows decreasing their number.

The similarity between clusters is defined according to the metric proposed in (Xiong, 2004). It can be described in the following way: firstly, the dispersions of two considered clusters are calculated, then the dissimilarity between these clusters is determined and, finally, the sum of dispersions is divided by the dissimilarity value (Xiong, 2004). Cluster's dispersion is the weighted sum of distances between instances belonging to this cluster and its center averaged by the number of considered instances. It should be noted that the weight coefficients here are the typicality values, even though in (Xiong, 2004) the membership values were used. The dissimilarity between two clusters is the Euclidian distance between their centers.

Below the pseudo-code of the proposed modification is demonstrated.

```
If CN_k > 2
  For i = 1 : CN_k
    For j = 1 : CN_k
      If i > j
        Calculate the similarity value FS_ij between the i-th and the
        j-th clusters
      End If
    End For
  End For
  Find the maxFS = max{FS_ij}, i = 1,…,CN_k and j = 1,…,CN_k
  Determine centres of the most similar clusters cntr_1 and cntr_2
  corresponding to the maxFS value
  If maxFS > θ_merge
    Create new cluster by merging two the most similar clusters
    Calculate the centre of the new cluster:
    cntr = 0.5 · (cntr_1 + cntr_2)
    Decrease the number of clusters:
    CN_k = CN_k − 1
    Execute the PFCM algorithm with new CN_k clusters to update them
  End If
End If
```

In the presented pseudo-code CN_k is the number of clusters belonging to the k-th class, θ_{merge} is the new parameter introduced for this modification, and it is called the merging threshold. For clusters of the k-th class the symmetric matrix FS of similarity values is calculated and two the most similar clusters are determined, their similarity value is denoted as *maxFS* here. If the *maxFS* value is greater than the mentioned threshold θ_{merge} then these two clusters should be merged and new cluster should be created instead of them with center calculated as it is shown in the pseudo-code. After the PFCM algorithm is used to update the clusters of the considered class (centers of existing clusters are taken as the initial points for them).

It should be noted that the proposed merging operator is applied on each iteration of the online phase before the novelty detection procedure is executed. To be more specific, it is applied each time when one of two conditions, described in the previous subsection, is met, i.e. the new considered instance is labelled as belonging to the already existing class (it could have been known on the offline phase or found on previous iterations of the online phase). Additionally, there is no need to merge clusters if their number is less than three, thus, it is used only if $CN_k > 2$.

3 Experimental Results

In order to verify the advantages of the proposed modification of the PFuzzND approach, its results were compared against the results obtained by the original PFuzzND algorithm. This comparison was not done against the well-known MINAS algorithm due to the fact that it was outperformed by the PFuzzND approach according to (Silva, 2020). To conduct the experiments 10 synthetic datasets from (Data stream repository) were used. Each experiment was executed 10 times due to the randomness of the clustering algorithm used as during offline so during the online steps.

Next, the datasets and the evaluation metrics used in the experiments as well as obtained results will be explained.

3.1 Experimental Setup and Benchmark Problems

The execution of the methods was done by using 10 synthetic datasets named as DS1, …, DS10 (Data stream repository). Table 1 presents the details of each dataset:

1. column *Instances* indicates the total number of examples for each dataset respectively;
2. column *Attributes* indicates the number of features for each dataset respectively;
3. column *Classes (Offline)* indicates the number of known at the offline step classes;
4. column *Classes* indicates the minimum total number of classes for each dataset (the difference between this number and the corresponding value from the column *Classes (Offline)* is the minimum number of novel classes that the algorithms are expected to detect).

