2015-11

Evaluation of multiclass novelty detection algorithms for data streams

IEEE Transactions on Knowledge and Data Engineering, Los Alamitos, v. 27, n. 11, p. 2961-2973, Nov. 2015
http://www.producao.usp.br/handle/BDPI/50734

Evaluation of Multiclass Novelty Detection Algorithms for Data Streams

Elaine Ribeiro de Faria, Isabel Ribeiro Gonçalves, João Gama, and Andre Carlos Ponce de Leon Ferreira Carvalho

Abstract—Data stream mining is an emergent research area that investigates knowledge extraction from large amounts of continuously generated data, produced by non-stationary distribution. Novelty detection, the ability to identify new or previously unknown situations, is a useful ability for learning systems, especially when dealing with data streams, where concepts may appear, disappear, or evolve over time. There are several studies currently investigating the application of novelty detection techniques in data streams. However, there is no consensus regarding how to evaluate the performance of these techniques. In this study, we propose a new evaluation methodology for multiclass novelty detection in data streams able to deal with: i) unsupervised learning, which generates *novelty patterns* without an association with the true classes, where one class may be composed of a novelty set, ii) confusion matrix that increases over time, iii) confusion matrix with a column representing *unknown* examples, i.e., those not explained by the model, and iv) representation of the evaluation measures over time. We propose a new methodology to associate the *novelty patterns* detected by the algorithm, in an unsupervised fashion, with the true classes. Finally, we evaluate the performance of the proposed methodology through the use of known novelty detection algorithms with artificial and real data sets.

Index Terms—Evaluation methodologies, novelty detection, data streams

1 Introduction

Data stream is an active research area that investigates the extraction of knowledge from large amounts of continuous data. A data stream (DS) presents important characteristics such as non-stationary distribution, in which the learned concepts can evolve over time, and unbounded data arrival. In this context, new problem classes (new true classes) may appear, disappear, reappear or evolve over time. According to [1], the most challenging tasks in DSs are concept drift, which means a change in the data distribution, and concept evolution, which means emergence of novel classes.

Novelty detection (ND) is a stream mining task that assesses if an example or a set of examples differ significantly from the previously generated examples. This is considered a useful ability for learning systems, especially when the data are acquired incrementally [2]. ND is an important task for DS mining as it allows for the recognition of novel concepts, which may indicate the appearance of a new concept, a change in known concepts or the presence of noise [3].

There are many real problems that generate data continuously, where the application of ND techniques is useful, as, for example, intrusion detection [4], [5], fault detection [6], [7], fraud detection [8], forest cover detection [1], spam filter [9] , and text classification [10].

Most of the previous studies see ND as a one-class task, whose goal is to discriminate examples from the "Normal" and "Not-Normal" classes [11], [12], [13], [14], [15]. These studies use binary classification measures to evaluate the investigated algorithms. However, several real problems are not one-class, which makes the use of these algorithms and evaluation measures impractical.

Recent studies have treated ND as a multiclass classification task, where the normal concept can be made up of by different classes, and novel classes may appear over the course of time [1], [16], [17], [18], [19], [20], [21]. However, most of them use binary classification measures to evaluate the investigated classifiers [1], [16], [18]. These measures are unable to express the problem properly, especially when different *novelty patterns* (NP) appear over time.

For ND in DSs, it is common to use the term *unknown* to represent an example not explained by the current model, but used to model new concepts or extensions of the known concepts [15], [17]. This is an important issue to be considered in the evaluation of ND, but it has been neglected by existing approaches.

In this study, we propose a new evaluation methodology for ND in multiclass DSs. The proposed methodology can also be used in one-class tasks, since it can be considered a particular case of multiclass classification. This methodology is able to deal with:

- unsupervised learning, which generates NPs without any association with the problem classes, and where one class may be composed of more than one novelty;
- confusion matrix that increases over time;

- *E. R. Faria is with the Computing School, Federal University of Uberlândia, Uberlândia, Brazil. E-mail: elaine@facom.ufu.br.*
- *I. Gonçalves is with the Instituto Politécnico de Viana do Castelo, Viana do Castelo, Portugal and the University of Porto, Porto, Portugal. E-mail: isagoncalves@estg.ipvc.pt.*
- *J. Gama is with the LIAAD-INESC Porto, University of Porto, Porto, Portugal. E-mail: jgama@fep.up.pt.*
- *André C. P. L. F. Carvalho is with the ICMC, University of São Paulo, São Carlos, Brazil. E-mail: andre@icmc.usp.br.*

Manuscript received 5 Aug. 2014; revised 19 May 2015; accepted 28 May 2015. Date of publication 3 June 2015; date of current version 2 Oct. 2015. Recommended for acceptance by L. Khan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2015.2441713

- confusion matrix with a column representing the *unknown* examples, i.e., those not explained by the current model;
- representation of the evaluation measures over time;
- problem of the reduction of the classification error rate as the number of NPs increases, i. e., the more NPs detected, the lower the classifier error.

This study proposes a new methodology to associate the NPs detected by a ND algorithm, in an unsupervised learning, with the problem classes, where one class may be composed of a set of NPs. To the best of our knowledge, this is the first study that considers an incremental confusion matrix, where the number of true classes may be different from the number of predicted classes and the detected NPs are unlabeled and do not have a direct match with the problem classes. We evaluate the performance of the proposed methodology by known ND algorithms with artificial and real data sets and compare it with the methodology commonly used in the literature.

A preliminary version of this work was presented in [22]. The current work incorporates a large number of modifications, resulting in several new contributions. First, it has a new formalism that facilitates the distinction between ND in one-class classification and ND in multiclass problems. Second, it extends the previous review of the state of the art. Additional algorithms found in the literature are discussed and more details are provided, especially when considering the confusion matrix computed for each algorithm. Third, the proposed methodology is incremented by the addition of model selection techniques. These techniques were added to address an important issue, that has not been addressed by the literature evaluation methodologies for ND algorithms in DSs, which is the decrease of the misclassification rate as the number of NPs increases. Thus, it is not enough to obtain a classifier with low error rate. It is also important to obtain a model with a low number of NPs. Fourth, the experimental section uses a larger number of algorithms and data sets. Finally, it presents a deeper comparative analysis between the methodology used in the literature and our proposed methodology.

The paper is organized as follows. Section 2 presents the problem formulation, the main challenges to be addressed and the importance of a new methodology to evaluate ND in DSs. Section 3 describes the main related work. Section 4 details the proposed methodology. Section 5 shows the experiments carried out and the results, measured by the proposed methodology, obtained for different data sets using ND algorithms from the literature. Finally, Section 6 summarizes the conclusions and limitations, as well as discusses future works.

2 Formalization of the Problem

A DS can be described as a sequence of examples that continuously flow, and whose data distribution can change over time.

Definition 1 (Data Stream). *A data stream $\mathcal{S}$ is a substantial sequence of examples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$,* i.e., *$\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^{N}$, which is potentially unbounded ($N \rightarrow \infty$). Each example is described by an $n-$dimensional attribute vector $\mathbf{x}_i = [x_i^j]_{j=1}^{n}$ [23].*

TABLE 1
Table of Symbols

Symbol	Description
C_{knw_i}	ith known problem class
C_{nov_i}	ith novel class
NP_i	ith *novelty pattern*
FP_i	number of false positives of the class i
FN_i	number of false negatives of the class i
TP_i	number of true positives of the class i
ExC_i	number of examples of the class i
Unk_i	number of *unknown* examples of the class i
Ex	number of examples considering all problem classes
M	number of problem classes
FE	number of misclassification in the known classes (other than FP)
NC	number of examples from novel classes
NS	number of examples in the stream

An important task related to DSs is ND. In general, algorithms for ND in DSs work in two phases, namely *offline* and *online*. In the *offline* phase, a set of labeled examples is used to induce a classifier. These labeled examples represent the known concept concerning the problem. Usually, the known concept is composed of examples from only one class, named *normal class*. In the *online* phase, whenever a new example arrives, it is classified in the normal class or it is rejected (or classified as abnormal, anomaly or novelty). This is the classical setting in one-class-classification [15], [24], [25], [26], [27], [28], [29], [30].

