

MINAS: multiclass learning algorithm for novelty detection in data streams

Elaine Ribeiro de Faria¹  ·
André Carlos Ponce de Leon Ferreira Carvalho² ·
João Gama³

Received: 10 November 2014 / Accepted: 9 August 2015 / Published online: 22 August 2015
© The Author(s) 2015

Abstract Data stream mining is an emergent research area that aims at extracting knowledge from large amounts of continuously generated data. Novelty detection (ND) is a classification task that assesses if one or a set of examples differ significantly from the previously seen examples. This is an important task for data stream, as new concepts may appear, disappear or evolve over time. Most of the works found in the ND literature presents it as a binary classification task. In several data stream real life problems, ND must be treated as a multiclass task, in which, the known concept is composed by one or more classes and different new classes may appear. This work proposes MINAS, an algorithm for ND in data streams. MINAS deals with ND as a multiclass task. In the initial training phase, MINAS builds a decision model based on a labeled data set. In the online phase, new examples are classified using this model, or marked as *unknown*. Groups of unknown examples can be used later to create valid novelty patterns (NP), which are added to the current model. The decision model is updated as new data come over the stream in order to reflect changes in the known

Responsible editor: Charu Aggarwal.

✉ Elaine Ribeiro de Faria
elaine@ufu.br

André Carlos Ponce de Leon Ferreira Carvalho
andre@icmc.usp.br

João Gama
jgama@fep.up.pt

- ¹ Faculty of Computer Science, Federal University of Uberlândia, Uberlândia, Brazil
- ² Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, Brazil
- ³ Laboratory of Artificial Intelligence and Decision Support (LIAAD), University of Porto, Porto, Portugal

classes and allow the addition of NP. This work also presents a set of experiments carried out comparing MINAS and the main novelty detection algorithms found in the literature, using artificial and real data sets. The experimental results show the potential of the proposed algorithm.

Keywords Novelty detection · Data streams · Multiclass classification · Concept evolution

1 Introduction

Data stream mining is concerned with the extraction of knowledge from large amounts of continuously generated data in a non-stationary environment. Novelty detection (ND), the ability to identify new or unknown situations not experienced before, is an important task for learning systems, especially when data are acquired incrementally (Perner 2008). In data streams (DSs), where new concept can appear, disappear or evolve over time, this is an important issue to be addressed. ND in DSs makes it possible to recognize the novel concepts, which may indicate the appearance of a new concept, a change in known concepts or the presence of noise (Gama 2010).

Algorithms for ND in DSs usually have two phases, initial training phase and ND phase (also named online phase). The first phase is offline and aims to learn a decision model based on labeled examples. The ND phase is concerned with the model application and training with the new unlabeled examples.

Several works consider ND as a one-class classification task, in which the initial training phase learns a model based on examples from one class (the normal class). In the ND phase, examples not explained by the current decision model are classified as novelty. In contrast, several works (Spinosa et al. 2009; Hayat and Hashemi 2010; Al-Khateeb et al. 2012a), instead of identifying a novelty by the presence of only one example not explained by the model, consider clusters of examples not explained by the current model (also named as *unknown* set) to build *novelty patterns* (NP).

However, ND can be viewed as a more general task than a one-class classification task. In ND problems, the known concept concerning the problem (or normal concept) may be composed of different classes, and novel classes may appear over the course of time, resulting in a phenomenon known as concept evolution. Thus, the decision model cannot be static, but it should rather evolve to represent the new emergent classes. Therefore, we understand that ND is a multi-class classification task.

Some works consider that the normal concept can be composed of a set of different classes (Masud et al. 2011; Farid and Rahman 2012; Al-Khateeb et al. 2012a; Liu et al. 2013). However, they consider that only one new NP appears at a time, i.e., examples from two or more different classes cannot appear interchangeably. In addition, these works evolve the decision model assuming that the true label for all instance will be available after Tl timestamps. This is an impractical assumption in several scenarios, since to label all the examples is an expensive and time consuming task.

In this study, we propose a new algorithm, named MINAS (MultiClass learNing Algorithm for DSs), to deal with ND in multiclass DSs. MINAS has five major contributions:

- The use of only one decision model (composed of different clusters) representing the problem classes, learned in the training phase, or in the online phase;
- The use of a set of cohesive and unlabeled examples, not explained by the current model, is used to learn new concepts or extensions of the known concepts, making the decision model dynamic;
- Detection of different NPs and their learning by a decision model, representing therefore a multiclass scenario where the number of classes is not fixed;
- Outliers, isolated examples, are not considered as a NP, because a NP is composed of a cohesive and representative group of examples;
- The decision model is updated without external feedback, or using a small set of labeled examples, even when available.

Here, the MINAS algorithm is compared with other algorithms from the literature using artificial and real data sets. New versions for some of these algorithms are also proposed, in order to verify their performance in scenarios where the class labels are not available.

A preliminary version of the MINAS algorithm was presented in (Faria et al. 2013a). The proposed approach presented in this paper differs from the original version in the following aspects:

- The initial training phase can use the CluStream or K-Means algorithms, producing a set of micro-clusters that are used later to represent the known concept concerning the problem;
- In the online phase, each element of the short-term memory, which stores the examples not explained by the system, has an associated timestamp, allowing the elimination of outdated examples from this memory;
- A new validation criterion is developed for the identification of groups of examples representing NPs as well as new heuristics to automatically calculate the threshold that distinguishes NPs from extensions of the known concepts;
- A new version of the algorithm MINAS using active learning, named MINAS-AL, is proposed;
- Additional algorithms from the literature are used in the experiments and variations of these algorithms are implemented and compared to MINAS;
- A recent and appropriate evaluation methodology from the literature is used to compare MINAS with other algorithms for multiclass classification tasks in DSs, presenting a deeper analysis of the strengths and weaknesses of these algorithms and a complexity analysis of MINAS.

The paper is organized as follows. Section 2 formulates the investigated issue. Section 3 describes the main related algorithms found in the literature. Section 4 describes in details how the MINAS algorithm and its variation using active learning work. Section 5 presents the data sets used in the experiments and the algorithm settings adopted. This section also describes the experiments carried out and discusses the experimental results. Finally, Sect. 6 summarizes the conclusions and points out future works.

2 Formalization of the problem

A DS can be described as an unbounded sequence of continuous generated examples, mostly in high speed, in which the data distribution can change over time.

Definition 1 Data stream: A data stream \mathcal{S} is an infinite set of multi-dimensional examples $\bar{\mathbf{X}}_1, \bar{\mathbf{X}}_2, \dots, \bar{\mathbf{X}}_N, \dots$, arriving at timestamps $\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_N, \dots$. Each example is described by a n -dimensional attribute vector, denoted by $\bar{\mathbf{X}}_i = (x_i^1 \dots x_i^d)$ (Aggarwal et al. 2003).

The ND tasks can be formalized as a decision problem where the target variable takes values in an open set $Y = \{y_1, \dots, y_l\}$. The training set contains labeled examples whose label comes from a subset of Y , e.g. the training set contains observations for some of the classes in Y , but not for all of them. ND learns a function $y = f(x)$ from training and test examples whose label y belongs to Y .

The algorithms for ND in DSs work in two phases, namely initial training phase and ND phase. In the classical view of the ND, the known concept concerning the problem is modelled in the training phase using a labeled data set. Most of the algorithms consider the known concept to be composed only by the *normal class*. In the ND phase, new examples are classified using the decision model or they are rejected (classified as abnormal, anomaly or novelty).

In a more generalist view, ND can be considered as a multiclass classification task. Some works regarded ND as a multiclass task, but they saw the novelty concept as composed of only one class (Farid and Rahman 2012; Masud et al. 2010, 2011; Al-Khateeb et al. 2012a, b). MINAS considers ND as a multiclass classification task, where the known concept can be composed of different classes and different classes can appear interchangeably over time. In a more formal way, we define:

Definition 2 Initial training phase (offline): Represents the training phase, which produces a decision model from a labeled data set. Each example from the training set has a label (y_i), where $y_i \in Y^{tr}$, with $Y^{tr} = \{C_{knw_1}, C_{knw_2}, \dots, C_{knw_L}\}$, where C_{knw_i} represents the i th known class of the problem and L is the number of known classes.

Definition 3 Online phase: Represents the classification of new unlabeled examples in one of the known classes or as *unknown*. As new data arrive, new *novel classes* can be detected, expanding the set of class labels to $Y^{all} = \{C_{knw_1}, C_{knw_2}, \dots, C_{knw_L}, C_{nov_1}, \dots, C_{nov_K}\}$, where C_{nov_i} represents the i th *novel class* and K is the number of *novel classes*, which is previously unknown. This phase is also concerned with the update of the decision model and execution of the ND process.

Definition 4 Novel class: A class that is not available in the initial training phase, appearing only in the online phase. In the literature, the appearance of new classes is also known as concept evolution (Masud et al. 2011).

Definition 5 Unknown set: Examples not explained by the current model. If the classification system is not able to classify a new example, it labels this example

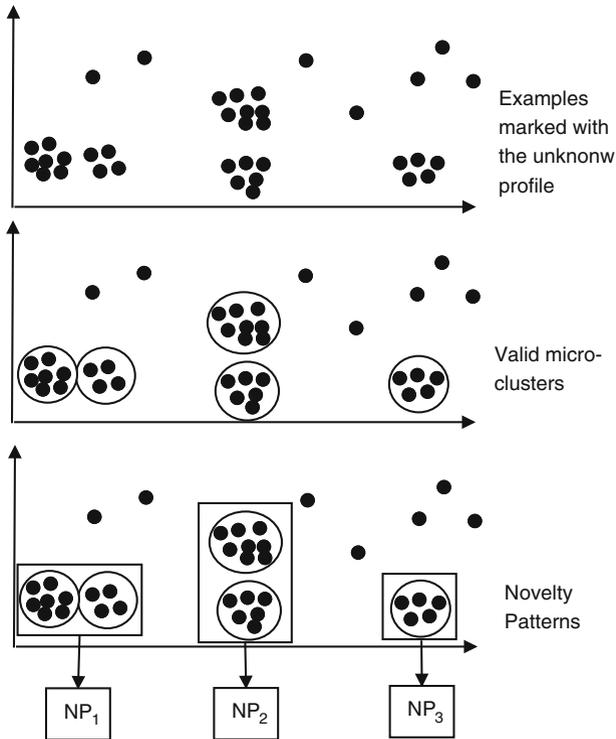


Fig. 1 MINAS procedure to obtain NPs from *unknown* examples

temporarily as *unknown*. In one-class classification, if the classification system is not able to classify a new example, it labels it as abnormal, rejected or anomaly, i.e., the example does not belong to the normal class. In some contexts, this is sufficient. In multiclass classification tasks, the *unknown* examples can be used in the ND process, in the identification of noise or outliers, and in the decision model update.

A common strategy adopted to classify the examples not explained by the current decision model is to label them temporarily as *unknown*. At this time, the adopted strategy is to wait for the arrival of other similar examples before deciding if it represents a noise or outlier, a change in the known concepts, or the appearing of a new concept. When there is a sufficient number of *unknown* examples, they are submitted to a ND procedure, producing different *NP*. In MINAS, the NPs are obtained by executing a clustering algorithm in the *unknown set*. Each obtained cluster represents a *NP*. Figure 1 illustrates the procedure used by MINAS to produce NPs from *unknown* examples (this procedure is detailed in Sect. 4).

Definition 6 Novelty pattern (NP): A pattern identified by groups of similar examples marked as *unknown* by the classification system. A novelty pattern can indicate the presence of a noise or outlier, a change in the known concepts, named concept drift, or a new concept, named concept evolution.

Definition 7 Concept drift: Every instance x_t is generated from a source following a distribution D_t . If for any two examples x_1 and x_2 , with timestamps t_1 and t_2 , $D_1 \neq D_2$, then a concept drift occurs (Farid et al. 2013c).

In order to address concept drift and concept evolution, the decision model needs to be updated constantly. Distinguish between the occurrence of new classes from changes in the known classes is an important issue to be addressed in DS research. In addition, it is also necessary to forget outdated concepts, which is not related with the recent stream activity. Finally, noise or outlier cannot be considered as a NP and must be eliminated, since they represent isolated examples or belong to groups of non-cohesive examples.

3 Related work

The works related to ND in DSs found in the literature belong to three main approaches:

- Works that treat ND as a one-class task (Rusiecki 2012; Krawczyk and Woźniak 2013);
- Works that propose improvements in the one-class classification task (Spinosa et al. 2009; Hayat and Hashemi 2010); and
- Works that consider ND as a multiclass task, but assume that more than one new class cannot appear interchangeably (Masud et al. 2010, 2011; Al-Khateeb et al. 2012a, b).

The algorithms from the first approach induce, in the initial training phase, a decision model using only the known concept concerning the problem, the normal class (Rusiecki 2012; Krawczyk and Woźniak 2013). In their online phase, new examples are classified as belonging to the normal class or as novelty. Another important feature of these algorithms is the constant update of their decision model in order to address concept drift. For such, they assume that the true label of all examples will be available immediately. The main problems with these algorithms are:

- Several real applications are multiclass, where the known concept can be composed of different classes and new classes can appear over time;
- The presence of only one example not explained by the decision model is not sufficient to detect a NP, due to the fact that the data set can contain noise and outliers;
- The true label of all examples is therefore not available.

In order to overcome some of these problems, new algorithms were proposed (Spinosa et al. 2009; Hayat and Hashemi 2010). In these algorithms, the online phase labels as *unknown* the examples not explained by the current decision model and stores them in a short-term memory. When there is a sufficient number of examples in this memory, these examples are clustered. The obtained validated clusters are evaluated as either an extension of the normal class or a novelty. Thus, the decision model is updated by the addition of new clusters obtained from unlabeled data. In these algorithms, the decision model is composed of three submodels, named normal, extension and novelty, each one consisting of a cluster set. The normal submodel represents the known

concept concerning the problem. It is static and created in the initial training phase. The extension submodel represents extensions of the normal submodel. It is created and updated in the online phase. The novelty submodel represents the NPs detected over time. The classification of a new example occurs by finding its closest cluster, from among all clusters of the three submodels. Thus, a new example is classified as normal, extension or novelty based on the submodel of the closest cluster. The main problem with these algorithms is the need to consider ND as a binary classification task, composed of the normal and novelty classes.