Dataset	Instances	Attributes	Classes (Offline)	Classes
DS1	1000	2	3	4
DS2	5500	2	2	3
DS3	10000	2	3	3
DS4	11000	2	2	3
DS5	20000	2	3	3
DS6	10000	2	4	4
DS7	10000	2	3	3
DS8	10000	3	3	3
DS9	10000	2	4	4
DS10	40000	3	3	4

Table 1: Datasets used in experiments

Both the original PFuzzND approach and its modification MPFuzzND use possibilistic fuzzy clustering algorithm PFCM during offline and online phases. Parameters of the PFCM algorithm were the same for all conducted experiments, and they are listed in Table 2. The following notations are used in this table: *expo* and *nc* are constants responsible for the membership and the typicality values, *a* and *b* are constants that show the influence of the membership and the typicality values during the decision making step, *MaxIter* is the maximum number of iterations for PFCM to generate a given number of clusters, and *MinImp* is the minimum difference between the objective function values calculated on two consequential iterations of the PFCM algorithm (if this difference was less than *MinImp* then calculations stopped).

Parameter	Value
Expo	2
MaxIter	1000
MinImp	0.0005
a	1
b	4
nc	3

Table 2: Parameters of the PFCM algorithm

It should be noted that the PFCM clustering method uses a specific parameters γ_i for each *i*-th cluster. These parameters influence the typicality values, which in their turn are necessary to determine clusters' centers. The γ_i parameters were introduced for the possibilistic clustering method PCM (Krishnapuram, 1993), in that work they were user-defined constants, while in (Krishnapuram,

1996) authors proposed to calculate these values on each iteration of the clustering algorithm. Thus, in this study experiments were conducted as for $\gamma_i = 1$ for all clusters so for the iteratively calculated γ_i.

Conducted experiments were related to the comparison between the original PFuzzND approach and its modification MPFuzzND. Thus, to make it fair the same basic parameters were used during the offline and the online phases (including the novelty detection procedure) of both algorithms. These parameters are listed in Table 3 and the following additional notations are used:

1. offline phase – m_{off} is the fuzzification parameter during the offline phase, *k_class* is the maximum possible number of clusters generated for each known class at that phase, *p_offline* is the portion of data instances used during the offline step (for these instances class labels are already known) to generate the first model;
2. novelty detection procedure – *SM* is the minimum amount of unknown examples in the short memory required for the novelty detection procedure to be executed, *ts* is a time limit corresponding to the removal of older unknown examples in short memory, m_{nd} is the fuzzification parameter for the novelty detection procedure, *k_short* is the maximum possible number of clusters generated for instances stored in the short memory, and φ is the parameter corresponding for the decision if the considered cluster is responsible for the concept drift or for the concept novelty.

Offline phase		Online phase		Novelty detection	
Parameter	Value	Parameter	Value	Parameter	Value
m_{off}	2	θ_{init}	0.3	*SM*	40
k_class	3	θ_{class}	0.2	*ts*	200
p_offline	0.1	θ_{adapt}	0.15	m_{nd}	2
		θ_{merge}	*	*k_short*	4
				φ	1

Table 3: Basic parameters of the PFuzzND algorithm and its modification MPFuzzND

It should be noted that for the original PFuzzND algorithm parameters *k_class* and *k_short* are fixed during the offline phase and the novelty detection procedure, namely exactly these numbers of clusters are generated for each known class and short memory respectively. Thus, initially the number of clusters is the same for all known classes but later it can be different for classes due to the procedures executed during the online phase. Additionally, in this study the maximum number of clusters created for one class is limited, namely this number cannot be greater than 8 or less than 2.

As for the modification MPFuzzND, the main problem was to determine which value to use for the θ_{merge} threshold. This value varied from 0.1 to 1.4 in conducted experiments. This range was chosen due to the fact that the merging idea is based on the study presented in (Xiong, 2004) and, thus, the similar values were used there.

Experiments were evaluated using the incremental confusion-matrix, which evolves as soon as a new data point is classified (de Faria, 2015). This matrix is composed by rows and columns that represent known classes, concept drift of known classes and new classes (concept evolution), and finally unlabelled data instances (the ones removed from the short memory). So, for each dataset three values were calculated over 10 program runs: the accuracy, the Macro *F*-score metric and the unknown rate measure *UnkR* (de Faria, 2016). The accuracy was calculated only for the labelled data instances (new classes weren't considered), while the Macro *F*-score values were calculated considering not only the examples classified with already existing class labels but also with new class labels, determined by algorithm during the novelty detection process.