Recently, several authors extended this framework to a multiclass context [1], [16], [17], [18], [19], [20], [31], [32]. In this work, we propose a generalization of the previous formalization to the multiclass context. In this generalization, in the *offline* phase, each example from the training set has a label (y_i), where $y_i \in Y^{tr}$, with $Y^{tr} = \{C_{knw_1}, C_{knw_2}, .., C_{knw_L}\}$, where C_{knw_i} represents the ith known class of the problem and L is the number of known classes. In the *online* phase, as new data arrive, new *novel classes* can be detected, expanding the set of class labels to $Y^{all} = \{C_{knw_1}, .., C_{knw_L}, C_{nov_1} \ldots, C_{nov_K}\}$, where C_{nov_i} represents the ith *novel class* and K is the number of *novel classes*, which is previously unknown. Table 1 presents the main symbols used in this paper.

Definition 2 (Novel Class). *A class that is not available in the training phase (*offline*), appearing only in the* online *phase.*

Initially, a classifier can deal effectively only with examples from the training classes. When examples belonging to novel classes appear over the stream, they are temporally classified as *unknown*.

Definition 3 (Unknown). *An example not explained by the current model. In one-class classification tasks, it is named as abnormal, rejected or anomaly, i.e., the example does not belong to the normal concept. In some contexts, this is sufficient. In multiclass classification tasks, a group of* unknown *examples can be used to model new concepts.*

The *unknown* examples are submitted to ND procedure in order to produce different NPs (see Fig. 1).

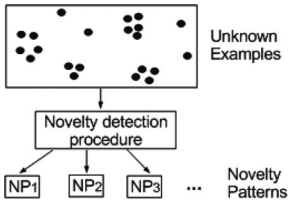


Fig. 1. Example of the process for NP detection from unknown examples [22].

Definition 4 (Novelty Pattern (NP)). *A pattern identified in unlabeled examples, previously considered as* unknown *by the classification system.*

A common unsupervised approach to detect NP is to group similar unknown examples using a clustering algorithm. A cluster or group of clusters constitute a NP, which is later used to classify new examples. However, it is not easy to associate NPs detected by a learning algorithm to the problem classes, where one problem class can be represented by one or more NPs. In addition, due to the presence of concept drift, phenomenon in which the known concept changes over time, the algorithm may also detect NPs to represent extensions of the known classes.

As proposed in [17], the confusion matrix (see Fig. 2) resulting from the classification task is not square and the number of columns increases when new NPs are discovered. Each row of the confusion matrix represents one of the problem classes (known and novel classes) and each column represents one of the predicted classes. The columns C_{knw_1}, $C_{knw_2}, \ldots, C_{knw_L}$ correspond to the classes learned during the *offline* phase, the columns $NP_1, NP_2, \ldots$ correspond to the NPs learned in the *online* phase and the last column is the *unknown* (*Unk*) label. For unsupervised algorithms, the NPs detected over time do not have a direct matching with problem classes. The NPs are sequentially labeled as NP_1, NP_2, etc. Besides, a given problem class can be associated with one or more NPs and a particular class may not be detected by the algorithm.

In order to use this confusion matrix, six requirements must be considered:

- one class may be represented by two or more NPs, thus it is possible to have more NPs than problem classes;
- ND algorithm can detect less NPs than the number of *novel classes*. This may happen if the algorithm did not properly distinguish the examples from all the *novel classes*;
- presence of examples not explained by the current model and labeled by the algorithm as *unknown*;
- multiclass scenario, i.e., the computation of accuracy or error measures have to consider the different classes learned in the *offline* and *online* phases, which is harder than to distinguish between normal and novel concepts;
- error rate tends to decrease when the number of NPs increases;
- representation of the confusion matrix that can vary over time.

The evaluation methodology employed by the proposed approach has to deal with all these requirements. In order to deal with the first two requirements, it is necessary to associate NPs to classes. Section 4.1 explains the approach adopted in this study. An alternative to deal with the third requirement is discussed in Section 4.2. A solution for the fourth problem is proposed in Section 4.3. An approach to deal with the fifth requirement is presented in Section 4.4. Finally, Section 4.5 describes how the last requirement is met.

3 Novelty Detection Evaluation

This section presents other studies found in the literature related to the evaluation of ND algorithms in DS mining. It also discusses evaluations that consider a rejection option and multiclass classification tasks.

3.1 Novelty Detection Evaluation in Data Streams

Many algorithms have been proposed to deal with ND in DSs. However, little attention has been devoted to an adequate evaluation of these algorithms. The ND evaluation methodologies found in the DS literature follow three approaches, regarding how they consider ND:

- ND as a one-class classification task;
- ND as a multiclass classification task but only one NP appears at a time;
- ND as a multiclass classification task where more than one NPs may appear.

OLINDDA [15] and DETECTNOD [24] are algorithms for ND in DSs that belong to the first group. They present the following features:

- known concept is composed by only one class, the normal class;
- decision model is composed by three sub-models: normal, representing the known concept, extension, representing extensions of the normal class, and novelty, created to represent the novel classes that appear over time;
- each decision model is represented by a set of clusters;

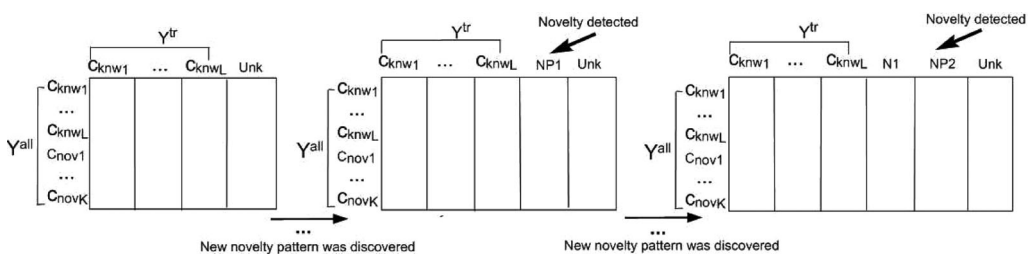


Fig. 2. Evolution of the confusion matrix over time (adapted from [22]).

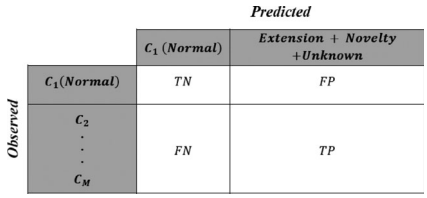


Fig. 3. OLINDDA and DETECTNOD confusion matrix.

- examples not explained by the current decision model are marked as *unknown* and can be later used to update the decision model;
- novelty sub-model does not distinguish between different NPs, i.e., examples from different novel classes may be classified simply as "novelty";
- extension and novelty sub-models are updated by adding new clusters created with the *unknown* examples. Thus, the update operation uses unlabeled examples only.

The confusion matrix generated by OLINDDA [15] and DETECTNOD [24] algorithms is illustrated in Fig. 3. In this figure, the first row represents the normal concept (negative class), C_1. The second row represents the other class or (positive class), which includes the observed classes C_2 to C_M, where M is the number of observed *classes* over the stream.

To evaluate this confusion matrix, OLINDDA and DETECTNOD use the following measures $Mnew$, the percentage of *novel class* examples misclassified as belonging to the normal model, and $Fnew$, the percentage of normal class examples wrongly labeled as belonging to the novelty or extension models, see Equation (1).

In this Equation, FP is the number of elements from the normal class wrongly classified as novelty, extension or *unknown*, FN is the number of examples from the *novel classes* classified as belonging to the normal classes, NC is the number of examples from the *novel classes* in the DS, and NS is the number of examples in the DS. This evaluation methodology has two limitations: the examples marked with the *unknown* profile are computed as error (for the normal class) or hits (for the novel classes) and it can only be used for binary classification tasks,

$$Mnew = \frac{FN * 100}{NC} \qquad Fnew = \frac{FP * 100}{NS - NC}. \tag{1}$$

The second approach is adopted by the ECSMiner [1] and CLAM [18] algorithms. These studies assume that the known concept concerning the problem can be composed of different classes, therefore they use multiclass classification algorithms. However, they consider that only one novel class can appear at each time instant. They present the following features:

- known concept may be composed of a set of classes;
- decision model represents the known classes and is updated over the stream whenever a new labeled chunk is available;
- use an ensemble of classifiers, induced by a decision tree induction algorithm or the KNN algorithm;
- each example can be classified in up to Tc time units. Thus, the examples explained by the model are immediately classified, while those not explained, wait for the arrival of new similar examples to build NPs. A new example is not added to the confusion matrix until it is classified. Before its classification, the example is seen as "*unknown*";
- consider only one NP per chunk. If examples from more than one novel class appear at the same data chunk, they may be classified only as "novelty", without distinguishing between them;
- assume that after a delay of Tl time units, the true label for all examples will be obtained. The model is updated using these labeled examples in a new training phase.

