Novelty Detection Algorithm for Data Streams Multi-Class Problems

Elaine R. Faria
University of São Paulo
Fed. University of Uberlândia
São Carlos/Uberlândia - Brazil
elaine@facom.ufu.br

João Gama
LIAAD-INESC Porto
University of Porto
Porto, Portugal
jgama@fep.up.pt

André C. P. L. F. Carvalho
ICMC
University of São Paulo
São Carlos, Brazil
andre@icmc.usp.br

ABSTRACT

Novelty detection has been presented in the literature as one-class problem. In this case, new examples are classified as either belonging to the target class or not. The examples not explained by the model are detected as belonging to a class named novelty. However, novelty detection is much more general, especially in data streams scenarios, where the number of classes might be unknown before learning and new classes can appear any time. In this case, the novelty concept is composed by different classes. This work presents a new algorithm to address novelty detection in data streams multi-class problems, the MINAS algorithm. Moreover, we also present a new experimental methodology to evaluate novelty detection methods in multi-class problems. The data used in the experiments include artificial and real data sets. Experimental results show that MINAS is able to discover novelties in multi-class problems.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Novelty detection, multi-class, data stream, clustering

1. INTRODUCTION

Novelty detection has been presented as a one-class problem where the goal is to discriminate examples from the "Normal" and not "Normal" classes. The learning phase is based only on examples from one class (the normal concept). In the application phase, the stream of unlabeled examples can be classified as either *normal*, belonging to the normal concept learned in the training phase, or *unknown*, not belonging to the normal concept. The *unknown* examples can indicate the presence of a new class, a novelty, which was not learned in the training phase.

We understand that novelty detection is much more general than the one class-problem. In novelty detection problems, the normal concept may be composed by different classes, and novel classes may appear in the course of time, resulting in a concept evolution. Thus, the decision model cannot be static, but it should rather evolve to represent the new emergent classes. Therefore, we understand that novelty detection is a multi-class classification problem.

Data streams are an important scenario for novelty detection techniques. A data stream is a sequence of objects that are continuously produced according to a probability distribution, which can change in the course of time. In this scenario, new concepts may appear and known concepts may change [14]. Since concepts are hardly ever constant, the application of novelty detection in data streams represents an important challenge [7].

In this study, we propose a new algorithm, named MINAS (*MultI-class learNing Algorithm for data Streams*), to deal with novelty detection in data streams multi-class problems. MINAS has five major contributions: i) the decision model that represents the known concept about the problem can address multi-class problems; ii) the use of a set of cohesive examples, not explained by the current model, to learn new concepts or extensions of the known concepts, making the decision model dynamic; iii) detection of different novelties and their learning by a decision model, representing therefore a multi-class scenario where the number of classes is not fixed, iv) the use of only one decision model (composed by different clusters) representing the classes of the problem, either learned in the training phase, or learned in the course of time and v) in the presence of outliers, isolated examples are not considered as a novelty, because a novelty is composed by a cohesive and representative group of examples. We also propose a new experimental methodology to evaluate the novelty detection methods for multi-class problems.

The paper is organized as follows. Section 2 describes the main related works. Section 3 presents the problem formulation and explains the main aspects of the our approach. Section 4 introduces the new experimental methodology for multi-class problems, the experiments carried out and the results obtained for different data sets. Finally, Section 5 summarizes the conclusions and discusses future works.

2. RELATED WORK

Most techniques treat novelty detection as a one-class classification problem [9, 14]. Thus, the decision model is learned using a single concept and new examples are classified as either member of this concept or not [7]. Several techniques were used for one-class-classification in novelty detection like Artificial Neural Networks (ANN) [10], Support

Vector Machine (SVM) [5] and kernel based approaches [2]. The most popular techniques are non-parametric, including Parzen windows [15] and kNN based approaches [4]. These techniques present at least one of the following problems: i) treat the novelty detection as one-class problem; ii) the presence of a single new example, not explained by the current decision model, indicates a novelty; iii) classify a new example as normal or novelty, but only one novelty exists iv) the decision model representing each class is statistic.