3.2 Numerical Results

The aim of the conducted experiments was to compare the MPFuzzND modification with the original PFuzzND algorithm. Thus, two configurations of both algorithms were executed on all 10 datasets 10 times: with γ_i set to 1 for each *i*-th cluster and with iteratively calculated γ_i for each *i*-th cluster.

Dataset		*Acc*	*MF*	*UnkR*	Dataset		*Acc*	*MF*	*UnkR*
	MPFuzzND $\theta_{merge} = 0.2$	99.89	100	4.50		*MPFuzzND* $\theta_{merge} = 0.2$	96.90	96.40	15.69
	MPFuzzND $\theta_{merge} = 0.6$	99.87	100	3.00		*MPFuzzND* $\theta_{merge} = 0.6$	98.51	98.01	50.42
DS1	*MPFuzzND* $\theta_{merge} = 0.8$	99.89	100	3.00	DS6	*MPFuzzND* $\theta_{merge} = 0.8$	97.74	97.09	48.14
	MPFuzzND $\theta_{merge} = 1.10$	99.89	100	2.58		*MPFuzzND* $\theta_{merge} = 1.10$	**98.89**	**98.51**	**48.30**
	PFuzzND	99.89	100	1.58		*PFuzzND*	98.81	98.39	49.76
	MPFuzzND $\theta_{merge} = 0.2$	**86.19**	**94.34**	**0.52**		*MPFuzzND* $\theta_{merge} = 0.2$	97.54	97.52	1.37
	MPFuzzND $\theta_{merge} = 0.6$	**85.14**	**93.06**	**0.00**		*MPFuzzND* $\theta_{merge} = 0.6$	97.28	98.26	0.31
DS2	*MPFuzzND* $\theta_{merge} = 0.8$	**85.86**	**93.91**	**0.00**	DS7	*MPFuzzND* $\theta_{merge} = 0.8$	**97.57**	**97.55**	**0.20**
	MPFuzzND $\theta_{merge} = 1.10$	81.43	88.19	0.00		*MPFuzzND* $\theta_{merge} = 1.10$	**97.56**	**97.55**	**0.28**
	PFuzzND	84.11	91.89	0.00		*PFuzzND*	97.54	97.52	0.27
	MPFuzzND $\theta_{merge} = 0.2$	**88.80**	**88.20**	**3.10**		*MPFuzzND* $\theta_{merge} = 0.2$	**98.41**	**98.40**	**1.69**
	MPFuzzND $\theta_{merge} = 0.6$	**88.87**	**88.48**	**1.73**		*MPFuzzND* $\theta_{merge} = 0.6$	**98.44**	**98.42**	**1.62**
DS3	*MPFuzzND* $\theta_{merge} = 0.8$	88.52	88.22	1.63	DS8	*MPFuzzND* $\theta_{merge} = 0.8$	**90.73**	**90.54**	**0.51**
	MPFuzzND $\theta_{merge} = 1.10$	**88.54**	**88.26**	**1.67**		*MPFuzzND* $\theta_{merge} = 1.10$	**92.52**	**92.41**	**0.51**
	PFuzzND	88.52	88.25	1.62		*PFuzzND*	90.51	90.09	0.60
	MPFuzzND $\theta_{merge} = 0.2$	**87.24**	**95.89**	**0.00**		*MPFuzzND* $\theta_{merge} = 0.2$	**95.07**	**95.05**	**0.43**
	MPFuzzND $\theta_{merge} = 0.6$	**82.16**	**89.16**	**0.00**		*MPFuzzND* $\theta_{merge} = 0.6$	91.98	91.91	0.36
DS4	*MPFuzzND* $\theta_{merge} = 0.8$	**75.29**	**79.39**	**0.00**	*DS9*	*MPFuzzND* $\theta_{merge} = 0.8$	93.50	93.41	0.16
	MPFuzzND $\theta_{merge} = 1.10$	**78.57**	**84.11**	**0.00**		*MPFuzzND* $\theta_{merge} = 1.10$	**94.19**	**94.16**	**0.11**
	PFuzzND	67.85	69.19	0.00		*PFuzzND*	93.60	93.60	0.09
	MPFuzzND $\theta_{merge} = 0.2$	**88.55**	**87.94**	**2.98**		*MPFuzzND* $\theta_{merge} = 0.2$	58.65	64.62	0.03
	MPFuzzND $\theta_{merge} = 0.6$	86.45	86.23	1.90		*MPFuzzND* $\theta_{merge} = 0.6$	**65.58**	**72.00**	**0.01**
DS5	*MPFuzzND* $\theta_{merge} = 0.8$	86.18	85.96	2.00	*DS10*	*MPFuzzND* $\theta_{merge} = 0.8$	62.59	68.14	0.03
	MPFuzzND $\theta_{merge} = 1.10$	86.35	86.13	2.08		*MPFuzzND* $\theta_{merge} = 1.10$	**65.87**	**72.51**	**0.01**
	PFuzzND	86.50	86.30	2.06		*PFuzzND*	65.43	71.93	0.02