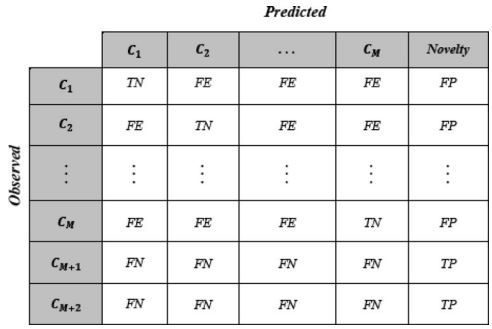


Fig. 4. ECSMiner confusion matrix.

In the confusion matrix generated by ECSMiner and CLAM, illustrated by Fig. 4, the rows correspond to the observed classes and the columns to the predicted classes. The classes C_1 to C_M represent the current known classes, which were learned either *offline* or *online*, using labeled examples. The classes C_{M+1} and C_{M+2} represent two novel classes observed in the current chunk. In this matrix, TN represents the hits in the known classes. For the examples from the novel classes, a hit means classify it as novelty (TP). In this confusion matrix, FN and FP are defined as previously, and FE is the number of misclassifications in the known classes.

In addition to the $Mnew$ and $Fnew$ measures, ECSMiner and CLAM, like other algorithms [16], use the Err measure—percent of total misclassification (see Equation (2)). The main limitation of this approach is, although considering ND as a multiclass classification task, they use binary evaluation measures. Besides, the Err measure does not consider as an error the classification of examples from different classes in the same NP,

$$Err = \frac{(FP + FN + FE) * 100}{N}. \tag{2}$$

The third approach, adopted by the MINAS algorithm [17], has the following features:

- distinct known concepts can be represented by different classes;
- there is only one decision model, created in the *offline* phase and updated over the stream;
- each class is represented by a set of clusters, as well each NP;
- examples not explained by the decision model are marked as *unknown* and used to model either NPs or extensions;

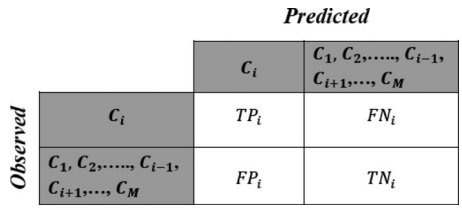


Fig. 5. Confusion matrix for multiclass tasks.

- a NP is composed of a set of clusters and different NPs can be identified over time.

The confusion matrix proposed by the authors and used by MINAS can be seen in Fig. 2. This matrix can be incrementally modified, by adding a new column whenever a new NP is detected. In its first version, evaluation measures were not extracted from this matrix.

3.2 Evaluation Measures for Classification with Reject Option

Most of the classifiers used in the literature predict the class label of all examples, even if there is uncertainty in this prediction. Some studies stress the importance of considering rejection explicitly [25], [33], [34], [35]. Classification with rejection adds a reject option to a classifier to highlight when there is not enough evidence to assign an example to one of the existing classes.

In the classification task with a reject option [25], [33], [34], [35], an example is rejected if its true class cannot be reliably predicted [34]. It is considered to be better to reject an example than to misclassify it. For these situations, the accuracy and error rate may be calculated either by considering all the examples or by taking into account only the examples accepted by the classifier. In both cases, the following properties can be verified [36]:

$$p(accept) = 1 - p(reject) \tag{3}$$

$$p(f(x) = y) + p(f(x) \neq y) + p(reject) = 1 \tag{4}$$

$$p(f(x) = y|accept) + p(f(x) \neq y|accept) = 1. \tag{5}$$

In Equation (3), the rejection rate is the probability that a given classifier rejects a new example and the acceptance rate is the probability that a given classifier accepts the example. The acceptance and rejection rates are complementary. Equation (4) considers that the rejected examples are neither a hit nor an error, but they are computed separately. Equation (5) is the probability of making an incorrect classification, given the classifier has accepted an example.

3.3 Evaluation Measures for Multiclass Classification

Several DS classification tasks have more than two classes. Although the studies on ND in DSs have not explored the use of multiclass evaluation measures, this issue has been studied intensely in other contexts.

One approach to evaluate the predictive performance of multiclass classifiers is to divide the original $M \times M$ confusion matrix into M binary one-against-all matrices, one for each class (see Fig. 5). For each class C_i, TP_i is the number of examples from the class C_i correctly classified, FP_i is the number of examples from the class $C_j (j = 1, \ldots, M, j \neq i)$ incorrectly classified as belonging to class C_i, FN_i is the number of examples from the class C_i wrongly classified as belonging to another class C_j, and TN_i is the number of examples from the class $C_j (j = 1, \ldots, M, j \neq i)$ not classified as belonging to the class C_i.

An error measure applied to multiclass classification tasks, defined in Equation (7), is the combined error (*CER*), which is the average of the weighted rate of false positive and false negative per class (see Equation (6)) [37]. In this equations, $\#ExC_i$ represents the number of examples from the class C_i and $\#Ex$ represents the total number of examples. Another measure used to evaluate the predictive performance of classification algorithms is the F-measure, defined as the weighted harmonic mean of precision and recall. This measure can be adapted to multiclass classification tasks, as proposed by [38]. In this case, the average of the F1-measure (F-measure with $\alpha = 1$) is calculated for each class.

These measures can be applied to multiclass classification tasks and they are adequate for dealing with unbalanced data. However, the *unknown* examples are not taken into account. In this study, we extend these measures to take into account the unknown examples,

$$FPR_i = \frac{FP_i}{FP_i + TN_i} \qquad FNR_i = \frac{FN_i}{FN_i + TP_i} \tag{6}$$

$$CER = \frac{1}{2} \sum_{i=1}^{M} \frac{\#ExC_i}{\#Ex} (FPR_i + FNR_i). \tag{7}$$

4 The Proposed Methodology

In order to evaluate the incremental confusion matrix generated by ND algorithms for DSs (see Fig. 2), some issues must be addressed.

- *How to process rectangular confusion matrices?*

  The problem. In general, the existing approaches apply evaluation measures to a confusion matrix, assuming that the matrix is square and the main diagonal represents the examples correctly classified for each class. However, as the confusion matrix presented in this work is rectangular (see Confusion Matrix 1 in Fig. 6), where the number of predicted classes is not the same as true classes, evaluation measures cannot be directly applied to this matrix. In addition, the NPs are not labeled, and thus are not directly associated with the true classes.

  Approach proposed in this study. To deal with this confusion matrix, unlabeled NPs are associated with the true classes (see Section 4.1). Fig. 6 (Confusion Matrix 1 and Confusion Matrix 2) illustrates a confusion matrix before and after this association.
- *How to evaluate the examples marked as unknown?*

  The problem. It is necessary to decide how to evaluate the last column of the confusion matrix (see Fig. 6, Confusion Matrix 2), which represents the examples marked with the *unknown* profile. It is expected that the number of *unknown* examples increases in the presence of novel classes or concept

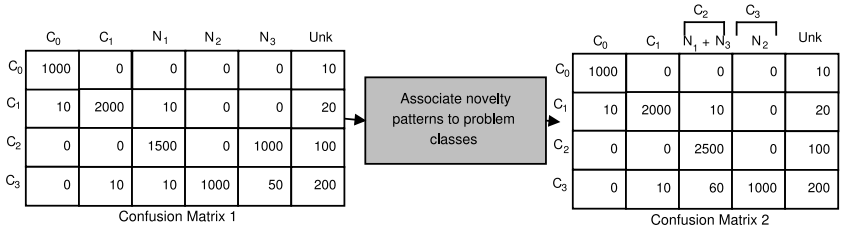


Fig. 6. Overview of the proposed evaluation methodology.

drift. However, when these changes are learned, this number should decrease. Therefore, to associate the *unknown* examples with hits or errors may not be an adequate solution.

Approach proposed in this study. Compute error/accuracy measures not taking into account the *unknown* examples and compute a measure to evaluate the number of *unknown* examples per class (see Section 4.2).

- *Which measures should be used to evaluate the confusion matrix?*

  The problem. The measures commonly used in the literature, $Mnew$, $Fnew$ and Err are inadequate for multiclass classification tasks and imbalanced classes.

  Approach proposed in this study. Use a measure developed for multiclass classification tasks, like CER, and plot its values on a graphic together with the evaluation measure for the *unknown* examples ($UnkR$).
- *How to treat the problem of reduction in the error rate as the number of NPs increases?*

  The problem. The error of the classifier decreases as the number of NPs increases. Thus, when comparing two or more classifiers, it is important to take into account not only the error rate, but also the number of NPs. A good model should obtain a low error rate with a feasible number of NPs. The difficulty is how to compare two classifiers regarding their error rate and number of NPs.