One algorithm that tries to overcome some of these problems, OLINDDA (*OnLIne Novelty and Drift Detection Algorithm*) [14], continuously detects and incorporates novel concepts from data streams using data clustering. OLINDDA uses *offline* and *online* learning phases and creates three main models: *normal*, *extension* and *novelty*. Each model is represented by a set of clusters. The normal model is static and built in the *offline* phase. The extension model identifies a small change in the normal concept. The novelty model is composed by examples that are distant from the normal model. In the OLINDDA algorithm, a new novelty is detected using a set of cohesive examples and the normal model can be extended using the extension model. However, OLINDDA is not suited to multi-class problems and only the normal model, learned in the training phase, can be extended over the stream.

Masud et al. [11] proposed ECSMiner, a supervised learning algorithm for data stream classification that integrates novel concepts in a concept drift data stream scenario. An ensemble of classifiers was used and the stream was divided into equal-size chunks, which were used to train a classification model as soon as all instances in a chunk were labeled. The ensemble is continuously updated, replacing the model with the highest error by a new model. The error of each one of the existing models was computed using the set of examples labeled in the last chunk. A new class is detected when all classifiers discover it. This approach addresses novelty detection in data stream multi-class problems. However, a new model is obtained only when all examples in the chunk are labeled. Different from MINAS, which uses an unsupervised learning based approach, ECSMiner uses a supervised learning algorithm. Thus, it is not possible to experimentally compare them. Masud et al. [12] also proposed a system to deal with recurring classes, a special case of concept evolution, where a class may appear and later disappear in the data stream.

Hayat et al. [8] proposed DETECTNOD, an approach for novelty detection that uses a clustering algorithm to model the normal concept. This approach relies on DCT (*Discrete Cosine Transform*) to build a compact representation of the clusters. In an *online* phase, examples that do not belong to the normal concept are considered *unknown*. At the end of a chunk k clusters are obtained from the *unknown* data, using *k-Means*. DETECTNOD compares these clusters with the normal model to detect drifts and novelties, using a threshold value. DETECTNOD has a few limitations. First, there is no validation criterion to identify the cohesion of the clusters obtained from the unlabeled examples in a chunk. Second, only the normal concept can be extended. Third, the clusters that are distant from the normal concept are labeled as novelty, but DETECTNOD does not identify if a set of clusters compose one or more than one novelty (class).

The approach proposed in this work addresses multi-class data stream problems using a unified decision model which represents all the knowledge learned until the current moment composed by the classes learned in the training phase and detected as novelty from the stream. This approach updates this model whenever a new class is discovered or an extension of the known class is identified. Besides, it detects new concepts (or extensions) using a cohesive cluster of examples that are distant from (or near to) all concepts previously learned. This work also proposes a new experimental methodology to evaluate the performance of the algorithm in a multi-class scenario.

3. MINAS ALGORITHM

3.1 Formalization of the problem

Let $D_{tr} = \{(X_1, y_1), (X_2, y_2), ..., (X_m, y_m)\}$ be a training set with m objects, where X_i is the vector of input attributes for the i^{th} examples and y_i is the corresponding target attribute. The class $y_i \in Y^{tr}$, with $Y^{tr} = \{c_1, c_2, .., c_l\}$, where l is the current number of classes. After training, a decision model is created, representing the known concept. As new data arrive, new classes can be detected, expanding the set of class labels to $Y^{all} = \{c_1, c_2, .., c_l, ..., c_k\}$, where the value of k, with $k > l$, is previously unknown.

The data stream classification goal is to classify a new example X_{new} in one of the classes from the set Y^{all}. If $y_{new} \in Y^{tr}$, we have a classical classification problem. Otherwise, a consistent group of examples must be analyzed to identify new classes that can be incorporated to the model and used to classify new examples.