The algorithms from the third category see ND as a multiclass classification task (Masud et al. 2010, 2011; Al-Khateeb et al. 2012a, b). In the initial training phase, these algorithms create a decision model using labeled examples from different classes. In the *online* phase, each new example is classified using the current decision model. If the model cannot explain the example, these algorithms wait for the arrival of new similar examples to identify a NP. The decision model is composed of an ensemble of classifiers, created and updated in an offline fashion. The decision model is only updated when a chunk of examples is labeled. In this case, the classifier with the highest error is replaced by a new classifier. The error of each classifier is computed using the set of examples labeled in the last chunk. The main problems with these algorithms are:

- The decision model is updated assuming that the label of all examples is available, and;
- Only one novel class appears at a time. Thus, if examples from two different classes appear interchangeably over time, they will be classified only as novelty, but it will not be possible to distinguish them.

4 MINAS

This section describes the main aspect of the proposed algorithm, named MINAS, which stands for Multiclass learnIng Algorithm for DSs.

4.1 Motivation and contributions

Although several algorithms for ND in DSs have been proposed, they do not address, in a single algorithm, all the issues involved in the problem described in Sect. 2.

The main problem with the previous algorithms is their treating of ND in the DSs issue as a one-class classification task or suppose that all the true labels of the examples will be available for the continuous update of the decision model. Besides, only few of these algorithms include strategies for treating recurring context or address noise and outliers.

As we previously argued, many ND tasks may present more than one novelty class. Thus, they should be treated as multiclass classification tasks.

Thus, the motivation to propose the MINAS arises from the need of an algorithm that addresses ND as a multiclass classification task, which supposes that the true label of all examples will not be available and deals with the presence of recurring contexts. MINAS, as most ND algorithms, presents the following features:

- Is composed of two phases, named initial training and online;
- Has a decision model composed of a set of clusters;
- Marks the examples not explained by the current model as *unknown* and stores them in a short-term memory; and
- Uses clusters of *unknown* examples to update the decision model.

Additionally, MINAS presents several new contributions:

- The use of a single decision model to represent the known classes, learned from labeled examples, and the NPs, learned online from unlabeled examples;
- A ND procedure that identifies and distinguishes different NPs over the stream;
- A ND procedure that identifies extensions of the known concepts, rather than only extensions of the normal class, and distinguishes between an extension and a NP;
- Use of a *memory sleep* to address recurring contexts;
- Removal of old elements from the short-term memory by associating a timestamp to each element of this memory;
- Use of a cluster validation based on the silhouette coefficient;
- Automatic calculation of the threshold employed to separate extensions from NPs;
- Classification of new examples in one of the known classes or in one of the NPs detected over the stream;
- Has a version based on active learning, for scenarios where it is possible to obtain the true label of a few sets of examples.

4.2 Phases of MINAS

4.2.1 Overview

MINAS algorithm is composed of two phases, named initial training and online. Figure 2 illustrates these two phases. The initial training phase, supervised learning phase, induces a decision model based on a labeled data set. It is executed only once. Each class is modeled by a set of micro-clusters. The initial decision model is the union of the micro-cluster sets created for each class.

The main motivation behind creating a classifier composed of a set of micro-clusters is that it can evolve over the stream in a supervised or unsupervised fashion. Micro-clusters are a statistical summary of the data, which present the additive and incremental properties. A micro-cluster can be updated by the addition of a new example or through the merging of two micro-clusters. During the online phase, a new micro-cluster can be created or an outdated micro-cluster can be eliminated (forgotten). Besides, a micro-cluster can be created from labeled and unlabeled data sets.

In the online phase, MINAS receives a set of unlabeled examples and classifies these examples using the current decision model. The examples explained by the decision model can be used to update it. The examples not explained by the model are marked as *unknown*. They are stored in a short-term memory for future analysis. When there is a sufficient number of examples in this memory, they are clustered, generating a set of new micro-clusters. MINAS evaluates each one of the new micro-clusters and removes the non-cohesive or representative ones. The valid micro-clusters are evaluated to decide if they represent an extension of a known class or a NP. In both cases, these

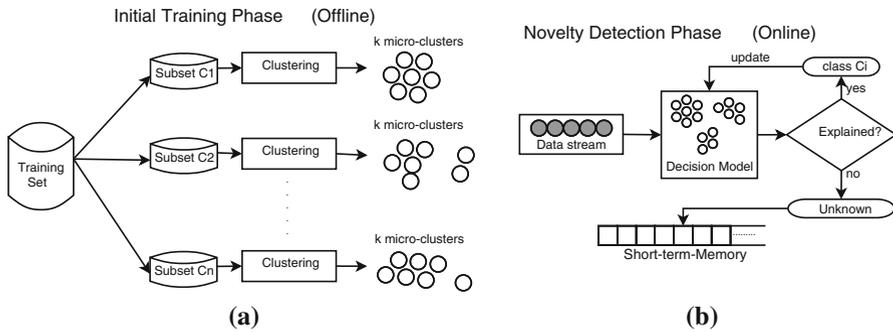


Fig. 2 Overview of the phases **a** initial training and **b** online

micro-clusters are added to the decision model and used in the classification of new examples.

MINAS algorithm also has a mechanism for the forgetting of old micro-clusters and the elimination outdated elements from the short-term memory.

4.2.2 Initial training phase

The initial training phase (offline phase), shown in Fig. 2a, receives as input a labeled data set containing examples from different classes. The Algorithm 1 details this phase. The training data set is divided into subsets, where each subset S_i represents one class of the problem. A clustering algorithm is run on each subset, creating k micro-clusters for each known class. Each micro-cluster is composed of four components: N number of examples, LS linear sum of the examples, SS squared sum of the elements and T timestamp of the arrival of the last example classified in this micro-cluster. Using these measures it is possible to calculate the centroid and radio of a micro-cluster (Zhang et al. 1996).

The clustering algorithms used in this work to create the micro-clusters are K-Means (MacQueen 1967; Lloyd 1982) and CluStream (Aggarwal et al. 2003). Although the K-Means algorithm has a low computational cost, it may not be suitable for large data sets. On the other hand, CluStream is an incremental algorithm developed to work with DSs and has a low computational cost. Thus, it is more adequate for large data sets. The online phase of the CluStream algorithm [(described in (Aggarwal et al. 2003))] is executed on the MINAS training set.

After the execution of the clustering algorithm, each micro-cluster is represented by four components (N , LS , SS and T). Each micro-cluster is labeled to indicate to which class it belongs. Thus, the decision boundary of each class is defined by the union of its k micro-cluster. The initial decision model is composed of the union of the k micro-clusters obtained for each class.

Using the summary statistic of the micro-cluster, measures like centroid and radius can be computed. MINAS uses these measures to classify new examples. For such, it computes the distance between a new example and the closest centroid. If this distance is less than the micro-cluster radius, the example is classified using this micro-

cluster. MINAS calculates the radius of a micro-cluster as the standard deviation of the distance between the examples and the centroid, multiplied by a factor f , as proposed in (Aggarwal et al. 2003).

It is also important to highlight that the decision model generated in this phase can evolve over time, by the insertion of new micro-clusters, removal of outdated ones or the updating of existing micro-clusters, which allows the update of the decision model with a low computational cost. Besides, this structure allows the decision model to evolve using unlabeled examples, i.e., without external feedback.

Algorithm 1 MINAS: Algorithm for the initial training phase

Require: k : number of micro-clusters, alg : clustering algorithm, S : Training Set

```

Model ← ∅
for all (class  $C_i$  in  $S$ ) do
  ModelTmp ← Clustering( $S_{Class=C_i}$ ,  $k$ ,  $alg$ )
  for all (micro-cluster  $micro$  in ModelTmp) do
     $micro.label$  ←  $C_i$ ;
  end for
  Model ← Model ∪ ModelTmp;
end for
return Model

```

4.2.3 Online phase

The online phase, shown in Fig. 2b, is composed of three operations: classify new examples, detect NPs and update the decision model. To perform these operations, MINAS receives as input a stream of unlabeled examples, continuously generating one example per time period. The Algorithm 2 details this phase.

For each new unlabeled example, MINAS verifies if it can be explained through the current decision model. For such, it calculates, using the Euclidean distance, the distance $Dist$ between the new example and the centroid of the closest micro-cluster. If $Dist$ is smaller than the micro-cluster radius, the example is classified using the micro-cluster label. When the example is classified by one of the micro-clusters, the timestamp of this micro-cluster is updated with the timestamp associated to the example. After this operation, the statistic summary of this micro-cluster is updated. Two approaches were investigated for this operation:

- Update the micro-cluster summary when a new example is explained by it,
- Do not update the statistic summary, since this can contribute to increase the classifier error. A comparison between these two approaches is presented in Sect. 5.

When a new example is not explained by any of the micro-clusters of the decision model, it is marked as *unknown* and moved to a short-term-memory, for future analysis. The *unknown* examples are used to model new micro-clusters that can represent NPs or extensions of the known concepts. Marking a new example as *unknown* means that the current decision model does not have sufficient knowledge to classify it. This last case may occur due to one of the three following situations:

- The example is a noise or outlier and needs to be removed;
- The example represents a concept drift; or
- The example represents a NP, which was not learned before.

In the two last situations, it is necessary to have a cohesive and representative set of *unknown* examples to update the decision model in an unsupervised fashion.

Besides, each time a data window is processed, the decision model must forget outdated micro-clusters, i.e., those that do not represent the current state of the DSs. For such, the micro-clusters that did not receive examples in the last time window are moved to a *sleep* memory. A time window represents the most recent data in the stream and it is updated in a sliding window model. The window size is a user-defined parameter, named *windowSize*.

This process is detailed in Sect. 4.3.1. Moreover, the outdated examples from the short-term memory also must be removed, as described in Sect. 4.3.2.

An important characteristic of MINAS is the use of a unified decision model, composed of different classes learned in the initial training phase, and NPs and extensions learned online. Thus, to classify a new example is to assign it the label of a known class or the label associated with an extension or NP.

Algorithm 2 MINAS: Algorithm for the online phase

Require: *Model*: decision model created in the initial training phase, *DS*: data stream, *T*: threshold, *NumExamples*: minimal number of examples to execute a ND procedure, *windowSize*: size of a data window, *alg*: clustering algorithm.

ShortMem $\leftarrow \emptyset$

SleepMem $\leftarrow \emptyset$

for all (example *ex* in *DS*) **do**

(Dist,micro) \leftarrow *closer-micro*(*ex,Model*)

if (*Dist* \leq *radius(micro)*) **then**

ex.class \leftarrow *micro.label*

update-micro(*micro,ex*)

else

ex.class \leftarrow *unknown*

ShortMem \leftarrow *ShortMem* \cup *ex*

if ($|\text{ShortMem}| \geq \text{NumExamples}$) **then**

Model \leftarrow *novelty-detection*(*Model,ShortMem, SleepMem, T, alg*)

end if

end if

CurrentTime \leftarrow *ex.time*

if (*CurrentTime* mod *windowSize* == 0) **then**

Model \leftarrow *move-sleepMem*(*Model,SleepMem,CurrentTime, windowSize*)

ShortMem \leftarrow *remove-oldExamples*(*ShortMem, windowSize*)

end if

end for

- (A) **Extension and novelty pattern detection** Whenever a new example is marked as *unknown*, MINAS verifies if there is a minimal number of examples in the short-term-memory. If so, MINAS executes a ND procedure in an unsupervised fashion, as illustrated in Fig. 3. The Algorithm 3 details this process. The first step is the application of a clustering algorithm on the examples in the short-term-memory, producing *k* new micro-clusters. The K-Means or CluStream algorithms

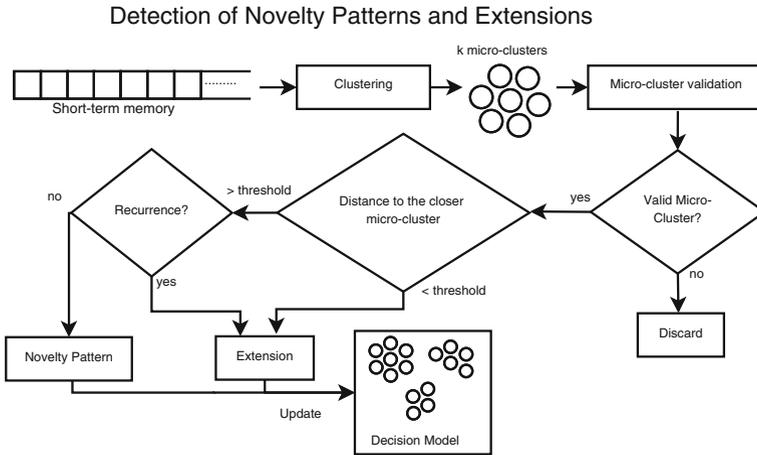


Fig. 3 Overview of ND procedure

Algorithm 3 MINAS: Algorithm for detection of NPs or extensions

```

Require: Model: current decision model, ShortMem: short-term memory, SleepMem: sleep memory,
T: threshold, alg: clustering algorithm
ModelTemp ← Clustering(ShortMem, k, alg)
for all (micro-grupo micro in ModelTemp) do
  if ValidationCriterion(micro) then
    (Dist, microM) ← closest-micro(micro, Model)
    if Dist ≤ T then
      micro.label ← microM.label
    else
      (Dist, microS) ← closest-micro(micro, SleepMem)
      if Dist ≤ T then
        micro.label ← microS.label
      else
        micro.label ← new label
      end if
    end if
    Model ← Model ∪ micro
  end if
end for
return Model
    
```

can be used in this step. MINAS evaluates each one of the micro-clusters to decide if it is valid, i.e., it is cohesive and representative.

A new micro-cluster is cohesive if its silhouette coefficient is larger than 0 (see Eq. 1). For such, MINAS uses a simplified silhouette coefficient (Vendramin et al. 2010). In Eq. 1, *b* represents the Euclidean distance between the centroid of the new micro-cluster and the centroid of its closest micro-cluster, and *a* represents the standard deviation of the distances between the examples of the new micro-cluster and the centroid of the new micro-cluster.

$$Silhouette = \frac{b - a}{\max(b, a)} \quad (1)$$

A new micro-cluster is representative if it contains a minimal number of examples, where this number is a user-defined parameter.

If a new micro-cluster is valid, the next step is to set its label and to add this micro-cluster to the current decision model. In order to decide if it represents an extension of a known concept or a NP, MINAS calculates the distance between the centroid of the new micro-cluster m and the centroid of its closest micro-cluster mp . If this distance is smaller than the threshold T , the new micro-cluster is an extension, and it is labeled with the same label as that of the micro-cluster mp . Otherwise, the micro-cluster is a NP, and a new sequential label (NP_1, NP_2, \dots) is associated with it. In both cases, MINAS updates the decision model adding this new micro-cluster. A discussion about the choice of the threshold T value is presented in the next section.