  Approach proposed in this study. Use penalized likelihood method for model selection, such as AIC. This method penalizes the complexity of the model, where complexity is measured by the number of classes detected by the algorithm (NPs plus known classes).

4.1 A Rectangular Confusion Matrix for ND

Considering that the *online* phase is unsupervised, the NPs detected by the algorithm do not have a direct matching with the true classes. In addition, the number of NPs is not equal to the number of true classes. To build a square confusion matrix it is necessary to associate the NPs with the true classes. However, the number of possible associations is exponential. The objective of this section is to explain the inspiration for the solution proposed for this problem and how the proposed solution works.

The inspiration to solve this association problem is the Hungarian method [39], which is a combinatorial algorithm used in the assignment problem. However, this method cannot be directly used because it assumes a one-to-one correspondence, but in the novelty-class matching problem, one true class can be represented by one or more NPs.

The problem can be formalized using a weighted bipartite graph.

Definition 5 (Bipartite Graph). *A graph can be represented by $G(V, E)$, where V is the set of vertices and E is the set of edges on the graph. A graph $G(V, E)$ is bipartite with two set of vertices X and Y, if $V = X \cup Y$ with $X \cap Y = \emptyset$ and each edge in E has one endpoint in X and one endpoint in Y. If for each $v_i \in X$, $v_j \in Y$, $\{v_i, v_j\} \in E$, the graph is named* complete bipartite graph.

Definition 6 (Weighted Bipartite Graph). *A bipartite graph $G(V, E)$ is weighted if each edge $\{v_i, v_j\} \in E$ has a associated weight $w_{ij} \geq 0$.*

In the context of ND in multiclass classification tasks, X represents the NPs predicted by the algorithm, Y the true classes, and w_{ij} the number of examples from the class i classified as belonging to the class j. Figs. 7a and 7b show an example of a confusion matrix and its corresponding complete bipartite graph G. The weight value associated with each edge is omitted to simplify the figure.

The aim here is to find a weighted bipartite subgraph $G'(V, E')$, where each vertex in X has a degree of one. In this case, $|E'|$, the number of edges on the graph G', is equal to $|X|$ (number of elements in X). Thus, it is necessary to associate each NP with a true class. However, there are many different ways to compute this new subgraph G'. For each novelty $x_j \in X$, there are $|Y|$ different possible associations between x_j and a class $y_i \in Y$, where $|Y|$ is the number of true classes. The number of possible combinations to associate each element from X with one element from Y is therefore $|Y|^{|X|}$.

The algorithm based on graph theory used to associate NPs to true classes is shown in Algorithm 1. Regarding

	C_{knw1}	C_{knw2}	NP_1	NP_2	NP_3	NP_4	Unknown
C_{knw1}	6000	2900	0	800	50	0	100
C_{knw2}	2000	5200	700	0	30	0	50
C_{nov1}	300	450	1950	100	100	50	50
C_{nov2}	200	300	1900	1400	100	20	100

(a)

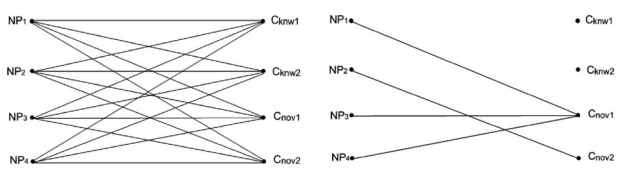


Fig. 7. Example of a confusion matrix and its correspondent bipartite graph (adapted from [22]); (a) Confusion Matrix; (b) Corresponding Complete Bipartite Graph; and (c) Resulting Bipartite Subgraph representing the association between NPs and true classes.

bipartite graphs, the goal is to find a perfect match with minimum cost, i.e., an approach that minimizes the classifier error. This restriction is overcome by choosing, for each element $x_j \in X$, the edge w_{ij} with the highest weight, indicating that the element x_j is associated with the element y_i. In case of a draw, any one of these can be chosen. Fig. 7c shows the subgraph G' resulting from the association process between NPs and true classes for the confusion matrix shown in Fig. 7a.

Algorithm 1. Association NPs to true Classes

Require: *NPSet*: *novelty patterns* set,
W: matrix of weight,
PC: true classes set
1: **for all** novelty pattern N_j in *NPSet* **do**
2: Find the highest weight w_{ij} for N_j, $1 \leq i \leq |PC|$
3: Associate N_j to PC_i
4: **end for**

One observes that every element in X has a corresponding element in Y, but the reciprocal is not true. Fig. 7c can be described as follows. The proposed algorithm associated three NPs to the class C_{nov_1} and one NP to the class C_{nov_2}. For the classes induced in the *offline* phase (C_{knw_1} and C_{knw_2}), the classifier did not associate any novelty to represent them. Thus, if the classes learned *offline* evolved over time, the classifier could identify these changes as extensions of the known concepts, instead of novelties.

4.2 The Problem of the *Unknown* Examples

An important issue to be addressed is the presence of *unknown* examples in the confusion matrix. In this paper, the *unknown* examples are not considered either as a hit or as an error, but they have to be computed separately. It is important to highlight that, according to the proposed methodology, $ACC + Err + UnkR = 1$, where ACC is the rate of examples correctly classified, Err is the rate of examples incorrectly classified, and $UnkR$ is the rate of examples classified as *unknown*.

As proposed by the classifiers with reject option [34], one possible alternative is to compute measures like error and accuracy using only the examples explained by the model. Thus, $ACC_{Exp} + Err_{Exp} = 1$, where ACC_{Exp} and Err_{Exp} are the accuracy and error rates, considering only the examples explained by the model. As a result, the FN_i measure could be the number of examples from the class i incorrectly classified as belonging to another class, except the examples classified as *unknown*.

In order to verify how the number of *unknown* examples varies over time, we computed the *unknown* rate for each class, and then the average of these unknown rates, according to Equation (8),

$$UnkR = \frac{1}{M}\sum_{i=1}^{M}\frac{\#Unk_i}{\#ExC_i}. \quad (8)$$

4.3 Adaptation of the Multiclass Measures

After associating classes with NPs and computing the *unknown* examples rate, we use an evaluation measure from the literature, CER, to express the classification errors of a learning algorithm, considering only the examples classified by the algorithm as not *unknown*. Thus, we propose the use of the CER measure (see Equation (7)), computing $\#ExC_i$, $\#Ex$, FPR_i and FNR_i without considering the *unknown* examples.

4.4 Model Selection

When a decision model is created, the error obtained in the classification of new examples is expected to be low. This is specially noted when comparing two models, the model with the lowest error is the best. Another important issue to be considered is the model complexity. It is important to create models with both low error and low complexity.

In the ND task, model complexity can be estimated as the number of detected NPs plus the number of classes used in the training phase. When this number increases, the error measure decreases, but resulting in a more complex decision model. In an extreme situation, in which each new unlabeled example from a novel class is identified as a NP, the error measure will decrease drastically, tending to zero. This will result in a low error measure, but a very complex model.

To find a model that not only provides a good prediction quality (low CER value), but also has a low level of complexity, an adaptation of a measure from the literature is used in the proposed methodology: the Akaike Information Criterion (AIC) [40], [41] (see Equation (9))

$$AIC = -2ln(L) + 2p/ln(N). \quad (9)$$

In Equation (9), L corresponds to the maximum value of the likelihood function, p is the number of free parameters and N is the total number of examples in the data set. This measure rewards a high model performance and penalizes a high model complexity, taking into account the number of classified examples. Using this criterion, the model with the lowest value for AIC is selected.

As L represents the maximum likelihood, i.e., the agreement between the model and the observed data, we can calculate it using the the error complement (1-CER), p is the number of classes detected (known classes plus NPs) by the system, and N is the total number of observations without considering the unknown examples.

4.5 Evaluation over Time

For ND in DSs, as new data arrive and new classes may appear, disappear or evolve, it is not sufficient to compute only one confusion matrix. It is necessary to evaluate the confusion matrices over time to verify how a classifier adapts to the non-stationary scenario.

The confusion matrix, illustrated by Fig. 7, can be easily maintained incrementally. Whenever a new example arrives, it is incremented and the evaluation measures can be updated. However, in terms of computational cost, it is not interesting to compute this measure every time a new example arrives. Thus, the evaluation measures are computed after a given period of time, but the confusion matrix is incremented every time a new example arrives.