Table 4: Results obtained by the PFuzzND algorithm and proposed modification MPFuzzND, $\gamma_i = 1$

Results obtained for both configurations of compared algorithms on all datasets are presented in Table 4 and Table 5. In these tables results averaged by the number of program runs are shown, and *Acc* means accuracy in % while *MF* denotes the Macro *F*-score metric values also given in %. Additionally, the *UnkR* values here should be multiplied by 10^{-3}.

Dataset		*Acc*	*MF*	*UnkR*	Dataset		*Acc*	*MF*	*UnkR*
	MPFuzzND $\theta_{merge} = 0.3$	**88.50**	**94.62**	**109.75**		*MPFuzzND* $\theta_{merge} = 0.3$	**98.84**	**98.45**	**51.68**
	MPFuzzND $\theta_{merge} = 0.7$	**93.36**	**97.65**	**114.17**		*MPFuzzND* $\theta_{merge} = 0.7$	93.34	91.40	46.58
DS1	*MPFuzzND* $\theta_{merge} = 0.8$	**85.23**	**89.66**	**103.33**	DS6	*MPFuzzND* $\theta_{merge} = 0.8$	98.55	98.08	48.87
	MPFuzzND $\theta_{merge} = 1.10$	**93.15**	**96.70**	**129.67**		*MPFuzzND* $\theta_{merge} = 1.10$	**98.81**	**98.39**	**53.34**
	PFuzzND	81.54	84.16	85.17		*PFuzzND*	98.80	98.37	52.36
	MPFuzzND $\theta_{merge} = 0.3$	**86.88**	**92.91**	**61.28**		*MPFuzzND* $\theta_{merge} = 0.3$	97.95	97.94	70.16
	MPFuzzND $\theta_{merge} = 0.7$	**86.78**	**92.66**	**59.59**		*MPFuzzND* $\theta_{merge} = 0.7$	98.10	98.09	71.02
DS2	*MPFuzzND* $\theta_{merge} = 0.8$	**86.30**	**92.76**	**64.91**	DS7	*MPFuzzND* $\theta_{merge} = 0.8$	98.09	98.08	69.14
	MPFuzzND $\theta_{merge} = 1.10$	**87.54**	**93.31**	**64.45**		*MPFuzzND* $\theta_{merge} = 1.10$	**98.16**	**98.16**	**71.20**
	PFuzzND	82.72	90.48	55.56		*PFuzzND*	98.11	98.10	72.88