When a micro-cluster is considered invalid, the short-term memory is not modified, i.e., its examples are maintained for future analysis. However, if these examples remain for a long time in this memory, they can negatively contribute to the decision model update, since they either do not represent the current characteristics of the DS or they represent noise or outliers. Thus, the old examples from the short-term-memory need to be removed. This removal process is discussed in Sect. 4.3.2.

An important characteristic of MINAS is its ability to distinguish among different NPs learned over time. This is possible because MINAS assigns to the same NP similar micro-clusters, but creates new NPs whenever a new detected micro-cluster is distant from the known micro-clusters. In addition, a domain specialist, if available, can label these new micro-clusters providing a fast and incremental decision model update.

(B) Selection of the threshold value to separate novelty patterns from extensions

This work proposes different strategies to select the best threshold value T to distinguish a NP from an extension. The first strategy uses an absolute value, pre-defined by the user. However, the best threshold value varies according to the data set. Besides, in most of the data sets, a single value is not able to separate NPs from extensions. In general, small T values produce many NPs and few extensions. In contrast, large values for T produce many extensions and few novelties. Thus, this strategy works only for artificial data, especially in very well separated spaces.

In order to automatically select the best threshold value, this work proposes three different strategies, named $TV1$, $TV2$, and $TV3$. All strategies first find the Euclidean distance $Dist$ between the centroid of a new micro-cluster m and its closest micro-cluster mp . If $Dist \leq T$, then the new micro-cluster is an extension. Otherwise it is a NP. In $TV1$, T is defined as a cohesiveness measure, computed as the standard deviation of the Euclidean distances between the examples of mp and its centroid, multiplied by a factor f . In $TV2$, T is computed as the maximum Euclidean distance b between the centroid of mp and the centroid of the micro-clusters that also belong to the same class of mp . In

$TV3$, instead the maximum distance, T is computed as the mean distance. In $TV2$ and $TV3$, if mp is the only micro-cluster of its class, T is computed as in $TV1$. The factor f used in $TV1$ was defined empirically to the value 1.1, by executing a set of experiments using different data sets.

4.3 Extensions of the base model

4.3.1 Forgetting mechanism and detection of recurring contexts

During the online phase, the decision model can be updated in two moments. First, when classifying a new example using a micro-cluster, the model can have its statistic summary updated. Second, when a new valid micro-cluster from *unknown* examples is produced, it can be categorized as either an extension or a NP, which is added to the decision model.

The decision model can also be updated by using a forgetting mechanism. When this mechanism is used, micro-clusters that stop contributing to the ND are removed. For such, MINAS stores, for each micro-cluster, a component representing the timestamp of the last example classified. The micro-clusters that do not receive new examples in the last time window are moved to a *sleep* memory (see the last if command in Algorithm 2). The size of a window is defined by a user parameter, named *window size*.

When MINAS detects a new valid micro-cluster, it consults the current decision model to find the closest existing micro-cluster, whose distance is less than T . However, if none is found, MINAS executes the same procedure in the *sleep* memory. If a micro-cluster in the *sleep* memory is found, then the new micro-cluster is marked as an extension and a recurrence of a concept is identified. Next, the micro-cluster of the *sleep* memory is awakened, i.e., removed from the *sleep* memory and added to the current decision model. Otherwise, the new micro-cluster is identified as NP.

The old micro-clusters remain forever in the *sleep* memory, if they are not awakened. The user can define a maximum limit to this memory and thus, when it is full, the oldest micro-cluster is removed. However, the higher this memory, higher will be the computational time of the ND procedure, since it needs to scan this memory. In this work our main objective is to propose a new algorithm that can obtain comparable performance to the state-of-art using few or no labeled examples. In fact, to define a limit to the main and sleep memories is an important issue to be addressed by DS algorithms. But, all of the algorithms suffer from the same problem, and thus this issue was not studied deeply in this work.

4.3.2 Treatment of noise data and outliers

An important aspect to be addressed by ND systems is the treatment of noise data and outliers, which can be confused with the appearance of new concepts.

When a new micro-cluster is obtained by the ND procedure, MINAS applies a validation criterion to eliminate the non-cohesive or representative micro-clusters. When an invalid micro-cluster is obtained, it is discarded, but its examples stay in the short-term memory for future analysis. However, if an example stays in the short-

term memory for a long period, it is removed. The reason is that this example had already participated in at least one clustering procedure and its corresponding micro-cluster was considered invalid. Thus, it becomes a noise or outlier candidate. Another motivation for its removal is the possible long period it has been in the short-term memory, and therefore does not contribute to the current features of the DS.

To remove old examples from the short-term memory, each time a new example is added to this memory, its timestamp is also stored. MINAS frequently verifies the short-term memory to decide upon the removal of outdated examples. This frequency is determined by a user-defined parameter named as *window size*. Thus, at the end of each data window, MINAS verifies if there are outdated examples to be removed. MINAS considers outdated an example whose difference between its timestamp and the current timestamp is higher than the data window size.

4.3.3 MINAS with active learning

MINAS assumes that the true label of the examples is not available, thus updating the decision model without external feedback. However, in several problems, the label of a reduced set of examples may be available after a delay, especially if a specialist is available to manually label some of the examples.

In order to benefit from these labeled examples, we propose a new version of MINAS, named MINAS-AL (MINAS Active Learning). The MINAS-AL algorithm uses active learning techniques to select a representative set of labeled examples to update the decision model.

The initial training and online phases of MINAS-AL and MINAS are very similar. They only differ in the decision model update. MINAS-AL, after reading Tr examples, chooses the centroids of the micro-clusters detected in this interval as the examples to be labeled by the specialist. After the label of a micro-cluster is obtained, the decision model is updated with the new label. If there is no sufficient information to label the micro-clusters, or there is no specialist available, the micro-clusters keep their previous label, represented by the sequential number.

5 Experiments

This section describes the experiments carried out for this study and analyzes the experimental results. In these experiments, MINAS was compared with state-of-the-art ND algorithms found in the literature. Modifications were made in these algorithms to allow a comparison with similar conditions.

5.1 Experimental settings

The experiments use artificial and real data sets. The main features of these data sets are summarized in Table 1. MOA,¹ created using the MOA framework ([Bifet](#)

¹ For details see ([Faria et al. 2013a](#)).

Table 1 Data sets used in the experiments

Data set	#Attributes	#Examples	#Classes	Training classes
MOA	4	100,000	4	0 and 1
SynEDC	40	400,000	20	0, 1, 2, 3, 7, 8 and 9
KDD	34	494,021	5	<i>normal</i> and <i>dos</i>
KDD-V2	34	494,021	5	<i>normal</i>
FCT	54	581,012	7	<i>spruce/fir</i> and <i>lodgepole_pine</i>

et al. 2010) and SynEDC² are artificial data sets. KDD Cup 99 Network Intrusion Detection (KDD) (Frank and Asuncion 2010) and Forest Cover Type (FCT) (Frank and Asuncion 2010) are real data sets frequently used in ND experiments. KDD-V2 is similar to KDD, but the training set contains only examples from the normal class. This data set can be used by algorithms that consider ND as a one-class task, such as the algorithm OLINDDA, proposed in (Spinosa et al. 2009).

In the performed experiments, for each data set, 10% of the data are used in the training phase, with only the examples from the selected classes (see Table 1). The remaining examples are used in the test phase. The order of the examples is the same as in the original data set.

The following algorithms are used in the experiments: OLINDDA³ (Spinosa et al. 2009), ECSSMiner⁴ (Masud et al. 2011), CLAM⁵ (Al-Khateeb et al. 2012a) and MINAS.⁶

Table 2 presents the main parameters of each algorithm from the literature compared with MINAS in this study, and the values adopted. Table 3 presents the main parameters of the MINAS algorithm, and the value settings used in the experiments. The MINAS versions configured with Setting 1, Setting 2, and Setting 3 are denominated here as MINAS-S1, MINAS-S2, and MINAS-S3, respectively.

We have experimentally tested different values to k and we detected the values between 50 and 200 result in a good performance for the MINAS algorithm. When using CluStream and K-Means, we executed the OMRk method in order to select the best number of clusters for the K-Means algorithm. The initial conclusions are that the larger the number of micro-clusters, the larger the time complexity of the algorithm, however larger values of k (> 200) do not improve the classifier performance significantly and $k=100$ presented the best results.

MINAS-AL uses the same parameters as MINAS and it is configured using Setting 1 with $TV1$. In the experiments, we simulated the action of a specialist by considering the label associated with the majority of the examples of a micro-cluster as its label. This is possible because the true label is available for all data sets used in the experiments.

² Available on <http://dml.utdallas.edu/Mehedy/indexfiles/Page675.html>.

³ We would like to thank to Eduardo Spinosa for providing the source codes.

⁴ The executable codes are available in <http://dml.utdallas.edu/Mehedy/indexfiles/Page675.html>.

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

⁶ The source code is available in <http://www.facom.ufu.br/~elaine/MINAS>.

Table 2 Algorithms from the literature used in the experiments

Algorithms	Parameters	Settings
ECSMiner	Window size	2000
	Number of window sizes in initial training phase	10 % of the data/2000
	T_c maximum timestamp to label an example	400
	T_l maximum timestamp to obtain the true label	1000
	Number of classifiers in the ensemble	6
	Number of examples to execute the ND procedure	50
	q: used in the ND procedure	50
	Number of clusters by data window	50
	Classifier parameters	KNN: $k=50$ Decision tree: default values from Weka
CLAM	Idem to ECSMiner	Idem to ECSMiner
OLINDDA	Size of the training set	10 % of the data set
	Size of the short-term-memory	200
	Minimum number of examples in the cluster	3
	Validation criterion	Mean distance between examples and centroid
	Cluster parameters (K-Means)	$k=50$

Table 3 Different configurations of MINAS

Parameters	Setting 1	Setting 2	Setting 3	Setting 4
Number of examples to execute a ND procedure ($\#ExND$)	2000	$\#ExClu * K$	2000	50
Minimum number of examples in the cluster ($\#ExClu$)	$\#ExMem/K$	3	$\#ExMem/K$	30
Window size to forget outdated data	$2*\#ExND$	1000–10,000	$2*\#ExND$	$2*\#ExND$
Threshold	$TV1$ or $TV2$ or $TV3$	$TV1$	$TV1$	$TV1$
Clustering algorithm	Clustream $k=100$	CluStream + KMeans $k=100$	Clustream $k=100$	Clustream $k=50$
Update micro-cluster	No	No	Yes	No

$\#ExMem$ is the current number of examples in the short-term memory

5.2 Evaluation measures and methodologies

In order to evaluate the ND algorithms for DSs, the following issues need to be considered:

- Unsupervised learning, which generates NPs without any association with the problem classes, and where one class may be composed of more than one NP;
- A confusion matrix that increases over time;
- A confusion matrix with a column representing the *unknown* examples, i.e., those not explained by the current model;
- The variation of the evaluation measures over time;
- The problem of classification error rate reduction as the number of NPs increases, i. e., the larger the number of NPs detected, the lower the classifier error.

In order to address all these issues, the evaluation methodology used in this work is the same as that proposed in Faria et al. (2013b). According to this methodology, the first step is to build a square confusion matrix by associating each NP, detected by the algorithm, to a problem class using a bipartite graph. The number of NPs is not equal to problem classes and there is not a direct matching between them.

In order to associate NPs to problem classes, a bipartite graph with vertices X and Y is used, where X represents the NPs detected by the algorithm and Y the problem classes. For each NP NP_i the edge with the highest weight w_{ij} is chosen, where weight represent the number of examples from the class C_j classified in the NP_i .

Following the same strategy adopted in (Faria et al. 2013b), since the NPs were associated to the problem classes, the confusion matrix is evaluated using the combined error measure—CER (see Eq. 2). The last column of this matrix, which represents the examples marked as *unknown*, is evaluated separately using the UnkR measure (see Eq. 3). In Eq. 2, ExC_i is the number of examples from the class C_i , M the number of problem classes, FPR_i false positive rate of the class C_i , FNR_i false negative rate of the class C_i , and Ex the total number of examples. All these terms are computed without considering the *unknown set*. In Eq. 3, Unk_i is the number of examples from the class C_i classified as *unknown*, and ExC_i is the total number of examples from the class C_i .

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

$$UnkR = \frac{1}{M} \sum_{i=1}^M \frac{\#Unk_i}{\#ExC_i} \quad (3)$$

In order to verify the classifier behavior over time, the evaluation methodology proposed in (Faria et al. 2013b) builds a 2D-graphic, where the axis X represents the data timestamps and axis Y represents the values for the evaluation measures. On this graphic, it is important to plot one measure representing the unknown rate in comparison to one or more measures of the accuracy or error rate. Additionally, it is important to highlight on this graphic the detection of a new NP by the algorithm.

When evaluating a classifier, an important issue to be considered, in addition to the error measure, is the complexity of the model. It is important to create models with both low error and low complexity. Considering that the model complexity can be estimated as the number of detected NPs, one can prefer less complex models with low error. Thus, the number of NPs detected over the stream is an important measure to be analysed in the evaluation of a classifier.

5.3 New versions of the algorithms from the literature

To allow a comparison of the ND algorithms from the literature with MINAS in an unsupervised scenario, we created new versions of the algorithms ECSMiner and CLAM, here named ECSMiner-WF and CLAM-WF, to update the decision model without using external label feedback. Both the original and modified versions use the same parameters and they are set as defined in Table 2.

Since ECSMiner-WF and CLAM-WF assume that the true label of the examples will not be available, their decision model is updated without external feedback. For these algorithms, the initial training phase is not modified, i.e., an ensemble of classifiers is created using labeled examples. In the online phase, the examples not explained by the ensemble are marked as *unknown* and stored in a temporary memory. Clusters of *unknown* examples are obtained using the ND procedure described by Masud et al. (2011). Each time a ND procedure is executed, all the clusters obtained receive the same label, which is a sequential number. New examples are classified using initially the ensemble of classifiers. If the ensemble cannot explain the example, the decision model created from the clusters of *unknown* examples is used.

We also analyze how the performance of ECSMiner and CLAM is affected when the parameter T_l is set considering a higher delay to obtain the true label of the examples. For such, the parameter T_l is set to 10, 000. ECSMiner and CLAM with this setting will be named ECSMiner-Delay and CLAM-Delay.