3.2 Overview

This section introduces MINAS, a new approach for novelty detection in data streams multi-class problems. In data stream multi-class problems, the normal concept can be composed by more than one class and different classes can be discovered in the stream. Thus, it is not enough to classify a new example as belonging to the normal concept or the novelty concept, since there can be more than one normal concept class and more than one novelty concept class.

MINAS divides the learning process into two phases: *offline* and *online*. The *offline* phase learns a decision model based on the known concept about the problem. It is executed only once. Next, the *online* phase receives new examples and classifies them either as one of the known classes or as *unknown*. The algorithm also looks for cohesive group of *unknown* examples to detect new classes or extensions of the known classes. The decision model incorporates these new classes and extensions. Moreover, the algorithm also allows the model to forget outdated data and to automatically adapt to concept drift. Figure 1 illustrates the *online* phase. The details of each phase are described as follows.

3.3 Offline phase

The *offline* phase (see Algorithm 1) receives as input a labeled data set containing examples from different classes. This training data set is divided into subsets, each one representing one class of the problem. A clustering algorithm, like *k-Means*, can be used in each subset to create k_{ini} clusters, representing each class.

Although *k-means* algorithm has a low computational cost, it may not be suitable for large data sets. We propose the use of the *CluStream* [1] algorithm for the creation of the clusters representing the classes, which was developed for data

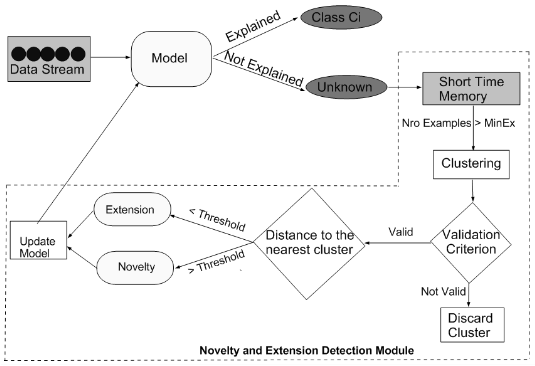


Figure 1: Overview of the online phase

streams clustering and can deal with large data sets. The *CluStream* algorithm stores a statistic summary about data in a structure called *micro-clusters*. These *micro-clusters* are subjected to a clustering process (the authors suggest the use of *k-Means*) in order to create the final clusters. In this work, we use *CluStream* to select, for each subset of the training set, a group of representative elements, the *micro-cluster* centroids, and apply the *k-Means* algorithm to them obtaining k_{ini} clusters for each class.

In the *offline* phase, the number of clusters (k_{ini})for each training set class is automatically detected using the *OMRk* method [13]. The initial model is composed by the union of the k_{ini} clusters obtained for each class. As *k-Means* finds spherical clusters, this work propose to represent each hypersphere by a center, a radius and a label (class associated with the hypersphere).

Algorithm 1 MINAS: Offline Phase

```
Require: $k_{ini}$, $algorithm$, $TrainingSet$
1: for all classes $C_i$ in $TrainingSet$ do
2:     $ModelTmp \leftarrow$ Clustering($TrainingSet_{Class=C_i}$, $k_{ini}$, $algorithm$)
3:     for all hyperspheres $h$ in $ModelTmp$ do
4:         $Label_h \leftarrow C_i$
5:     end for
6:     $Model \leftarrow Model \cup ModelTmp$
7: end for
8: return $Model$
```

3.4 Online phase

The *online* phase (see Algorithm 2 and 3) receives as input an unlabeled data stream. MINAS checks each new example to verify if it can be explained by the current model. If the Euclidean distance between the new example and the centroid of any hypersphere is less than the radius of this hypersphere, then the example is considered to be covered by this hypersphere and classified with the hypersphere label (step 3 and 4, Algorithm 1). Otherwise, the example is marked with the *unknown* profile and moved to a *short-time memory* for further analysis (steps 6 and 7, Algorithm 1).