	MPFuzzND $\theta_{merge} = 0.3$	67.54	67.38	57.51		*MPFuzzND* $\theta_{merge} = 0.3$	99.86	99.87	42.72
	MPFuzzND $\theta_{merge} = 0.7$	**69.72**	**70.07**	**56.14**		*MPFuzzND* $\theta_{merge} = 0.7$	99.87	99.87	46.36
DS3	*MPFuzzND* $\theta_{merge} = 0.8$	**70.32**	**70.68**	**57.96**	DS8	*MPFuzzND* $\theta_{merge} = 0.8$	99.87	99.88	47.83
	MPFuzzND $\theta_{merge} = 1.10$	**72.32**	**72.94**	**59.90**		*MPFuzzND* $\theta_{merge} = 1.10$	99.88	99.88	47.44
	PFuzzND	69.04	69.30	57.02		*PFuzzND*	99.89	99.89	47.61
	MPFuzzND $\theta_{merge} = 0.3$	78.91	86.27	46.22		*MPFuzzND* $\theta_{merge} = 0.3$	94.83	94.83	83.79
	MPFuzzND $\theta_{merge} = 0.7$	**89.19**	**94.10**	**54.00**		*MPFuzzND* $\theta_{merge} = 0.7$	**95.03**	**95.03**	**82.21**
DS4	*MPFuzzND* $\theta_{merge} = 0.8$	74.74	84.10	43.37	*DS9*	*MPFuzzND* $\theta_{merge} = 0.8$	**95.03**	**95.03**	**83.08**
	MPFuzzND $\theta_{merge} = 1.10$	85.86	92.24	47.13		*MPFuzzND* $\theta_{merge} = 1.10$	**95.03**	**95.09**	**84.08**
	PFuzzND	87.65	93.08	49.05		*PFuzzND*	94.92	94.90	83.22
	MPFuzzND $\theta_{merge} = 0.3$	**71.90**	**72.48**	**58.72**		*MPFuzzND* $\theta_{merge} = 0.3$	**66.97**	**72.76**	**8.84**
	MPFuzzND $\theta_{merge} = 0.7$	**70.15**	**70.47**	**57.82**		*MPFuzzND* $\theta_{merge} = 0.7$	**67.08**	**72.71**	**8.83**
DS5	*MPFuzzND* $\theta_{merge} = 0.8$	**72.03**	**72.68**	**61.72**	*DS10*	*MPFuzzND* $\theta_{merge} = 0.8$	**62.20**	**68.19**	**9.75**
	MPFuzzND $\theta_{merge} = 1.10$	69.72	70.08	56.90		*MPFuzzND* $\theta_{merge} = 1.10$	57.19	64.52	8.41
	PFuzzND	70.00	70.46	59.10		*PFuzzND*	61.11	67.42	7.83

Table 5: Results obtained by the PFuzzND algorithm and proposed modification MPFuzzND with iteratively calculated γ_i