A possible way to verify the classifier behavior over time is by building a 2D-graphic, where axis X represents the data timestamps and axis Y represents the values for the

TABLE 2
Data Sets Used in the Experiments

Data set	Attributes	Examples	Classes	Training classes
MOA	4	100,000	4	2
SynEDC	40	400,000	20	7
KDD	34	490,000	5	2
KDD-V2	34	490,000	5	1
FCT	54	540,000	7	3

evaluation measures. On this graphic, it is important to plot one measure representing the *unknown* rate in comparison with one or more measures of the accuracy or error rate. Additionally, it is important to highlight on this graphic the detection of a new concept by the algorithm.

5 Experiments

In this section, the experiments carried out in this study are presented. Initially, the data sets and algorithms used in the experiments are briefly described. Afterwards, the experiments performed are presented followed by the analysis of their results.

5.1 Experimental Settings

Table 2 summarizes the main features of the data sets used in the experiments. MOA (for details see [17]) and SynEDC[1] are artificial data sets. KDD Cup 99 Network Intrusion Detection (KDD) [42] and forest cover type (FCT) [42] are real data sets frequently used in ND experiments.

Since OLINDDA, one of the algorithms used in the experiments, considers the ND task as a binary classification task (one normal class and one novelty class), another version of the KDD data set was created, named KDD-V2, which contains in its training subset only examples from the normal class.

There is no consensus concerning the best sampling methodology for the model validation used in the experiments involving ND in DSs. Algorithms like OLINDDA [15] and DETECTNOD [24] use 10-fold-cross validation. Other algorithms, like ECSMiner [1] and CLAM [18], use w data windows of size s in the training phase (*offline*) and the remaining data in the test phase.

In the experiments performed for this study, for each data set, 10 percent of the data are used in the training phase, and the remaining data in the test phase. The order of the examples are the same as in the original data set.

The following algorithms were used in the experiments: OLINDDA[2] [15], MINAS[3] [17], ECSMiner[4] [1] and CLAM[5] [18]. The main motivation to choose these algorithms is that they present the ND task under different perspectives: multiclass supervised task with a time constraint, one class unsupervised task and multiclass unsupervised task. However, the proposed methodology can be easily applied to other algorithms for ND in DSs.

Since OLINDDA assumes that the known concept has only one class, OLINDDA was applied only to the KDD-V2 data set, in which the known concept is composed of the normal class. The clustering algorithm used is k-Means [43], [44], with $k = 20$. The other settings are the same as those used in [15].

The clustering algorithm used in the MINAS algorithm is CLUSTREAM [23]. The ND procedure is executed when the temporary memory size reaches at least 2,000 samples. The window size for the forgetting mechanism is equal to double that of the temporary memory size. The threshold is automatically determined as the standard deviation of the distance among the examples and the centroid multiplied by a factor 1.1.

The setting for the experiments with the ECSMiner algorithm is the same used in [1]: window size equal to 2,000, number of ensembles equal to 6, parameters Tl and Tc equal to 1,000 and 400, respectively. The ND procedure is executed when the temporary memory reaches 50 examples, and the number of clusters is 50. The same setting is adopted for the experiments using the CLAM algorithm. As the base algorithm, ECSMiner uses J48 [45] and CLAM uses KNN [46].

In OLINDDA and MINAS, the *online* phase updates the decision model without external feedback, therefore it does not use the class label of the examples. ECSMiner and CLAM, on the other hand, update the decision model using external feedback. Thus, a new training phase is executed when a set of labeled examples is available. Thus, it is difficult to experimentally compare these two groups of algorithms. The algorithms with external feedback are expected to obtain better predictive performance.

MINAS and OLINDDA perform the supervised training phase (offline) only once, while ECSMiner and CLAM execute it at pre-defined time intervals. In order to compute the hits/errors of the classifiers, we need to define what will be considered as novel classes for each algorithm. For MINAS and OLINDDA, the novel classes are the classes not learned in the single offline phase. For ECSMiner and CLAM, the novel classes are updated constantly, since whenever a new training phase is executed, the examples from the novel classes of the last chunk are labeled and these classes are no longer considered novel. Thus, if a new training phase is executed between t and $t + n$, a class assumed as novel in a time window t, may not be considered novel in a time window $t + n$.

ECSMiner and CLAM use a time constraint in which an example can be classified in up to Tc time units after its arrival. When an example is not explained by the current model, the system can wait until Tc time units to label it and include it in the confusion matrix. In the experiments presented herein, we considered these examples as *unknown*, as in OLINDDA and MINAS. When the examples labeled as *unknown* are used to model extensions or as NPs, they are moved to the corresponding column in the confusion matrix.

The four algorithms used in the experiments have a NP detection procedure based on clustering. For MINAS, this procedure is described in [17], where a NP is composed of a

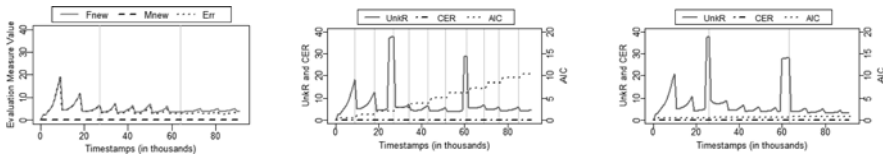
1. The data set SynEDC is available on http://dml.utdallas.edu/Mehedy/index_files/Page675.html

2. We would like to thank to Eduardo Spinosa for providing the source codes.

3. The source code is available in http://www.facom.ufu.br/~elaine/MINAS

4. The executable codes is available in http://dml.utdallas.edu/Mehedy/index_files/Page675.html

5. We would like to thank the authors for providing the executable codes.



(a) Literature Method. (b) Proposed Method. MINAS-V1 (c) Proposed Method. MINAS-V2

Fig. 8. Comparison between evaluation methodologies using MINAS and the MOA data set.

set of clusters. For OLINDDA, every time a new cluster is considered valid and labeled as a novelty concept, it is considered as a new NP. For ECSMiner and CLAM, every time a ND procedure is executed, a set of clusters is generated, which are considered as a new NP.

5.2 Comparison Between the Proposed Methodology and the Methodology in the Literature

In this section, the proposed evaluation methodology is compared with the evaluation methodology found in the literature. This comparison will emphasize the issues covered by the proposed methodology that are not present in the literature methodology. For such, several experiments were performed, which highlight the importance of:

1) the evaluation of the *unknown* examples;
2) the use of model selection;
3) the association of NPs with true classes
4) the use of weighted multiclass classification measures.

The first experiment shows the importance of evaluating the *unknown* examples separately. For such, it applies the MINAS algorithm to the MOA data set. In the results illustrated in Figs. 8a and 8b, the vertical lines in gray represent the timestamps where the algorithm identifies a novelty. To reproduce the methodology found in the literature, all NPs detected by MINAS were summarized in one column in the confusion matrix, the novelty column. The $MNew$, $Fnew$ and Err measures are computed from the matrix (see Equations (1) and (2) respectively).

By following the methodology found in the literature, in the presence of concept drift in the known classes, MINAS initially classifies these examples as *unknown*, thus, increasing the $Fnew$ measure. Next, the *unknown* examples are used to model extensions of the known concepts, decreasing $Fnew$. The $Fnew$ and Err values are determined only by the *unknown* examples.

We believe that the behaviour of MINAS would be more clear if the *unknown* examples were computed separately, which occurs in the proposed methodology. Through the proposed one notes that the classifier did not misclassify any example, because the CER value is zero over the stream. Also, the variations in $UnkR$ help in the understanding of the behaviour of the algorithm. $UnkR$ increases in the presence of examples from the novel classes and concept drift. Finally, it shows that only two NPs were detected, representing the two novel classes that appear in the test set.

The second experiment shows the importance of using model selection techniques to select the best algorithm. Figs. 8b and 8c illustrate the results from the execution of two different versions of MINAS, corresponding to two different values of the threshold parameter, in the MOA data set. In both cases, the value of the CER measure is equal to zero over the stream. However, while MINAS-V1 detected 11 NPs, MINAS-V2 detected only 2. Besides, according to the AIC values, the model produced by MINAS-V2 is also less complex than the model induced by MINAS-V1.

In the third experiment, whose results are illustrated in Fig. 9, shows the effect of associating NPs to the true classes. In these experiments, MINAS was applied to the SynEDC data set. The continuous and dotted lines in Fig. 9 represent the Err measure computed using the methodology found in the literature and the proposed methodology, respectively.