5.4 Evaluation of MINAS in comparison to the algorithms without feedback

This first set of experiments compares MINAS with the algorithms without feedback, ECSMiner-WF, CLAM-WF, and OLINDDA. It uses the data sets described in Table 1 and the evaluation methodology described by Faria et al. (2013b). The ECSMiner-WF-KNN and ECSMiner-WF-Tree algorithms are ECSMiner-WF using the classification learning algorithms KNN and C4.5, respectively.

Figures 4, 5, 6, 7, and 8 illustrate the experimental results using artificial and real data sets. In these figures, the gray vertical lines represent the timestamps where the algorithm identifies a NP. The continuous vertical lines represent the *UnkR* measure and the dotted line the *CER* measure. The gray lines at the top of the performance diagram show which of the timestamps detected as a NP by the algorithm named at the bottom, have, in fact, at least one example from a novel class.

Regarding the MOA data set (Fig. 4), all algorithms present two peaks for *UnkR*. These peaks indicate the timestamps where examples from the two novel classes appear. After each peak, a NP is identified (vertical line), and *UnkR* decreases,

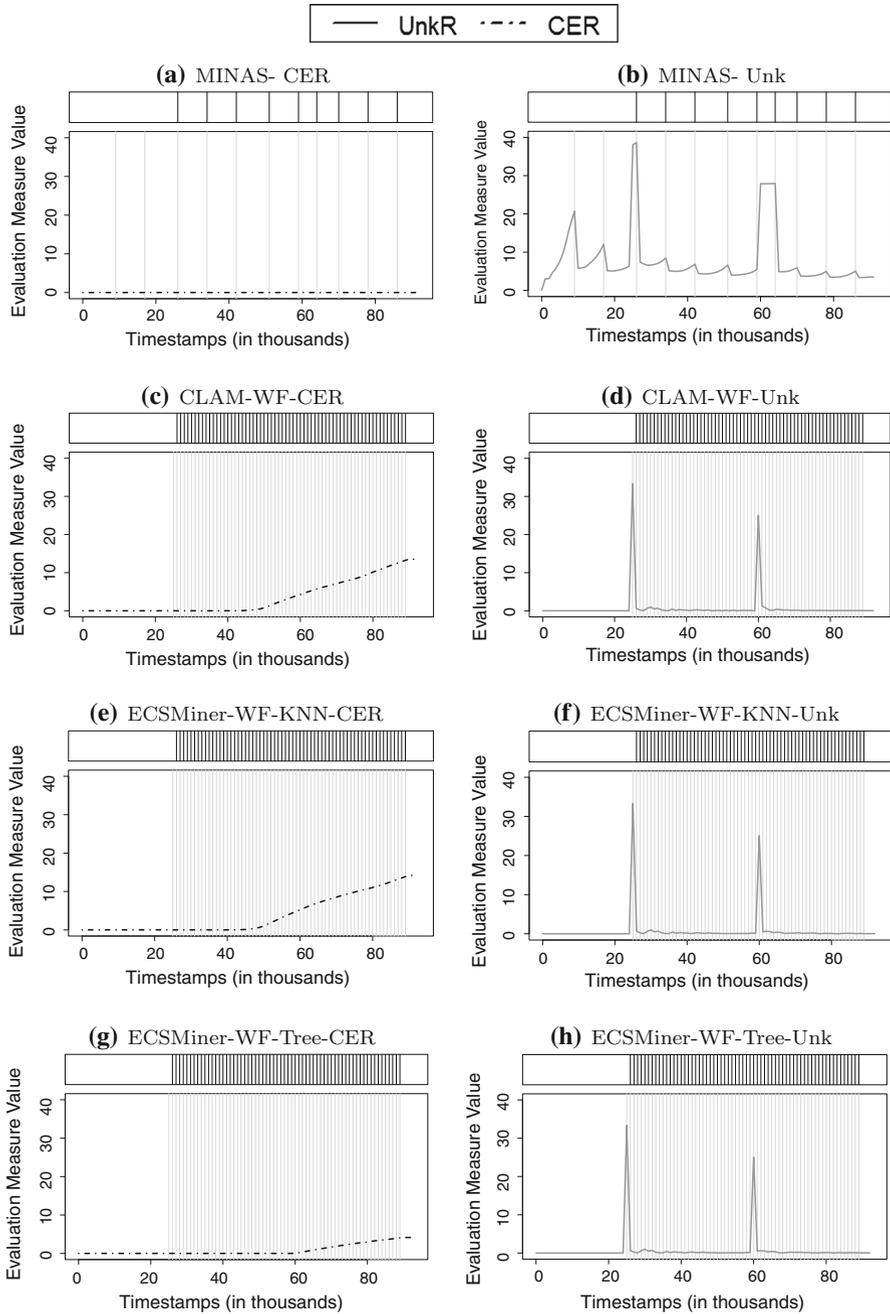


Fig. 4 Performance of MINAS and ND algorithms from the literature in the artificial data set—MOA

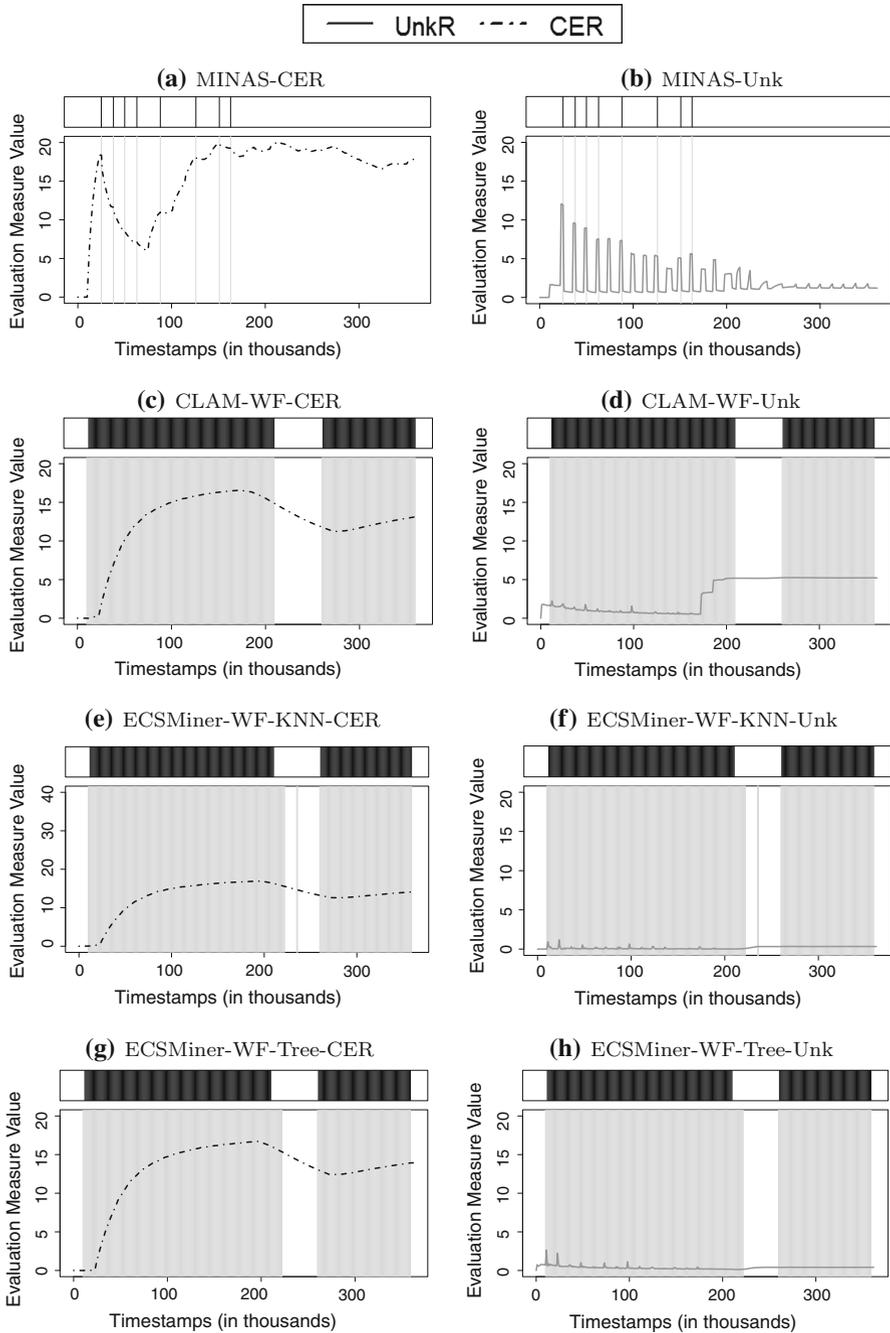


Fig. 5 Performance of MINAS and ND algorithms from the literature in the artificial data set—SynEDC

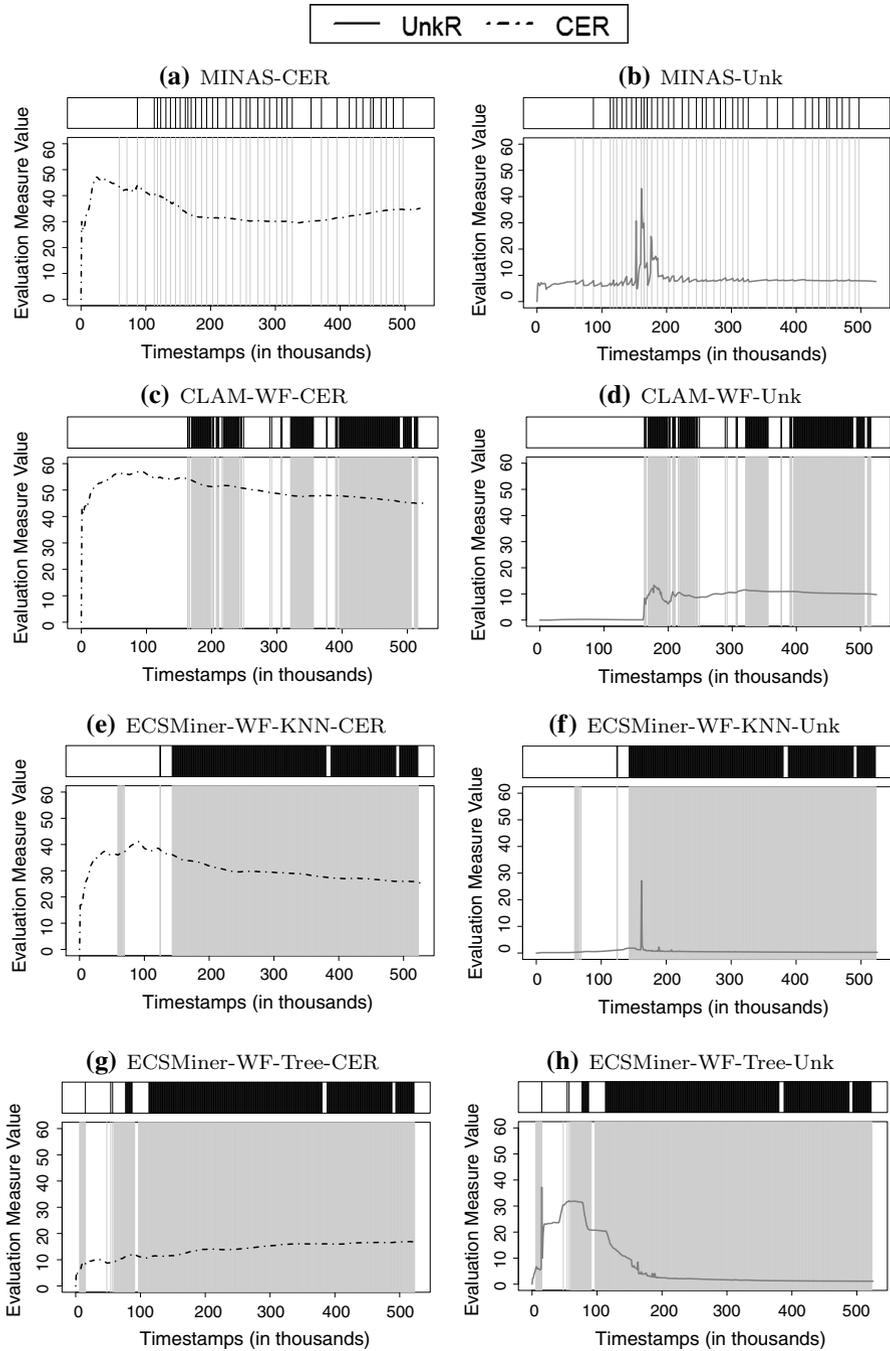


Fig. 6 Performance of MINAS and ND algorithms from the literature in the real data set—FCT