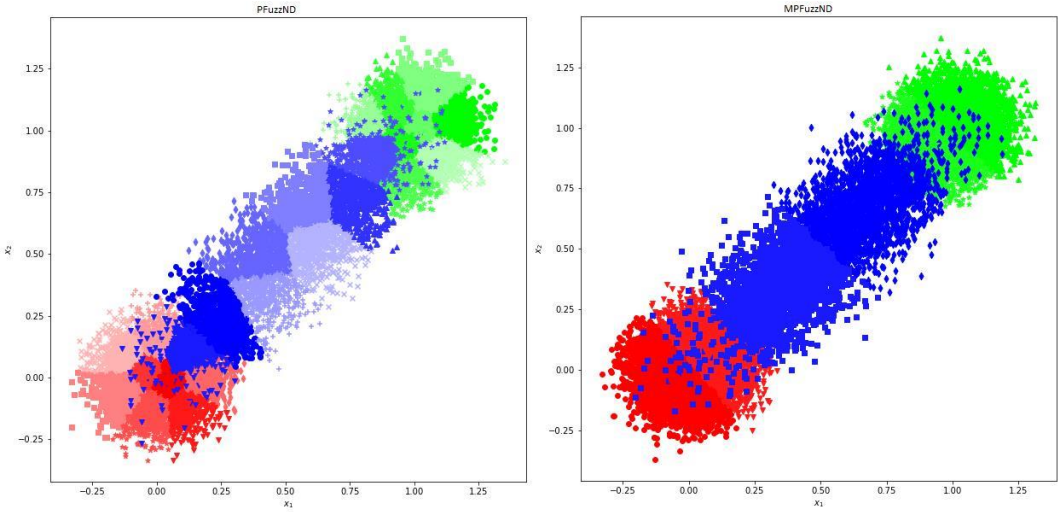

Figure 1: Comparison of PFuzzND and MPFuzzND, DS3

In these tables results, given in bold, are the ones, which were obtained by a given configuration of the proposed modification and were better compared to the results achieved by the original algorithm.

As was mentioned before, the PFuzzND algorithm and as result its modification MPFuzzND are sensitive to their parameters, which now include also the merging threshold θ_{merge}. This threshold was varied in range from 0.1 to 1.4 and only results for 4 different variants of θ_{merge} are presented in both tables, namely:

1. $\theta_{merge} = 0.2$, $\theta_{merge} = 0.6$, $\theta_{merge} = 0.8$, $\theta_{merge} = 1.1$ when γ_i are set to 1;
2. $\theta_{merge} = 0.3$, $\theta_{merge} = 0.7$, $\theta_{merge} = 0.8$, $\theta_{merge} = 1.1$ when γ_i is iteratively calculated.

Lesser values of the θ_{merge} threshold lead to situations when the merging operator is used more frequently, while larger values of θ_{merge} cause it to occur not as often. According to the obtained results, proposed modification MPFuzzND outperforms the original algorithm on 7 datasets out of 10 when $\theta_{merge} = 1.1$ in case if γ_i are set to 1 for all clusters and similarly when $\theta_{merge} = 0.7$ in case if γ_i are iteratively calculated. Moreover, for some problems configurations of the MPFuzzND approach with low values of θ_{merge} outperform the ones with the greater θ_{merge}. Thus, conducted experiments showed that there is a need to adjust the merging threshold automatically.

Figure 1 shows the comparison of PFuzzND and MPFuzzND with $\gamma_i = 1$ and $\theta_{merge} = 0.3$ on DS3. As it can be seen from the figure the original algorithm PFuzzND divided the biggest class into seven clusters, while the proposed modification MPFuzzND merged them all into one cluster, and in the end model generated by the modification was able to outperform the one generated by the original approach in terms of all metrics, used in this study. Thus, from Figure 1 it can be clearly seen that MPFuzzND is merging the clusters within one class, so that the data is described better: in the left part of the figure the PFuzzND generated too many clusters for the central part, which led to incorrect classification in the overlapping areas.

Overall, the proposed method was able to detect class extensions in the classification step and adapt the model during the online phase. Moreover, the MPFuzzND algorithm also could detect the new classes, which emerged along the online phase for datasets, which had unlabelled instances (the latter is shown in the column with the unknown measure rate values).

It should be noted that generally both the original PFuzzND algorithm and introduced in this study MPFuzzND modification show better results in term of the unknown rate measure (closer to zero) and

the Macro *F*-score metric when γ_i are set to 1. Additionally, in general the new MPFuzzND algorithm classified correctly instances not labelled as unknown more often compared to the PFuzzND approach. Thus, the MPFuzzND algorithm was able to better represent the distribution of the given data.

4 Conclusions

In this study a new modification MPFuzzND of the Possibilistic Fuzzy multiclass Novelty Detector for data streams is introduced. The main idea was to merge clusters belonging to one class so that the number of its clusters would not be too large because the latter can cause bad classification results. The merging procedure was inspired by the approach introduced in (Xiong, 2004) but was changed according to the characteristics of the original PFuzzND algorithm. It is applied on each iteration of the online phase for each considered class.