Table 3 shows the confusion matrix obtained, where the rows represent the true classes and the columns the classes predicted by the algorithm. The classes C_1 to C_7 represent the classes used in the training phase. The remaining classes only appear in the test set. The columns NP_1 to NP_9 represent the detected NPs. The column Unk represents the examples marked with the *unknown* profile.

According to Table 3, most of the examples from the known classes are correctly classified. For the novel classes C_9 to C_{12}, MINAS could identify the corresponding NPs and correctly classify most of the examples from these classes. However, most of the examples from the classes C_{14} to C_{18} are classified in the NPs NP_6 and NP_7. Using the literature methodology, the NPs NP_1 to NP_9 are merged into a single column, named novelty column. The examples from the novel classes classified in one of the NPs are computed as hits. The proposed methodology, on the other hand, first associates NPs to true classes followed by the computation of the evaluation measures. Thus, using the proposed methodology, the NP NP_6 will be assigned to the true class C_{14}. The examples from the classes C_{15} to C_{19} classified in NP_6 will be computed as error. This explains the difference between the Err and CER values obtained using the proposed and literature methodologies.

The fourth experiment shows how the predictive performance evaluated for each class, weighted by the number of examples, differs from a general evaluation, not discriminating between the classes. For such, ECSMiner was applied to the KDD data set.

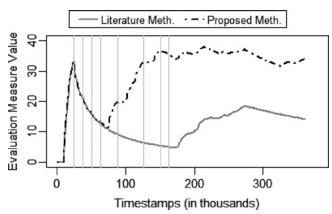


Fig. 9. Comparison between the methodologies using MINAS in the SynEDC data set.

TABLE 3
Final Confusion Matrix Obtained by MINAS in the SynEDC Data Set

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	NP_1	NP_2	NP_3	NP_4	NP_5	NP_6	NP_7	NP_8	NP_9	Unk
C_1	11,966	0	0	0	0	497	0	0	0	0	0	0	0	261	67	61	96
C_2	0	0	0	0	0	80	0	0	0	0	0	0	0	22	11,873	1	456
C_3	845	0	17,764	2,592	147	0	0	0	0	0	0	0	0	0	1,594	0	590
C_4	44	0	0	12,712	6	0	0	0	0	0	0	0	0	0	3,428	0	220
C_5	0	0	0	0	13,749	0	0	0	0	0	0	0	0	109	12	0	82
C_6	0	0	0	0	0	12,142	0	0	0	0	0	0	0	338	0	73	93
C_7	0	0	0	0	0	0	0	0	0	0	0	0	0	12,412	20	15	93
C_8	12,722	0	2,003	1	7	0	0	9,216	0	0	0	0	0	0	10	0	1,159
C_9	2	0	131	1	2	0	0	0	24,821	0	0	0	0	0	5	0	0
C_{10}	0	0	219	226	0	0	0	0	0	24,221	0	0	0	0	0	0	190
C_{11}	0	0	184	0	0	0	0	0	1	121	24,562	0	0	0	0	0	176
C_{12}	0	0	0	0	0	0	0	691	1	2	1	24,031	0	0	0	0	171
C_{13}	0	0	0	0	0	0	0	0	0	0	24,345	1	0	0	0	0	165
C_{14}	0	0	0	0	0	0	0	0	0	138	49	24	22,872	0	0	0	181
C_{15}	0	0	0	0	0	0	0	0	0	0	572	0	19,619	0	0	0	174
C_{16}	0	0	0	0	0	0	0	0	0	0	16	0	12,493	0	0	0	91
C_{17}	0	0	0	0	0	0	0	0	0	0	187	0	303	11,755	0	0	85
C_{18}	0	0	0	0	0	0	0	0	0	0	0	0	247	12,057	0	0	279
C_{19}	0	0	0	0	0	0	0	0	0	0	0	0	207	201	12,058	0	78
C_{20}	0	0	0	0	0	0	0	0	0	0	0	0	1	28	0	12,350	87

Table 4 shows the final confusion matrix for this experiment. In this matrix, the rows represent the true classes and the columns the predicted classes. The training set contains examples from the classes C_1 and C_2. The classes C_3, C_4 and C_5 appear in the columns of the confusion matrix, because ECSMiner has several training phases, allowing the learning of new classes. The columns $NP - C_2$ and $NP - C_4$ represent the set of NPs associated to the classes C_2 and C_4, respectively. The KDD data set illustrates an unbalanced multiclass scenario. The number of examples from the classes C_1 and C_2, (normal class and *dos* attack) is considerable larger than the number of examples for the classes C_3 to C_5 (other types of attack).

In this experiment (see Fig. 10), three different error measures are used, Err, CER, and $AvgError$ (average of the errors per class). The Err measure presents the largest values among these three measures. This happens because it computes a global error measure, not considering the number of examples in each class. The CER measure, on the other hand, assigns a weight to the FP and FN rates. Most of the examples from the novel classes C_3, C_4 and C_5 are misclassified. However, they represent few examples among the total number of examples in this data set, producing low values for CER. The lowest error values were obtained using $AvgError$. This measure is a simple average of the error obtained for each class, not weighting the error. These results show that any of these measures can be used and the choice depends on the problem being treated.

TABLE 4
Final Confusion Matrix Obtained by ECSMiner in KDD Data Set

	C_1	C_2	C_3	C_4	C_5	$NP - C_2$	$NP - C_4$	Unk
C_1	51,662	66	5374	95	22	0	0	72
C_2	1,078	357,369	20,638	89	19	1,932	0	299
C_3	792	3	1	259	0	0	0	10
C_4	784	8	2,033	244	0	4	609	82
C_5	18	0	0	13	8	1	0	7

5.3 Evaluating the Algorithms without External Feedback Using the Proposed Methodology

This set of experiments shows the predictive performance of the OLINDDA and MINAS algorithms without feedback using the proposed methodology. For this data set, while OLINDDA treats ND as a two class task, MINAS treats ND as a multiclass task. In Fig. 11, the vertical lines at the top of each performance diagram show which of the timestamps the algorithm named at the bottom detected as a NP, have in fact, at least one example from a novel class.

The comparison between the results from MINAS and OLINDDA shows that MINAS obtained a CER value better than OLINDDA for this data set (see Figs. 11i and 11j). However, the AIC values increase at the end of the stream, because the number of NPs also increases. However, OLINDDA keeps high CER values while new NPs are detected.

In MINAS results for the SynEDC (see Fig. 11c) data set, it is possible to see that MINAS presented peaks of $UnkR$ whenever examples from the novel classes arrive. For the timestamps smaller than 100,000, short peaks of $UnkR$ are followed by a NP detection, which contributes to the decrease of CER. After the timestamp 100,000, $UnkR$ peaks appear, but there is no NP detection. This may have occurred because the NPs were incorrectly identified as an extension of the known concepts, thus

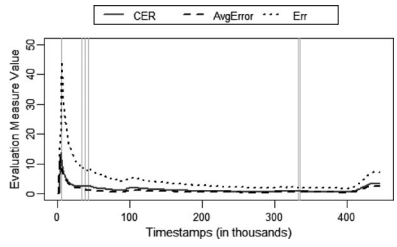


Fig. 10. Comparison among different error measure using ECSMiner, the proposed methodology, and the KDD data set.