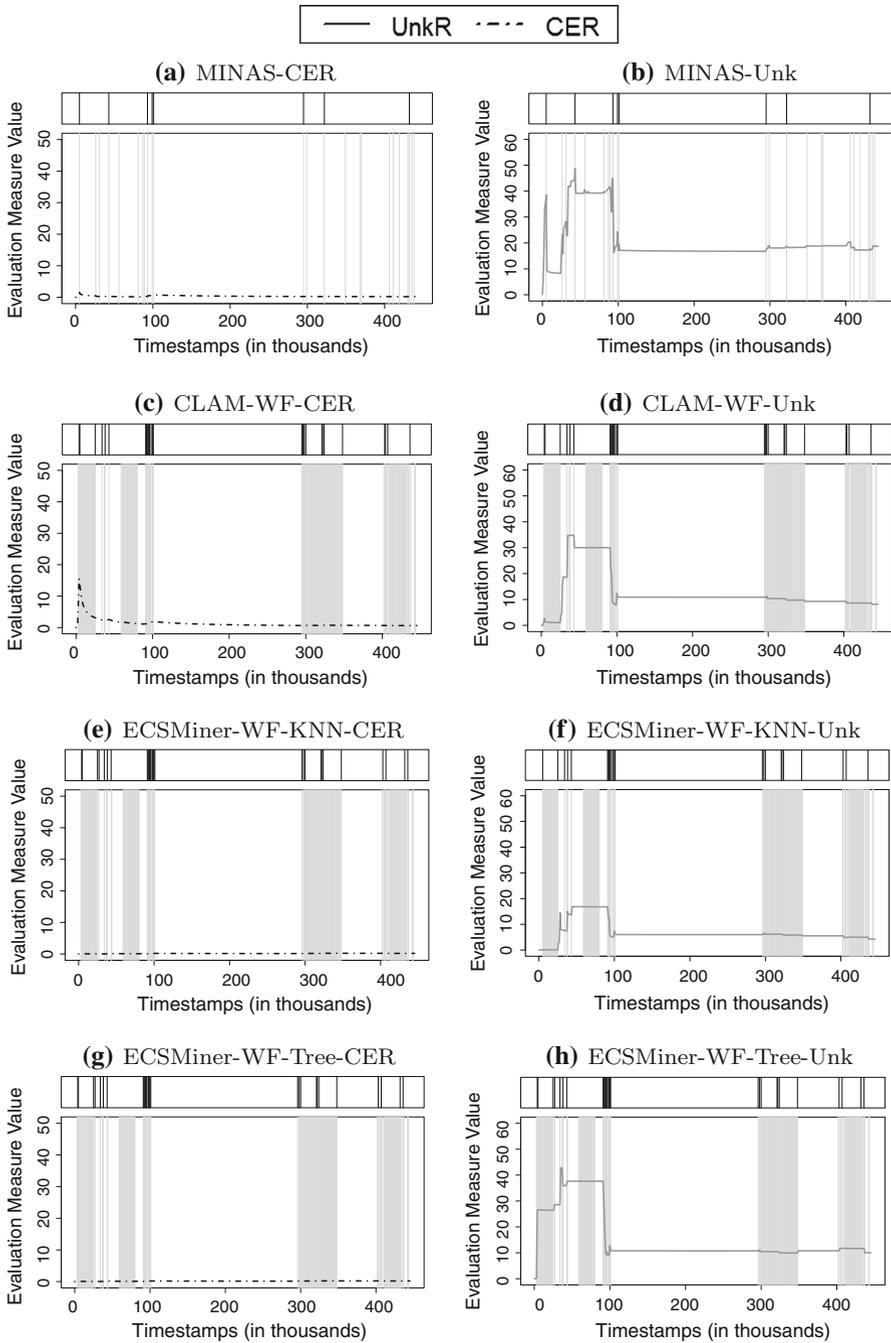


Fig. 7 Performance of MINAS and ND algorithms from the literature in the real data set—KDD

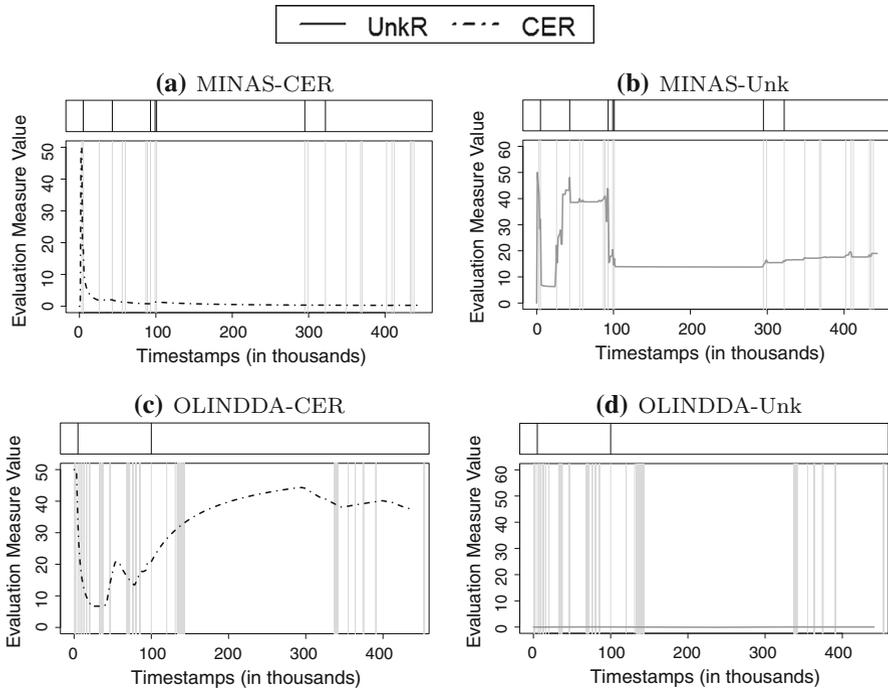


Fig. 8 Performance of MINAS and OLINDDA in the real data set—KDDV2

indicating that these examples make up a NP. For MINAS, the peaks before and after the highest two peaks represent the concept drift present in the known classes. These changes in the known concepts are initially identified as *unknown* examples. These examples will compose clusters representing extensions of the known concepts. Although all algorithms present *CER* equals to zero over the stream, ECSMiner-WF and CLAM-WF identify a larger number of NPs than MINAS. A possible explanation is that ECSMiner-WF and CLAM-WF usually identify a new NP when a ND procedure is executed, while MINAS identifies only when the distance between a new micro-cluster and the closest existing micro-cluster is higher than a threshold. Also, ECSMiner-WF and CLAM-WF present a sudden rise in *CER* around the 50k timestamp. This happens because at this time, a concept drift in one of the known classes is introduced and these algorithms could not identify it. In addition, a new class appears and these algorithms could not identify a NP to represent it.

Regarding the SynEDC data set (Fig. 5), MINAS identified peaks of *UnkR* whenever examples from the novel classes arrive. For those timestamps smaller than 100,000, most of the *UnkR* peaks are followed by a NP identification and a decrease in the *CER*. After this timestamp, peaks of *UnkR* appear, but there is no ND. We identify that the NPs were incorrectly identified as an extension of the known concepts, increasing *CER*, because the threshold value was not able to properly separate novelty from extensions. ECSMiner-WF and CLAM-WF present similar behavior. As the ND procedure is executed at a high frequency in these algorithms, when at least

50 examples are marked with *unknown*, the number of generated NPs is high, which can be seen through the high number of vertical lines in Fig. 5c, e, g. Additionally, as these algorithms do not identify extensions of the known concepts, the examples from the novel classes are always considered NPs, which can explain the low *CER* values.

MINAS, ECSMiner-WF-KNN and CLAM-WF present similar results for the Covertypes data set (Fig. 6). The *CER* is high in the beginning of the DS and decreases over the stream. Although MINAS identified some peaks of *UnkR*, it obtained high values for *CER*. ECSMiner-WF-Tree presents the lowest *CER* values, suggesting that the decision model based on decision trees is a good choice for this data set.

Regarding the KDD data set (Fig. 7), all algorithms present similar behavior, with low *CER* over the stream. MINAS presented the highest *UnkR* values. A possible reason is the more restrictive validation criterion used by MINAS. As a result, these examples remain marked as *unknown* in the short-term memory for a longer period, or they are moved from this memory to create new micro-clusters before they are used. An important characteristic of MINAS is to keep low *CER* values while maintaining the lowest number of NPs.

A new version of KDD, containing only examples from the normal class in the training set, was also used, named KDD-V2 (Fig. 8). This data set was created to compare the predictive performance of MINAS with OLINDDA, which assumes that the training set is composed of only one class. OLINDDA obtains higher values for *CER* than MINAS and presents few examples marked with *unknown*. A possible reason is that the initial decision model, created in the initial training phase, is very specialized to the normal class, classifying every new example from the novel classes as belonging to the normal class. For a better understanding of the OLINDDA results, a second execution of this algorithm was performed, with only 6000 examples in the training set and the remaining in the test set. In this execution, OLINDDA obtained a better performance (but worse than MINAS), with a low *CER* value, showing the occurrence of overfitting in the first execution.

Analysing all these experiments, one can conclude MINAS achieves a similar or better performance than its competitors ECSMiner-WF, CLAM-WF and OLINDDA, considering scenarios where the label of the examples will not be available to update the decision model. In addition, MINAS also identifies less NPs, which results in a less complex model that processes the examples of stream in less time (the time complexity will be discussed in Sect. 5.7). A weak point of MINAS can be identified from the experiments using the SynEDC data set, through the difficulty to distinguish between NPs or extensions. In this data set, the examples not explained by the current decision model are marked as *unknown*, but MINAS falls into this identification if they represent extensions or NPs, increasing the *CER* value.

5.5 Evaluation of MINAS with different settings

This section presents the predictive performance of MINAS under different parameter settings using artificial and real data sets, as shown in Figs. 9, 10, 11, and 12. MINAS is performed using the settings 2, 3 and 4, here named MINAS-S2, MINAS-S3 and MINAS-S4, respectively, as described in Table 3. Besides, MINAS-S1-TV2

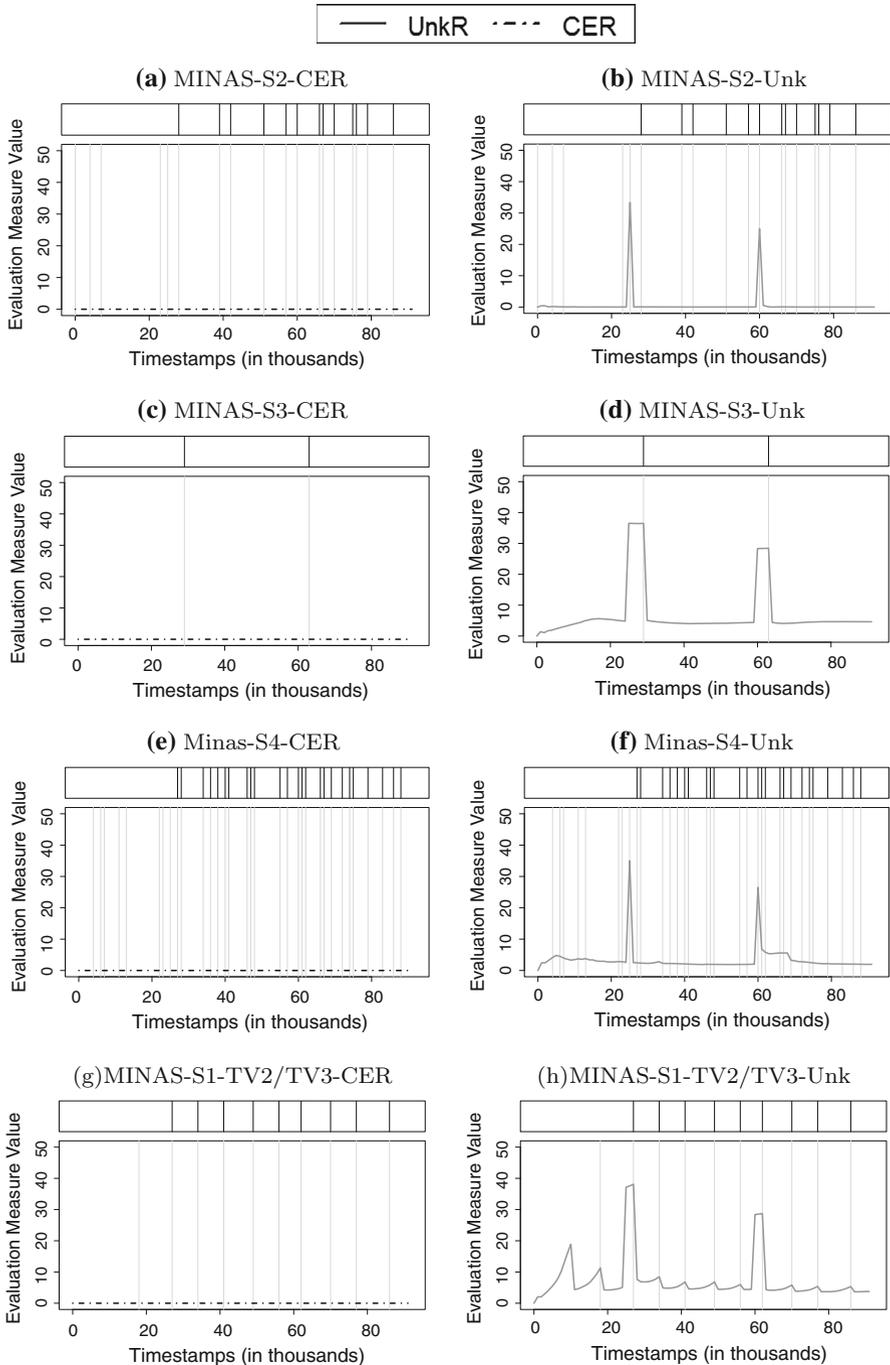


Fig. 9 Performance of different settings to MINAS in the artificial data set—MOA

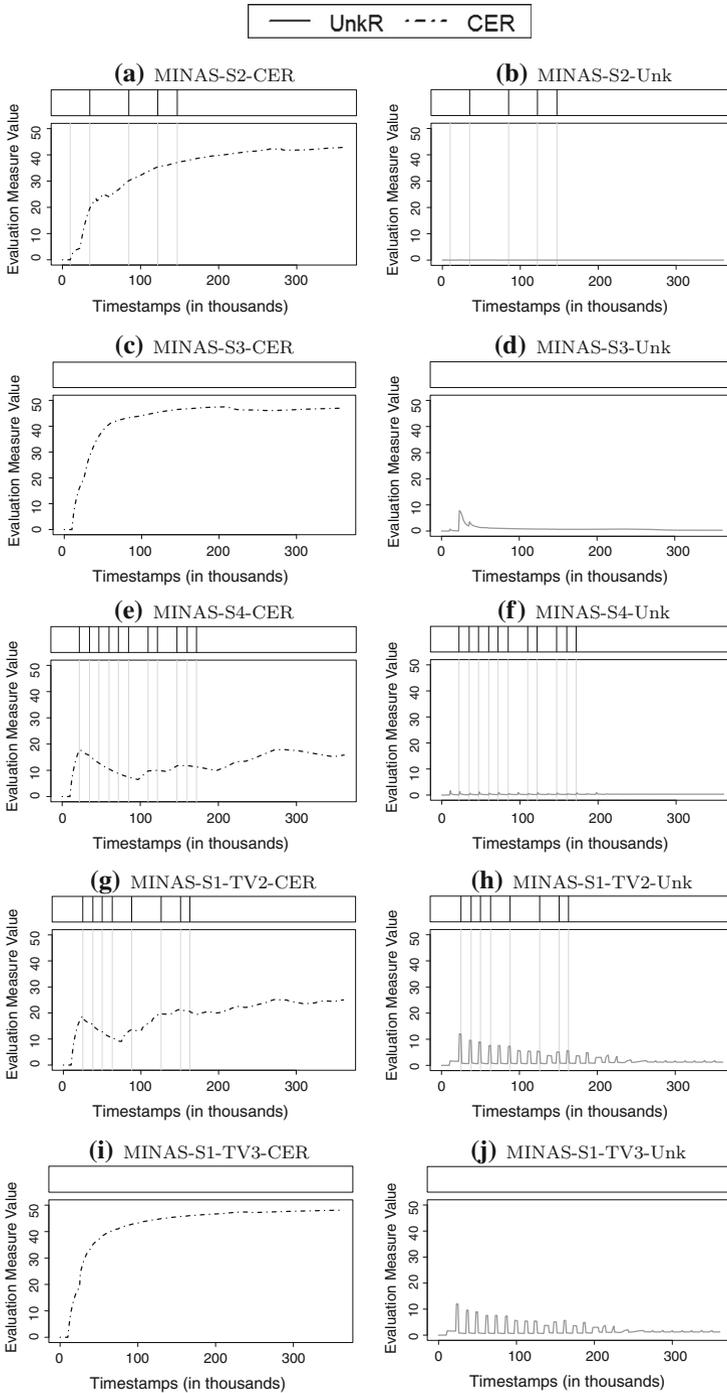


Fig. 10 Performance of different settings to MINAS in the artificial data set—SynEDC