Proposed modification was tested on 10 synthetic datasets and its results were compared with results obtained by the PFuzzND approach. To do that three different metrics were used: the accuracy, the Macro *F*-score metric and the unknown rate measure. It was established that generally the MPFuzzND algorithm outperforms the original PFuzzND regardless of the chosen configuration. However, proposed merging technique has new parameter, which should be accurately chosen for each problem, as it is significant in terms of the classification results.

Therefore, in future research an approach for automated adjustment of the proposed modification's parameters, including the new one introduced in this study, should be developed. Additionally, in future more complex datasets can be used to determine whether the proposal provides good results in these cases, with the parameters obtained automatically.

References

Bezdek J.C. (1981). *Pattern recognition with fuzzy objective function algorithms*. New York: Plenum.

Bifet A., Gavaldà R., Holmes G., Pfahringer B. (2018). *Machine Learning for Data Streams: with Practical Examples in MOA.* The MIT Press.

Campello R.J., Hruschka E.R. (2006). A fuzzy extension of the silhouette width criterion for cluster analysis. *Fuzzy Sets and Systems, 157(21)*, 2858-2875.

Cejnek M., Bukovsky I. (2018). Concept drift robust adaptive novelty detection for data streams. *Neurocomputing*, *309*, 46-53.

Chandola V., Banerjee A., Kumar V. (2009). Anomaly detection: a survey. *ACM Computing Surveys, 41*(3), 1-58.

Data stream repository. (n.d.). *Computational Intelligence Group. Department of computingFederal University of Sao Carlos. Sao Carlos, Brazil.* Retrieved from GitHub: https://github.com/CIG-UFSCar/DS_Datasets

de Faria E.R., Gonçalves I.R., Gama J., de Carvalho A.C.P.L.F. (2015). Evaluation of multiclass novelty detection algorithms for data streams. *IEEE Transactions on Knowledge and Data Engineering, 27(11)*, 2961-2973.

de Faria E.R., de Leon Ferreira A.C.P., Gama J. (2016). MINAS: multiclass learning algorithm for novelty detection in data streams. *Data Mining and Knowledge Discovery*, *30*, 640-680.

Faria E.R., Gonçalves I.J.C.R., de Carvalho A.C.P.L.F., Gama J. (2016). Novelty detection in data streams. *Artificial Intelligence Review*, *45*, 235-269.

Gama J. (2010). *Knowledge Discovery from Data Streams.* Chapman & Hall/CRC.

Krishnapuram R., Keller J. (1993). A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems, 1(2)*, 98-110.

Krishnapuram R., Keller J. (1996). The possibilistic c-means algorithm: insights and recommendations. *IEEE Transactions on Fuzzy Systems, 4(3)*, 385-393.

Ouafae B., Oumaima L., Mariam R., Abdelouahid L. (2020). Novelty detection review state of art and discussion of new innovations in the main application domains. In *Proceedings of the 1st International Conference on Innovative Research in Applied Science, Engineering and Technology* (pp. 1-7).

Pal N.R., Pal K., Keller J.M., Bezdek J.C. (2005). A possibilistic fuzzy c-means clustering algorithm. *IEEE Transactions on Fuzzy Systems, 13(4)*, 517-530.

Pimentel M.A.F., Clifton D.A., Clifton L., Tarassenko L. (2014). A review of novelty detection. *Signal Processing, 99*, 215-249.

Silva T.P., Schick L., Lopes P.A., Camargo H.A. (2018). A fuzzy multiclass novelty detector for data streams. In *Proceedings of the 2018 IEEE International Conference on Fuzzy Systems* (pp. 1-8). IEEE.

Silva T.P., Camargo H.A. (2020) Possibilistic approach for novelty detection in data streams. In *Proceedings of the 2020 IEEE International Conference on Fuzzy Systems* (pp. 1-8). IEEE.

Škrjanc I., Iglesias J.A., Sanchis A., Leite D., Lughofer E., Gomide F. (2019). Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey. *Information Sciences, 490*, 344-368.

Xiong X., Chan K.L., Tan K.L. (2004). Similarity-driven cluster merging method for unsupervised fuzzy clustering. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence* (pp. 611-618).