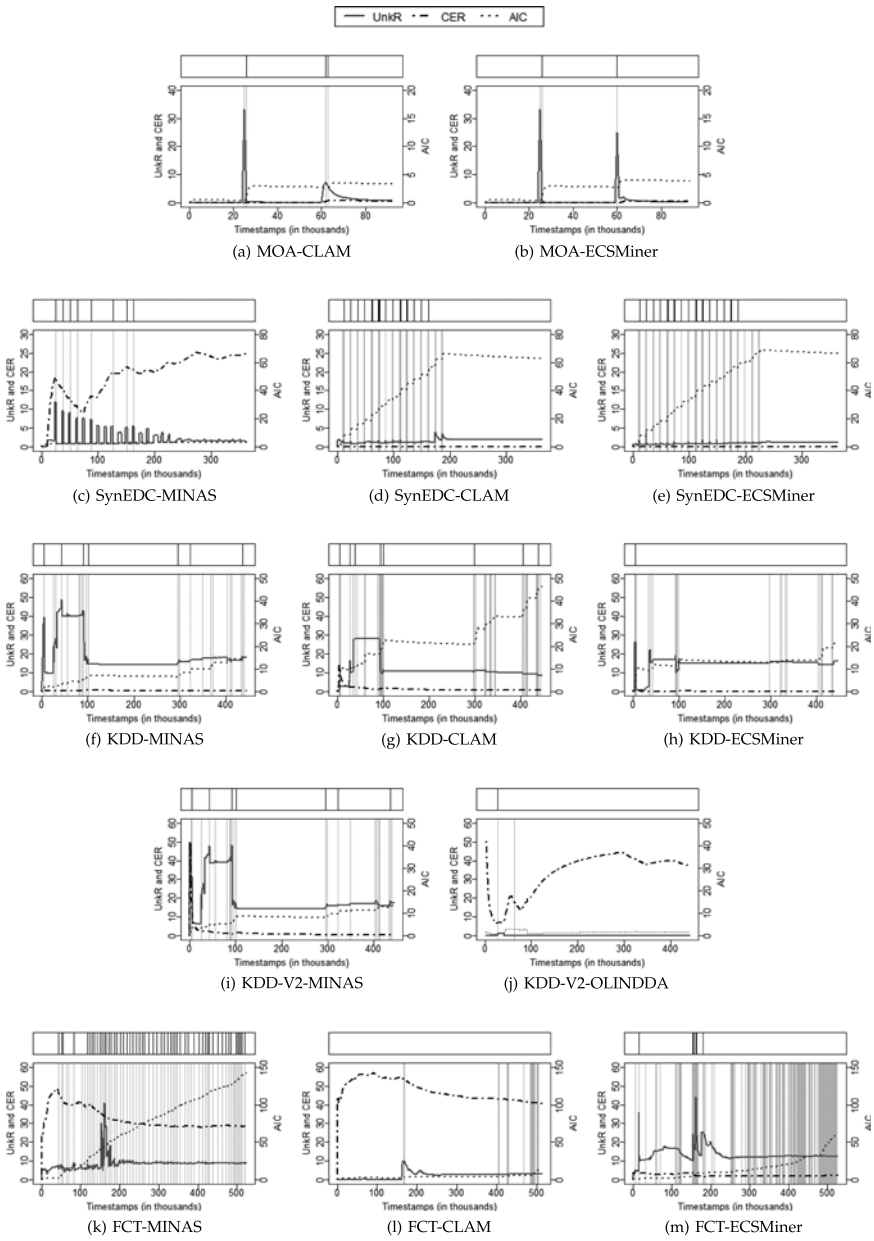


Fig. 11. Performance of the algorithms OLINDDA, MINAS, CLAM, and ECSMiner in different data sets.

increasing CER. In this case, the threshold value could not properly distinguish novelty from extensions. Table 3 corroborates this rationale. However, for a regular number of NPs, AIC is low.

Regarding the KDD data set (see Fig. 11f), although MINAS achieves low CER values, AIC increases over time, due to the increase in the number of NPs over the stream. It is important to highlight that even if MINAS does not use external feedback to update the decision model, it obtains values for CER comparable to CLAM and ECSMiner (see Figs. 11g and 11h), while achieving lower AIC values. In this data set, CLAM and ECSMiner detected more NPs than MINAS.

For the FCT data set, even presenting high $UnkR$ values, MINAS frequently obtains a high value for CER. Besides, the number of NPs, increases over the stream. As a result, the AIC values increase considerably over the stream.

5.4 Evaluating the Algorithms with External Feedback Using the Proposed Methodology

The experiments in this section compare the performance of two algorithms for ND that use external feedback to update the decision model, CLAM and ECSMiner.

The experimental results show that they present a similar performance regarding the CER and $UnkR$ measures for the MOA (see Figs. 11a and 11b), SynEDC (see Figs. 11d and 11e), and KDD (see Figs. 11g and 11h) data sets.

Considering the AIC measure, CLAM obtains a similar performance to that of ECSMiner in MOA and SynEDC. For the KDD, even if both algorithms present similar performance for CER, CLAM obtains higher values for AIC, suggesting that ECSMiner constructs a less complex, and thus preferable, model for this data set.

Regarding the FCT (see Figs. 11l and 11m) data set, even if CLAM obtained the highest values for CER, its number

of NPs is considerably lower, which can explain its low values for AIC.

5.5 Discussion

Several evaluation measures have been proposed in the literature, but mainly for static problems. New evaluation measures are necessary to assess incremental learning tasks in DSs, where new concepts are discovered and the decision model can be updated. These measures should be able not only to properly separate known concepts from novel concepts, but also to verify the hits and errors for each one of the true classes. Since many of these tasks are multiclass, these measures should be able to deal with multiclass classification.

The use of global measures as *Err* may are not be suitable for multiclass tasks, especially in unbalanced scenarios. For example, a situation with many examples from the known classes, few examples from the novel classes, and most of the novel class examples incorrectly classified, low *Err* values would be obtained. Besides, without an association between NPs and classes, a multiclass classification would be treated as a binary task.

Moreover, the evaluation of the examples classified as *unknown* must be carefully considered. In ND systems, even if it is not able to classify some new examples, a classifier should be able to use them to model new concepts. Thus, it is important to verify the variations in the number of *unknown* examples. The evaluation of these *unknown* examples as errors may lead to a poor task modelling.

In a methodology for ND in DSs, the complexity of the induced model needs to be addressed. A good trade-off between the error rate and the number of NPs identified must be found. These two measures can be used together to decide which model is better for a given data set. In this work, we suggested the use of the AIC measure to select the best model for a given data set because it takes into account the classifier error and the number of NPs detected. However, other measures can be used to fulfil this task, such as BIC [47], which gives more weight to the number of detected classes than to the classifier error.

Finally, it is important to understand that the proposed methodology for the evaluation of ND multiclass tasks is not restricted to multiclass classification tasks. It can also be used in one-class classification tasks, since they are a specific case of multiclass classification tasks. In addition, algorithms that update the decision model with or without external feedback can also use them.

6 Conclusions

In this paper, we proposed a new evaluation methodology for ND in DSs. In general, the classification learning algorithms available in the ND literature learn new concepts in an unsupervised fashion, without a matching between the novelties and the classes of the original decision problem. Besides, one class may be represented by one or more NPs and it is necessary to find the matching between NPs and classes. We proposed and experimentally investigated a new methodology for multiclass tasks able to map NPs to the original classes, using a confusion matrix that increases over time.

The experimental results are encouraging, showing that the proposed methodology can be a useful tool to evaluate and compare classifiers developed to multiclass ND in DS. This methodology can be used to evaluate different settings of the same algorithm, which generates different models to represent the same data set, allowing the selection of the more suitable model for a data set.

As future work, we intend to investigate the application of this methodology to other algorithms for ND in DSs. Besides, we intend to investigate strategies to evaluate how fast an algorithm adapts its models to concept drift, which can involve the analysis of different confusion matrices over the stream.

Acknowledgments

This work was supported by Sibila research project (NORTE-07-0124-FEDER-000059), financed by North Portugal Regional Operational Programme (ON.2 O Novo Norte), under the National Strategic Reference Framework (NSRF), through the Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT), and by European Commission through the project MAESTRA (Grant number ICT-2013-612944). The authors acknowledge the support given by CAPES, CNPq, and FAPESP, Brazilian funding agencies.

References

[1] M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 6, pp. 859–874, Jun. 2011.

[2] P. Perner, "Concepts for novelty detection and handling based on a case-based reasoning process scheme," *Eng. Appl. Artif. Intell.*, vol. 22, pp. 86–91, 2008.

[3] J. Gama, *Knowledge Discovery from Data Streams*, 1st ed. Boca Raton, FL, USA: CRC Press, 2010, vol. 1.

[4] E. J. Spinosa, de A. C. P. L. F. de Carvalho, and J. Gama, "Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks," in *Proc. ACM Symp. Appl. Comput.*, 2008, pp. 976–980.

[5] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *The VLDB J.*, vol. 16, no. 4, pp. 507–521, 2007.

[6] J. Zhang, Q. Yan, Y. Zhang, and Z. Huang, "Novel fault class detection based on novelty detection methods," in *Proc. Int. Conf. Intell. Comput. Signal Process. Pattern Recognit.*, 2006, vol. 345, pp. 982–987.

[7] B. S. J. Costa, P. P. Angelov, and L. A. Guedes, "Real-time fault detection using recursive density estimation," *J. Control, Autom. Elect. Syst.*, vol. 25, no. 4, pp. 428–437, 2014.

[8] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 226–235.