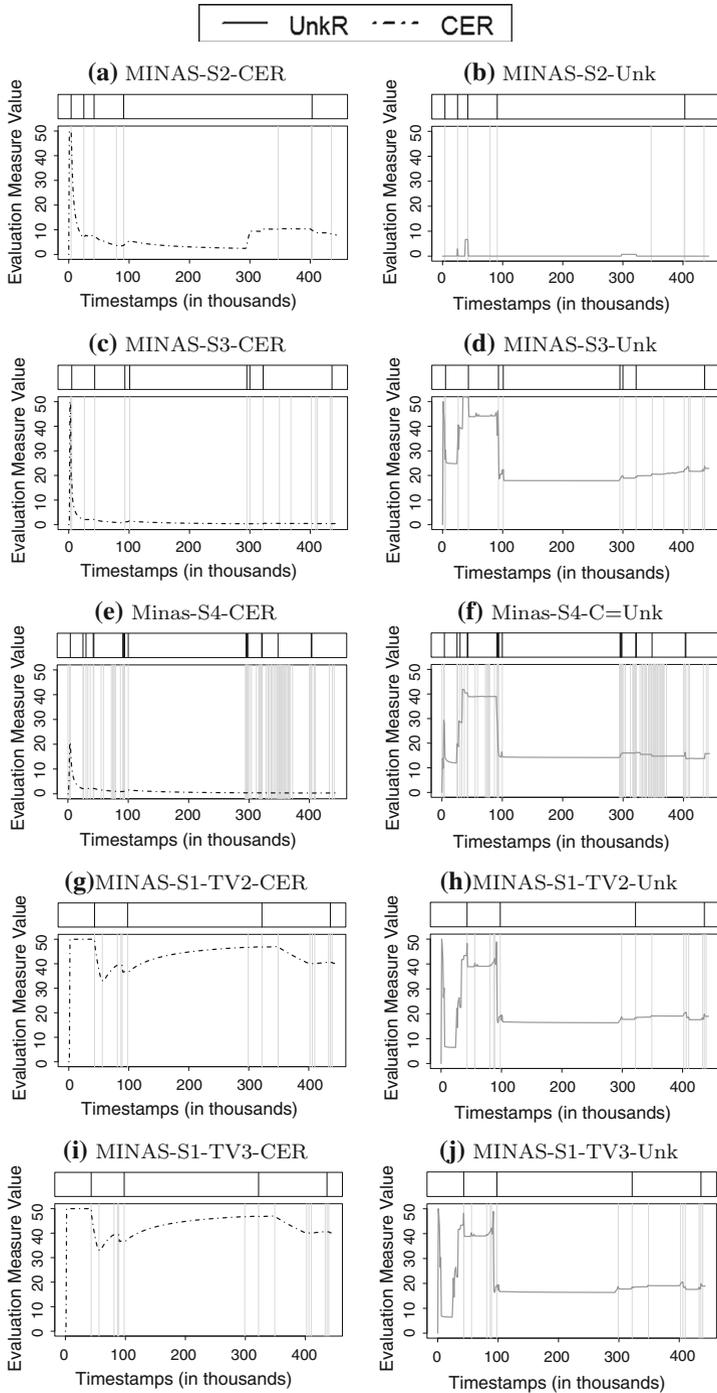


Fig. 11 Performance of different settings to MINAS in the real data sets—KDD

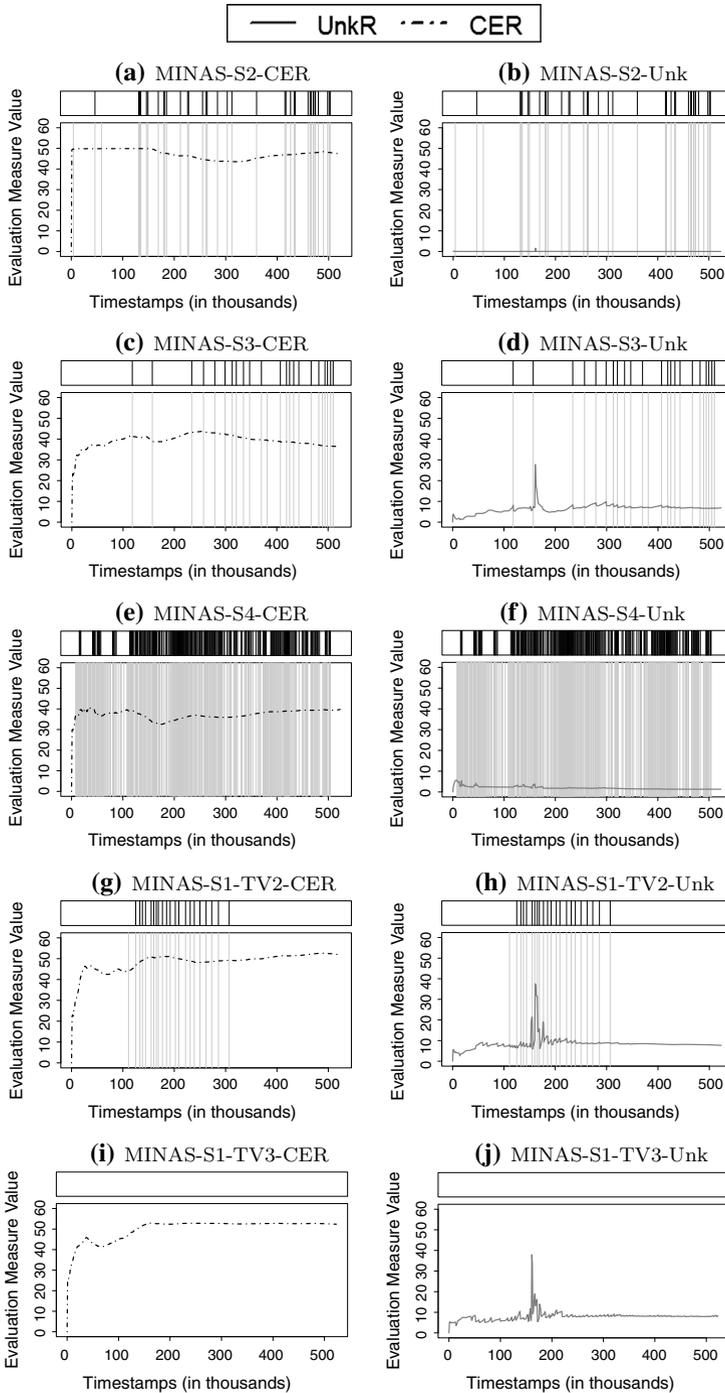


Fig. 12 Performance of different settings to MINAS in the real data sets—FCT

and MINAS-S1-TV3 represent MINAS using the setting 1 and the threshold value set according to $TV2$ and $TV3$, respectively, as described in Sect. 4.2.3.

In the MOA data set (Fig. 9), MINAS presents different values to $UnkR$ for different settings, but it keeps the CER value equal to zero over the stream. For MINAS-S2, as the ND procedure is executed at a high frequency and the minimum number of examples to validate a cluster is small, the changes in the known concepts are rapidly identified as an extension of these concepts, not increasing the number of *unknown* examples. A similar behavior is seen in MINAS-S4. MINAS-S3 presents a different behavior, since extensions are not identified, but only NPs. This happens because the decision model is updated whenever a new micro-cluster classifies an example (center and radius updated). Thus, the decision model is able to deal with constant changes in the known concepts representing them as extensions instead of NPs. MINAS-S1-VL2 and MINAS-S1-VL3 present the same results, which are very similar to MINAS-S1-VL1, presented in Fig. 4a. This shows that the use of different strategies to compute the threshold value did not improve the MINAS performance in the MOA data set.

Regarding the SynEDC data set (Fig. 10), MINAS-S4 obtained the best performance, while MINAS-S2 and MINAS-S3 detected a small number of NPs and low $UnkR$ values. MINAS-S3, which updates the center and radius each time a new example is classified, despite presenting good results for the MOA data set, was not successful with the SynEDC data set. This may have occurred because, in the SynEDC data set, the clusters are very close and some are overlapped. Thus, when MINAS-SF-C3 incorrectly classifies an example, its corresponding micro-cluster is updated, modifying its center and radius incorrectly. For MINAS-S2, the high CER values are explained due to the initial decision model not being adequate for the data. MINAS-S2 first uses the CluStream algorithm producing 100 micro-clusters per class. Next, MINAS-S2 executes the K-Means algorithm, selecting the number of clusters using the OMRk technique (Naldi et al. 2011), which selects $k = 2$ as the best value. Considering different threshold selection strategies, the VL3 approach VL3 was not adequate. Although the algorithm identified peaks of $UnkR$ whenever a new class appears, the new micro-clusters compound by examples from this class were always identified as extension of the known concepts instead of NPs. On the other hand, MINAS-S1-VL2 presents similar results to MINAS-S1-VL1, as shown in Fig. 5a.

Regarding the KDD data set (Fig. 11), MINAS-S2, MINAS-S3 and MINAS-S4 present similar behavior for the CER measure. However, for MINAS-S2, this value increases at the end of the stream. A possible explanation for a low CER value in MINAS-SF-C2, and the consequent increase in the CER at the end of the stream, is the inadequacy of the decision model created in the initial training phase. Again, CluStream, K-Means and OMRk created micro-clusters with high radius, making the separation between the known and novel classes difficult. Comparing MINAS-C3 and MINAS-C4, we can observe that an increase in the frequency of the execution of the ND procedure does not improve the CER values. In addition, MINAS-C3 presents less NPs because it updates the center and radius, constantly. MINAS-S1-TV2 and MINAS-S1-TV3 present the worst results. The new micro-clusters computed from examples of novel classes are incorrectly classified as extensions of the known classes, therefore increasing the CER value.

Regarding the FCT data set (Fig. 12), the different MINAS settings presented high *CER* values. To increase the frequency of execution of the ND procedure, to update the center and radius, constantly and to use different strategies to select the best threshold value did not improve the MINAS performance. The best results were achieved by MINAS-S1, as shown in Fig. 6a.

Considering the artificial and real data sets, the different settings of MINAS did not improve the original obtained results using the setting S1 (see Figs. 4, 5, 6, and 7). The results using MINAS-S2 showed that executing the algorithms CluStream and KMeans produce a worst decision model than from only executing CluStream. MINAS-S3, whose aim is to update a micro-cluster whenever it classifies a new example, also did not present good results. The problem here is if an example is incorrectly classified by a micro-cluster, the characteristics of the micro-cluster are incorrectly updated and this may lead to more errors. MINAS-S4, which executes a ND procedure with a high frequency, produces a larger number of NPs than the other settings of MINAS, but the *CER* values are not decreased. Finally, the different approaches for finding the best threshold to separate extensions of NPs, named TV2 and TV3, did not produce better results than TV1. The development of other approaches to find the best threshold value is an important issue to be treated in future studies.

5.6 Comparison of MINAS-AL in with ND algorithms using feedback

Most of the ND algorithms in DS applications assume that the true label of all examples will be immediately available after its classification, or after a delay of *TI* time units. Using these labels, the decision model can be constantly updated. However, to obtain the true label of new examples is a time consuming task, which requires the presence of a domain specialist and can be impracticable in several scenarios. On the other hand, to ask a specialist the label of part of the examples requires less effort and time and can contribute to improvements in the algorithm performance. This section compares the original versions of CLAM and ECSMiner, which assume that the true label of all examples is available, with MINAS-AL, which uses active learning. In this comparison, we set the *TI* parameter of the ECSMiner and CLAM algorithms to 10,000, to simulate a scenario where the true label of the examples is obtained with a high delay. The algorithms with this setting will be named here ECSMiner-Delay and CLAM-Delay. Figures 13, 14, 15, and 16 show the results of the experimental comparison.

For the MOA data set (Fig. 13), all algorithms presented, in most of the DS, *CER* was equal to 0. The *CER* value for the CLAM-Delay and ECSMiner-Delay algorithms increased at the end of the DS. Besides, CLAM-Delay and ECSMiner-Delay detected a larger number of NPs than in their original versions, CLAM and ECSMiner. These results show that when the true label of the examples is available with a delay, the decision model update is also delayed, degrading the algorithm performance. MINAS-AL showed a behavior similar to MINAS-SF, with *CER* values comparable to ECSMiner and CLAM.

For the SynEDC data set (Fig. 14), the use of external feedback by the CLAM and ECSMiner algorithms to update the decision model produced significantly better

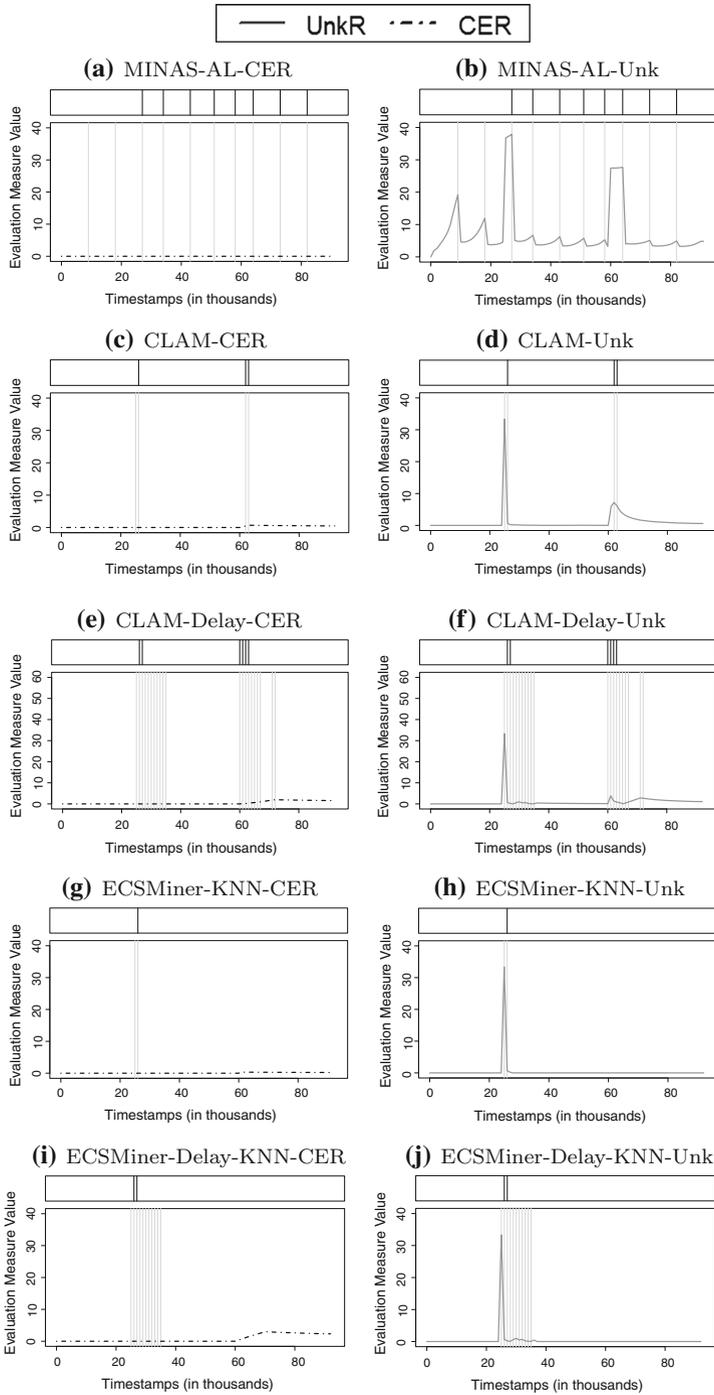


Fig. 13 Performance of MINAS-AL and ND algorithms from the literature using external feedback in the artificial data set—MOA

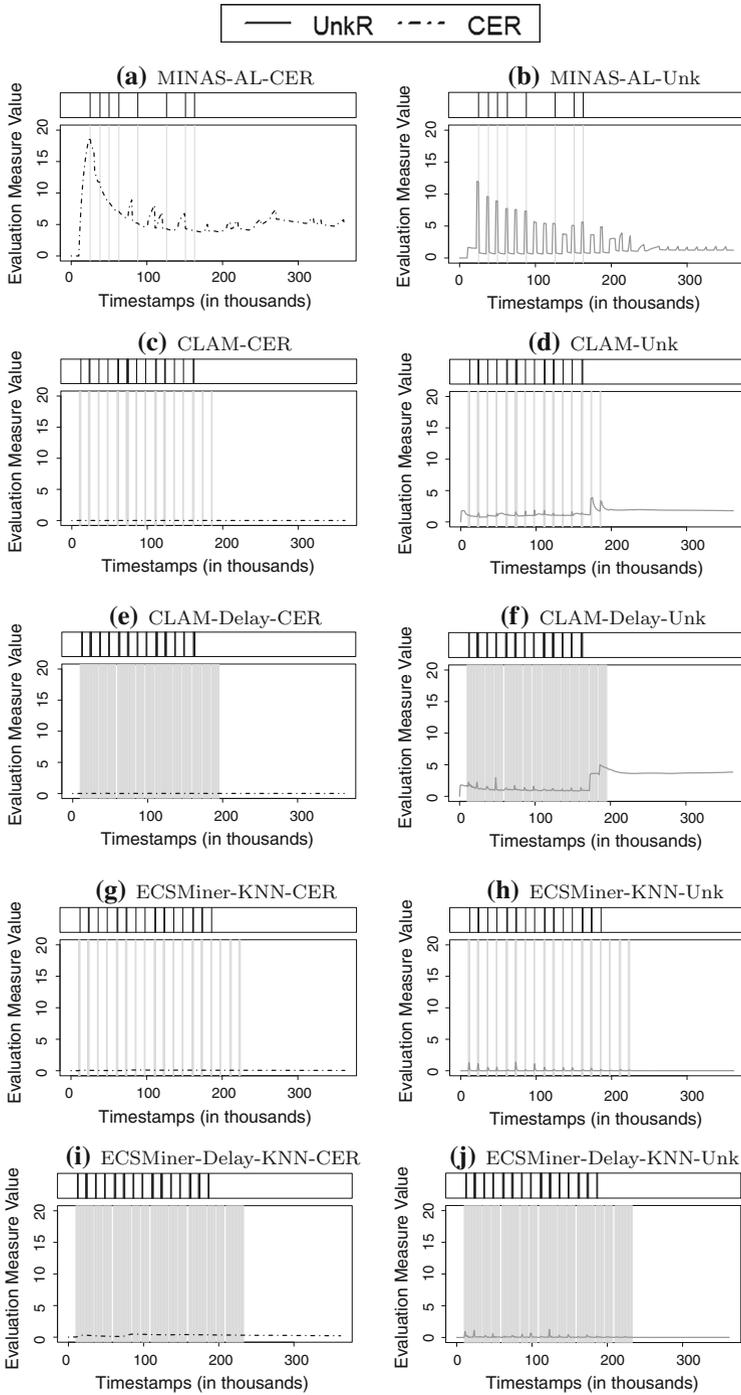


Fig. 14 Performance of MINAS-AL and ND algorithms from the literature using external feedback in the artificial data set—SynEDC

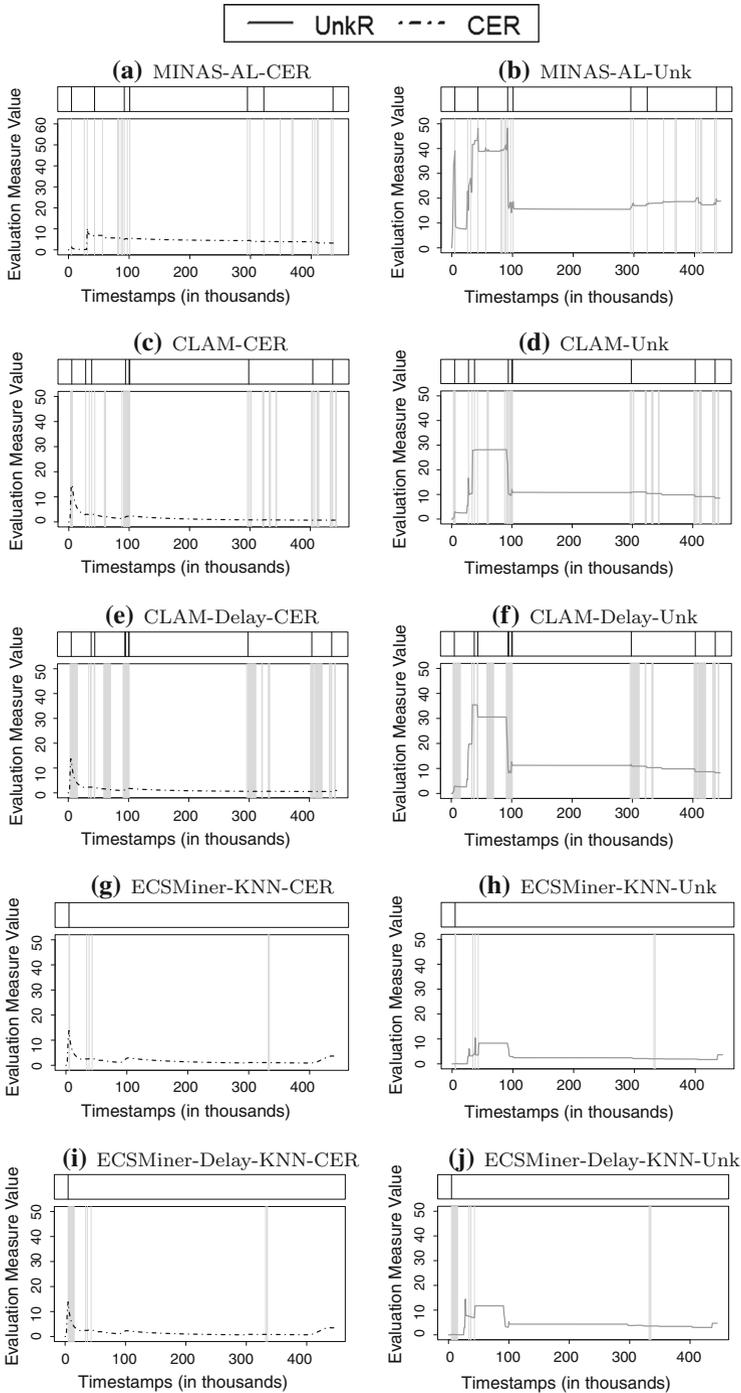


Fig. 15 Performance of MINAS-AL and ND algorithms from the literature with feedback in the real data set—KDD

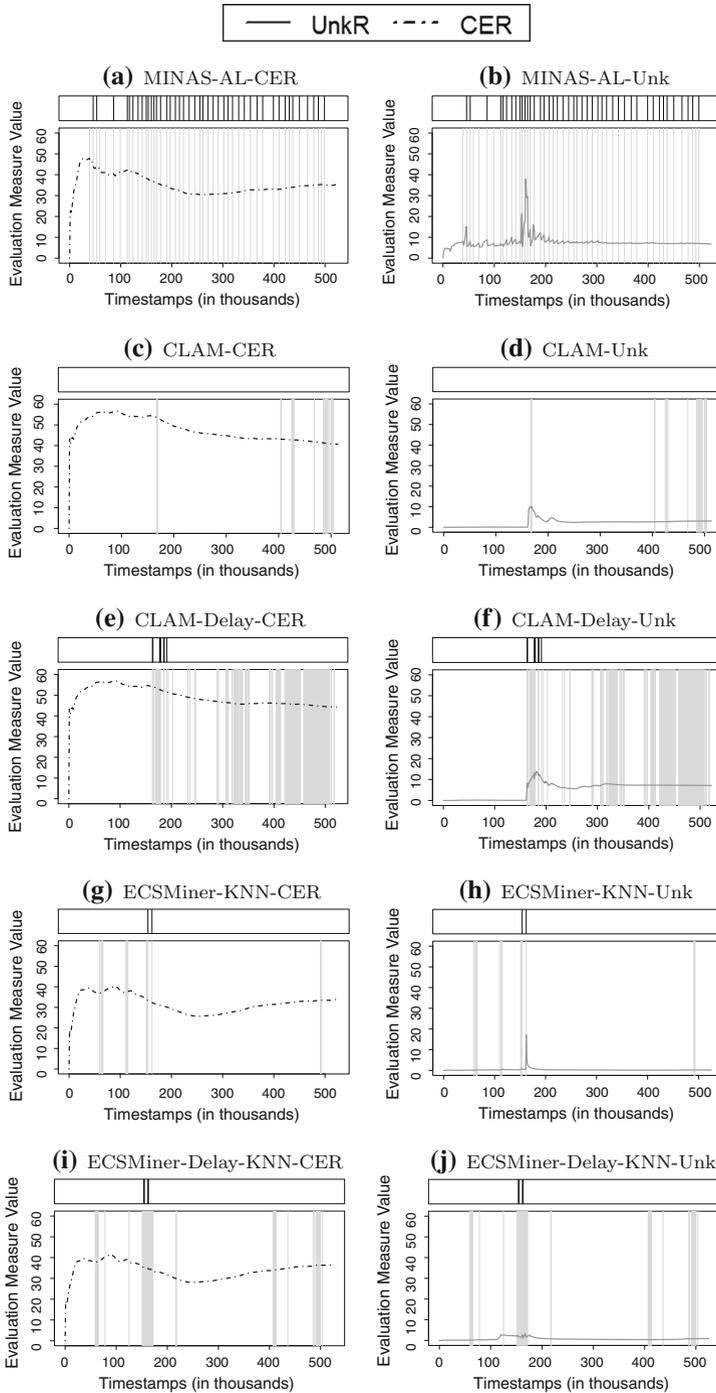


Fig. 16 Performance of MINAS-AL and ND algorithms from the literature with feedback in the real data set—FCT

results than the versions without feedback (see Fig. 5c and e). CLAM-Delay and ECSMiner-Delay considerably increased the number of NPs detected, represented by the vertical lines in the graphics. MINAS-AL obtained better results than the version without feedback (see Fig. 5a), as shown by the decrease of *CER*. It should be noted that MINAS identified less NPs than the other algorithms.

Regarding the KDD data set (Fig. 15), there were no sensible modifications in the *CER* value for the CLAM, ECSMiner, ECSMiner-Delay, CLAM-Delay, ECSMiner-WF (Fig. 7e), and CLAM-WF (Fig. 7c) algorithms. The modified versions detected a higher number of NPs than the original versions. MINAS-AL had a worse performance than its original version. Although this result looks contradictory, since a better performance is expected when the true label of some examples is available, this decrease may be due to the evaluation methodology. In the evaluation methodology used in this work (Faria et al. 2013a), MINAS was evaluated at each 1000 timestamps, associating the NPs detected by the algorithm to the problem class.

For example, consider the following scenario: a NP NP_1 , composed of 50 examples, was created in the timestamp t_1 to represent the examples from the novel class C_{nov1} . From t_1 up to the timestamp $t_1 + \Delta t$, 200 examples from the novel class C_{nov2} appear and are incorrectly classified as belonging to the novelty NP_1 . Thus, MINAS in t_1 associates NP_1 with the novel class C_{nov1} , but in $t_1 + \Delta t$ associates NP_1 with the class C_{nov2} . Using MINAS-AL, after the timestamp t_1 , the specialist will be asked to label the NP NP_1 . In order to simulate the specialist behavior, MINAS-AL will label NP_1 to C_{nov1} , because this is the label of the majority of examples from this cluster. In the timestamp $t_1 + \Delta t$, the NP_1 will not be labeled again and the examples from the C_{nov2} will be incorrectly classified as in the NP_1 , thus increasing the *CER*.