[9] M. Hayat, J. Basiri, L. Seyedhossein, and A. Shakery, "Content-based concept drift detection for email spam filtering," in *Proc. 5th Int. Symp. Telecommun.*, 2010, pp. 531–536.

[10] X. Li, "Improving novelty detection for general topics using sentence level information patterns," in *Proc. 15th ACM Int. Conf. Inform. Knowl. Manage.*, 2006, pp. 238–247.

[11] S. Marsland, J. Shapiro, and U. Nehmzow, "A Self-organising network that grows when required," *Neural Netw.*, vol. 15, pp. 1041–1058, 2002.

[12] M. Markou and S. Singh, "Novelty detection: A review part 1: Statistical approaches," *Signal Process.*, vol. 83, no. 12, pp. 2481–2497, 2003.

[13] L. A. Clifton, H. Yin, and Y. Zhang, "Support vector machine in novelty detection for multi-channel combustion data," in *Proc. 3rd Int. Conf. Adv. Neural Netw.-Volume Part III*, 2006, pp. 836–843.

[14] T. Ahmed and M. Coates, "Multivariate online anomaly detection using kernel recursive least squares," in *Proc. IEEE Infocom*, Anchorage, Alaska, 2007, pp. 625–633.
[15] E. J. Spinosa, A. C. P. L. F. Carvalho, and J. Gama, "Novelty detection with application to data streams," *Intell. Data Anal.*, vol. 13, no. 3, pp. 405–422, 2009.
[16] D. M. Farid and C. M. Rahman, "Novel class detection in concept-drifting data stream mining employing decision tree," in *Proc. 7th Int. Conf. Elect. Comput. Eng.*, 2012, pp. 630–633.
[17] E. R. Faria, J. Gama, and A. C. P. L. F. Carvalho, "Novelty detection algorithm for data streams multi-class problems," in *Proc. 28th Symp. Appl. Comput.*, 2013, pp. 795–800.
[18] T. Al-Khateeb, M. M. Masud, L. Khan, C. Aggarwal, J. Han, and B. Thuraisingham, "Stream classification with recurring and novel class detection using class-based ensemble," in *Proc. IEEE 12th Int. Conf. Data Mining*, Washington, DC, USA, 2012, pp. 31–40.
[19] E. Lughofer and P. Angelov, "Handling drifts and shifts in on-line data streams with evolving fuzzy systems," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 2057–2068, 2011.
[20] P. P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *Trans. Fuz Syst.*, vol. 16, no. 6, pp. 1462–1475, 2008.
[21] P. P. Angelov, *Autonomous Learning Systems: From Data Streams to Knowledge in Real-Time*, 1st ed. Hoboken, NJ, USA: Wiley, 2012, vol. 1.
[22] E. R. Faria, I. R. Gonçalves, J. Gama, and A. C. P. L. F. Carvalho, "Evaluation methodology for multiclass novelty detection algorithms," in *Proc. 2nd Brazilian Conf. Intell. Syst.*, 2013, pp. 19–25.
[23] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proc. 29th Conf. Very Large Data Bases*, 2003, pp. 81–92.
[24] M. Z. Hayat and M. R. Hashemi, "A DCT based approach for detecting novelty and concept drift in data streams," in *Proc. Int. Conf. Soft Comput. Pattern Recognit.*, 2010, pp. 373–378.
[25] D. M. J. Tax and R. P. W. Duin, "Growing a multi-class classifier with a reject option," *Pattern Recognit. Lett.*, vol. 29, no. 10, pp. 1565–1570, 2008.
[26] F. Denis, R. Gilleron, and F. Letouzey, "Learning from positive and unlabeled examples," *Theoretical Comput. Sci.*, vol. 348, no. 1, pp. 70–83, 2005.
[27] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Building text classifiers using positive and unlabeled examples," in *Proc. 3rd IEEE Int. Conf. Data Mining*. Washington, DC, USA, 2003, pp. 179–186.
[28] D. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognit.*, vol. 36, pp. 229–243, 2003.
[29] D. Yeung and C. Chow, "Parzen-window network intrusion detectors," in *Proc. 16th Int. Conf. Pattern Recognit.*, 2002, pp. 385–388.
[30] A. Aregui and T. Denœux, "Fusion of One-class classifiers in the belief function framework," in *Proc. 10th Int. Conf. Inform. Fusion*, 2007, pp. 1–8.
[31] M. M. Masud, Q. Chen, L. Khan, C. C. Aggarwal, J. Gao, J. Han, and B. M. Thuraisingham, "Addressing concept-evolution in Concept-drifting data streams," in *Proc. 10th IEEE Int. Conf. Data Mining*, 2010, pp. 929–934.
[32] T. M. Al-Khateeb, M. M. Masud, L. Khan, and B. Thuraisingham, "Cloud guided stream classification using class-based ensemble," in *Proc. IEEE 5th Int. Conf. Comput.* Washington, DC, USA, 2012, pp. 694–701.
[33] I. Pillai, G. Fumera, and F. Roli, "A classification approach with a reject option for multi-label problems," in *Proc. 16th Int. Conf. Image Anal. Process.: Part I*. Berlin, Germany, 2011, pp. 98–107.
[34] M. S. A. Nadeem, J.-D. Zucker, and B. Hanczar, "Accuracy-rejection curves (arcs) for comparing classification methods with a reject option," *J. Mach. Learn. Res.-Proc. Track*, vol. 8, pp. 65–81, 2010.
[35] C. Marrocco, P. Simeone, and F. Tortorella, "A framework for multiclass reject in ecoc classification systems," in *Proc. 15th Scandinavian Conf. Image Anal*, 2007, pp. 313–323.
[36] B. Hanczar and E. R. Dougherty, "Classification with reject option in gene expression data," *Bioinformatics*, vol. 24, no. 17, pp. 1889–1895, 2008.
[37] A. Rosenberg, "Automatic detection and classification of prosodic events," Ph.D. dissertation, Columbia Univ., New York, NY, USA, 2009.
[38] A. Özgür, L. Özgür, and T. Güngör, "Text categorization with class-based and corpus-based keyword selection," in *Proc. 20th Int. Conf. Comput. Inform. Sci.*, 2005, pp. 606–615.
[39] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Res. Logist. Quart*, vol. 2, pp. 83–97, 1955.
[40] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Automatic Control*, vol. 19, no. 6, pp. 716–723, Dec. 1974.
[41] A. Karagrigoriou, K. Mattheou, and I. Vonta, "On asymptotic properties of AIC variants with applications," *Am. Open J. Statist.*, vol. 1, no. 2, pp. 105–109, 2011.
[42] A. Frank and A. Asuncion. (2010). UCI machine learning repository. [Online]. Available: http://archive.ics.uci.edu/ml
[43] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, vol. 1, pp. 281–297.
[44] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Trans. Inform. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.
[45] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.
[46] D. Aha and D. Kibler, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, pp. 37–66, 1991.
[47] G. Schwarz, "Estimating the dimension of a model," *The Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.

Elaine R. Faria received the BS and MS degrees in computer science from the Federal University of Uberlândia, Brazil, and the PhD degree in 2014 from the University of São Paulo. She is a professor of computer science with Computing School, Federal University of Uberlândia, Uberlândia, Minas Gerais, Brazil. Her research interests include data mining and machine learning.

Isabel R. Gonçalves is graduated in system and informatic engineering from the University of Minho and received the MS degree in remote sensing from the University of Porto. She is working towards the PhD degree in applied mathematics at the University of Porto. She is an assistant in mathematics at the Polytechnic Institute of Viana do Castelo. Her research interests include data mining and machine learning.

João Gama received the licenciado degree from the Faculty of Engineering of the University of Porto, Portugal. In 2000, he received the PhD degree in computer science from the Faculty of Sciences of the same University. He joined the Faculty of Economy where he holds the position of an associate professor, He is also a senior researcher at LIAAD, a group belonging to INESC Porto. He has worked in projects and authored papers in areas related to machine learning, data streams, and adaptive learning systems and is a member of the editorial board of international journals in his area of expertise.

Andre C. P. L. F. Carvalho received the BSc and MSc degrees in computer science from the Universidade Federal de Pernambuco, Brazil, and the PhD degree in electronic engineering from the University of Kent, United Kingdom. He is a full professor in the Department of Computer Science, Universidade de São Paulo, Brazil. He has published around 90 Journal and 200 Conference refereed papers. He has been involved in the organization of several conferences and journal special issues. His main interests include machine learning, data mining, bioinformatics, evolutionary computation, bioinspired computing, and hybrid intelligent systems.