Regarding the FCT data set (Fig. 16), the versions of CLAM and ECSMiner with external feedback did not improve the performance of the versions without feedback (CLAM-WF and ECSMiner-WF), as shown in Figs. 6c, e, respectively. CLAM-Delay and ECSMiner-Delay detected a higher number of NPs than the original versions. MINAS-AL presented lower *CER* values than CLAM and CLAM-Delay and results similar to ECSMiner and ECSMiner-Delay. In this data set, updating the decision model using all the true labels of the examples did not improve the performance. The best results for this data set were obtained by ECSMiner using a decision tree induction algorithm (see Fig. 6g).

Considering the different artificial and real data sets, one observes that MINAS-AL achieves comparable results to ECSMiner and CLAM using only few labeled examples. Here, the main limitation of MINAS-AL is the difficulty to distinguish NPs from extensions, specially in scenarios with a large number of classes as the SynEDC data set.

5.7 Time analysis

In addition to analysing the classifier error and model complexity, another important issue to be considered by ND algorithms for DSs is its time performance.

Figure 17 shows a time comparative analysis among the algorithms without feedback using four different data sets. In all of these, MINAS spent less time to process

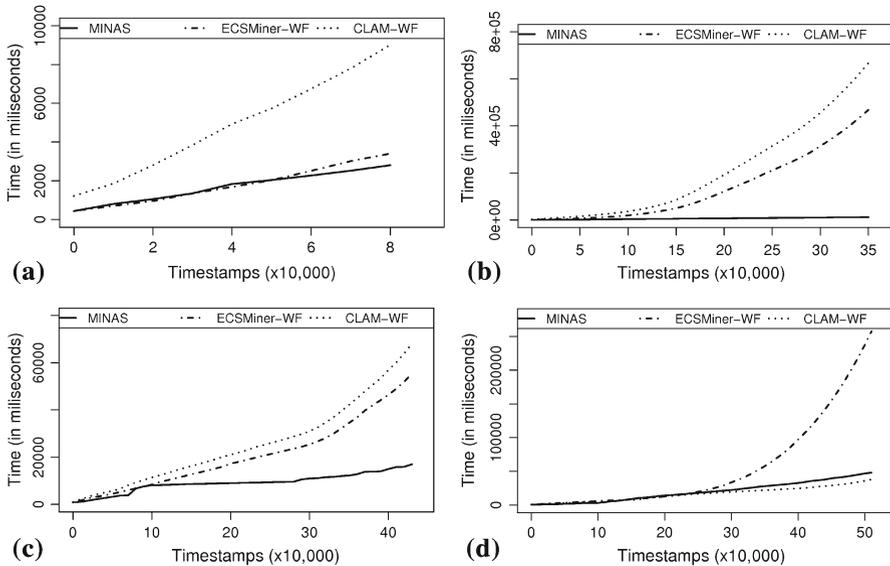


Fig. 17 Time Performance of MINAS-AL and ND algorithms from the literature without feedback in artificial and real data sets

the examples of the stream than the other algorithms, while CLAM-WF is the most costly. A possible motivation to explain the higher time spent by CLAM is that it uses an ensemble of classifiers, one per class, where each classifier is composed of a set of micro-clusters, while ECSMiner-WF uses only one ensemble and MINAS uses a set of micro-clusters as the decision model. ECSMiner-WF as well as CLAM-WF identify a larger number of NPs than MINAS, specially because they execute a ND procedure with a higher frequency. The process for identifying NPs is costly because it executes a clustering algorithm.

Analysing the original versions of ECSMiner and CLAM, which supposes that the true label will be available to update the decision model, the time spent to process the stream is still higher than ECSMiner-WF and CLAM-WF, because they need to update the ensemble of classifiers constantly. In contrast, MINAS-AL, which updates the decision model using only some labeled examples, does not spend more time than MINAS because it only changes the label associated to a NP.

5.8 Critical analysis of MINAS in comparison against state-of-art

This section aims at discussing the main strengths and weakness of the algorithms used in the experiments.

For scenarios, in which the label of the examples is available immediately or after a small delay (for example, 1000 time units), the best performance is achieved using the original versions of ECSMiner and CLAM algorithms. The motivation behind these versions is they update the decision model constantly with supervised information, using labeled examples. Thus, it is possible to train a new classifier with the most

recent characteristics of the stream and replace any which are outdated. But, on the other hand, the constant training of a new classifier and the replacement of outdated ones is a task that takes time. In addition, obtaining the true label of all examples is a time consuming task and can be impracticable in several scenarios. Another important point to be highlighted is as the delay to obtain the true label of the examples increases, the performance of these algorithms also decreases. Thus, as the decision model is not updated, for each time window, new NPs are detected, and the classifier error increases. This happens because in these algorithms, the NPs detected in a time window are not used to classify the examples from a new window, i.e. they are not added to the decision model. The decision model is only updated using labeled examples.

Considering a more realistic scenario, where only few examples of the stream will be labeled after a delay, there is a lack of approaches to address it. MINAS-AL comes as an approach to address this problem. Here, the important point to be highlighted is even MINAS-AL uses only few labeled examples, its performance is comparable to algorithms that update the decision model considering all examples to be labeled. The major weakness of MINAS-AL, as well as the unsupervised version, is the difficulty to automatically identifying a threshold value that can distinguish the different NPs properly, specially when the number of classes is large. In this case, MINAS can identify the examples not explained by the system as *unknown*, builds new NPs using these examples, but fails in identifying if a NP is an extension or a novelty. These results can be viewed in the experiments using the SynEDC data set (see Fig. 5a), which contains 20 classes. On the other hand, a strong point of MINAS-AL is its time complexity $O(1)$ to update the decision model when the label of the examples is obtained, requiring only that the center of the new cluster, which represents a new NP, be labeled. Another important strong point is, if the labels of the examples are obtained after a longer delay, its performance does not decrease. This happens, because the NP detected by the algorithm will be added to the decision model and new examples can be classified using them. However, the NP will not be associated to a class, but they will be named NP_1 , NP_2 , etc.

For a scenario, in which the true label of the examples will not be available and the decision model needs to be updated without feedback, MINAS presents a better performance than the main competitor, OLINDDA. The main limitation of OLINDDA is that the normal concept is composed of only one class, which restricted the experiments to only one data set. In order to better study this scenario, we developed modifications to the original versions of ECSMiner and CLAM algorithms to update the decision model without feedback, i.e., considering that the true label of the examples will not be available. The experiments have shown MINAS achieved performance levels comparable to the main competitors, while maintaining a less complex model. One of the motivations is MINAS executes a ND procedure but only the cohesive and representative clusters are promoted to NPs. Here, again the main limitation of MINAS is the difficulty in distinguish extensions from NPs, which contribute to increasing the classifier error, mainly in the SynEDC data set (see Fig. 5). As the new proposed versions of ECSMiner and CLAM (ECSMiner-WF and CLAM-WF) do not distinguish between concept drift or concept evolution, both will be identified as a new NP, which explains the higher number of NPs identified by these algorithms.

6 Conclusions

This work presented and evaluated a new algorithm for ND in multiclass DSs, named MINAS. MINAS creates a decision model in the initial training phase, composed of a set of micro-clusters, representing the different known classes of the problem. In the online phase, MINAS classifies new examples and detects NPs. In this phase, MINAS updates its decision model without external feedback by adding new micro-clusters, updating existing micro-clusters, or forgetting outdated micro-clusters. MINAS also treats recurring contexts. It does so by maintaining a *sleep* memory, which allows for the forgetting of micro-clusters or recovers them. MINAS presents innovative features, when compared with the existing ND algorithms:

- Updates the decision model without external feedback, or considering a small set of labeled examples;
- Treats ND as a multiclass task, where the normal and novelty concepts can be composed of one or more classes;
- Uses a single model to represent the known classes, extensions and NPs. The experimental results show that MINAS has a predictive performance comparable to the ND algorithms found in the literature, even updating the decision model without external feedback.

This work opens up several perspectives for future works, especially in the development of new strategies to update the decision model when only a small set of labeled examples is available. Other perspectives include the investigation of new strategies to address recurring contexts and decision model updating based on unlabeled examples.

In future steps, we intend on investigating new strategies to automatically select the best threshold value and mechanisms to use active learning. Besides this, we intend on improving the treatment of noise and outliers, as well as recurring contexts.

Acknowledgments This work was partially supported by Sibila and Smartgrids research projects (NORTE-07-0124-FEDER-000056/59), 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 (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo), Brazilian funding agencies.

Appendix: Complexity analysis

The computational cost is an important aspect to be considered in the development of ND algorithms for DSs. One of the requirements for ND algorithms is to execute only one scan in the data. This is very important because the memory is short when compared with the size of the DS.

In MINAS, the initial training phase is batch and run on a small portion of the data set. In this phase, a clustering algorithm is executed for each one of the c known classes, resulting in k micro-clusters per class.

MINAS can use two algorithms in the initial training phase, CluStream and K-Means. Using K-Means, the time complexity for each known class is $O(k \times N \times d \times v)$, where k is the number of micro-clusters, N is the number of examples to be clustered, d is the data dimensionality and v is the maximum number of iterations of K-Means. Using the CluStream algorithm, the first step is to initialize the micro-clusters running K-Means on the first *InitNumber* examples. The next step associates each example to one micro-cluster. The time complexity for the execution of the K-Means for each class is $O(k \times \text{InitNumber} \times d \times v)$. The complexity to include each example (of each class) in its closest micro-cluster is $O(k \times d)$. If the micro-cluster can absorb the example, its statistic summary is updated. Otherwise, the two closest micro-clusters are identified, with complexity $O(k^2 \times d)$, and merged, with time complexity $O(1)$.

In the online phase, whenever a new example arrives, its closest micro-cluster is identified. For such, each one of the q micro-clusters that composes the decision model is consulted, with time complexity $O(q \times d)$. The set of micro-cluster in the decision model, q , is composed by the micro-cluster learned in the initial training phase, k micro-clusters for each known class, plus micro-clusters learned online, the extensions and NPs. Regarding this sum, it is necessary to subtract the micro-clusters moved to the *sleep* memory over time. Although using a large number of micro-clusters allows separability between classes and representation of classes with different shapes, the classification of new examples has a higher computational cost. In addition, the maximal value of q is determined by the memory size.

For the continuous identification of NPs, examples from the short-term memory are clustered using the K-Means or CluStream algorithm, whose time complexity was previously discussed. To identify if a new micro-cluster is an extension or a new NP, its closer micro-cluster is identified, with time complexity $O(q \times d)$. The complexity of the task of move the old micro-clusters to the *sleep* memory is $O(q \times d)$.

References

- Aggarwal CC, Han J, Wang J, Yu PS (2003) A framework for clustering evolving data streams. In: Proceedings of the 29th conference on very large data bases (VLDB'03), pp 81–92
- Al-Khateeb T, Masud MM, Khan L, Aggarwal C, Han J, Thuraisingham B (2012a) Stream classification with recurring and novel class detection using class-based ensemble. In: Proceedings of the IEEE 12th international conference on data mining (ICDM '12), pp 31–40
- Al-Khateeb TM, Masud MM, Khan L, Thuraisingham B (2012b) Cloud guided stream classification using class-based ensemble. In: Proceedings of the 2012 IEEE 5th international conference on computing (CLOUD'12), pp 694–701
- Bifet A, Holmes G, Pfahringer B, Kranen P, Kremer H, Jansen T, Seidl T (2010) MOA: massive online analysis, a framework for stream classification and clustering. *J Mach Learn Res* 11:44–50
- Farid DM, Rahman CM (2012) Novel class detection in concept-drifting data stream mining employing decision tree. In: 7th international conference on electrical computer engineering (ICECE' 2012), pp 630–633
- Faria ER, Gama J, Carvalho ACPLF (2013) Novelty detection algorithm for data streams multi-class problems. In: Proceedings of the 28th symposium on applied computing (SAC'13), pp 795–800
- Faria ER, Goncalves IJCR, Gama J, Carvalho ACPLF (2013) Evaluation methodology for multiclass novelty detection algorithms. In: 2nd Brazilian conference on intelligent systems (BRACIS'13), pp 19–25
- Farid DM, Zhang L, Hossain A, Rahman CM, Strachan R, Sexton G, Dahal K (2013) An adaptive ensemble classifier for mining concept drifting data streams. *Exp Syst Appl* 40(15):5895–5906

- Frank A, Asuncion A (2010) UCI machine learning repository. <http://archive.ics.uci.edu/ml>. Accessed 20 Aug 2015
- Gama J (2010) Knowledge discovery from data streams, vol 1, 1st edn. CRC press chapman hall, Atlanta
- Hayat MZ, Hashemi MR (2010) A DCT based approach for detecting novelty and concept drift in data streams. In: Proceedings of the international conference on soft computing and pattern recognition (SoCPaR), pp 373–378
- Krawczyk B, Woźniak M (2013) Incremental learning and forgetting in one-class classifiers for data streams. In: Proceedings of the 8th international conference on computer recognition systems (CORES' 13), advances in intelligent systems and computing vol 226, pp 319–328
- Liu J, Xu G, Xiao D, Gu L, Niu X (2013) A semi-supervised ensemble approach for mining data streams. *J Comput* 8(11):2873–2879
- Lloyd SP (1982) Least squares quantization in PCM. *IEEE Trans Inf Theory* 28(2):129–137
- MacQueen JB (1967) Some methods of classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley symposium on mathematical statistics and probability, pp 281–297
- Masud M, Gao J, Khan L, Han J, Thuraisingham BM (2011) Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Trans Knowl Data Eng* 23(6):859–874
- Masud MM, Chen Q, Khan L, Aggarwal CC, Gao J, Han J, Thuraisingham BM (2010) Addressing concept-evolution in concept-drifting data streams. In: Proceedings of the 10th IEEE international conference on data mining (ICDM'10), pp 929–934
- Naldi M, Campello R, Hruschka E, Carvalho A (2011) Efficiency issues of evolutionary k-means. *Appl Soft Comput* 11:1938–1952
- Perner P (2008) Concepts for novelty detection and handling based on a case-based reasoning process scheme. *Eng Appl Artif Intell* 22:86–91
- Rusiecki A (2012) Robust neural network for novelty detection on data streams. In: Proceedings of the 11th international conference on artificial intelligence and soft computing—volume part I (ICAISC'12), pp 178–186
- Spinosa EJ, Carvalho ACPLF, Gama J (2009) Novelty detection with application to data streams. *Intell Data Anal* 13(3):405–422
- Vendramin L, Campello R, Hruschka E (2010) Relative clustering validity criteria: a comparative overview. *Stat Anal Data Min* 3:209–235
- Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 103–114