MINAS classifies a new example as either *unknown*, not explained by the current model, or as *belonging to a class*, which can be learned in the *offline* or *online* phase. Thus, each hypersphere has a label. If the hypersphere is learned in the *offline* phase, it receives a concept class label. For hyperspheres learned in the *online* phase, it is necessary to choose between two options. If the hypersphere represents an extension of a known class, then its label is the same label of the extended class. If the hypersphere represents a novelty, then a new label must be created.

Algorithm 2 MINAS: Online Phase

```
Require: $T$, $Model$, $Stream$, $NumMinExamples$
1: $ShortTimeMem \leftarrow \emptyset$
2: for all $elem$ in $Stream$ do
3:     if $elem$ is inside an hypersphere $h$ of $Model$ then
4:         Classify $elem$ in Class $Label_h$
5:     else
6:         Classify $elem$ as $Unk$
7:         $ShortTimeMem \leftarrow ShortTimeMem \cup elem$
8:         if $|ShortTimeMem| \geq NumMinExamples$ then
9:             $Model \leftarrow$ Novelty-Extension-Detection ($Model$, $T$, $ShortTimeMem$)
10:        end if
11:    end if
12: end for
```

3.5 Identifying novel concepts and determining the nature of new concepts

Every time a new example is labeled as *unknown*, it is necessary to check if a minimum number of unknown examples has already been found. If so, MINAS applies a clustering algorithm to the examples with *unknown* profile in the *short-time memory* to discover new clusters (step 8 and 9, Algorithm 2). Each new cluster is evaluated to verify if it represents a valid cluster (step 3, Algorithm 3).

A new cluster is considered valid if its cohesiveness, defined by the sum of squared distances between examples and centroid divided by the number of examples, is at least half of the cohesiveness of the normal model. If a new cluster is valid, it is necessary to decide if it represents an extension or a novelty. Otherwise, it is necessary to update the value of k (step 13, Algorithm 3). The value of k is adjusted whenever a cluster is considered invalid, according to the following conditions [14]: i) If most of clusters are invalid because they have low density, the value of k is increased. ii) If most of clusters are invalid because they have few examples, the value of k is decreased. iii) If all clusters are valid, the k value is not adjusted.

If a valid cluster is detected, the next step is to find the distance d between the centroid of the new hypersphere h_1 and the centroid of its nearest hypersphere h (step 5, Algorithm 3). If this distance is smaller than a threshold T, the new cluster is labeled as an extension. Its label is the same label of the hypersphere h (step 7, Algorithm 3). Otherwise, the algorithm labels the new cluster as a new concept (novelty) and creates a new label (step 9, Algorithm 3). In both cases, the algorithm updates the model to incorporate this new cluster (step 11, Algorithm 3). A hypersphere representing a new class can overlap part of one of the hyperspheres of the model. In this case, MINAS does not consider this hypersphere as a new class, but as an extension.

MINAS uses only one model representing the knowledge learned until the current moment. This model is composed by the clusters representing each one of the classes. In contrast to the majority of approaches in the literature that discovers a new extension or novelty only using the model learned in the *offline* phase, MINAS uses of only one model representing the knowledge acquired in the *offline* and *online* phases. Thus, it is possible, to extend concepts learned in the *online* phase. Additionally, in the presence of noise and outliers it is necessary to distinguish between the appearance of a novel concept and the noise/outlier, represented by sparse examples. Thus, MINAS applies a validation criterion in order to guarantee that a novel concept is represented by a cohesive and representative set of examples.

Algorithm 3 Novelty-Extension-Detection

```
 1: $ModelTemp \leftarrow$ k-Means($ShortTimeMem$,$k$)
 2: for all hypersphere $h_1$ in $ModelTemp$ do
 3:   if Validation-Criterion($h_1$) then
 4:     Let $h$ the nearest hypersphere to $h_1$ ($h \in Model$)
 5:     Let $d$ the distance between $h_1$ and $h$
 6:     if $d \leq T$ then
 7:       $Label_{h_1} \leftarrow Label_h$ {Extension}
 8:     else
 9:       $Label_{h_1} \leftarrow$ new label {New concept}
10:     end if
11:     $Model \leftarrow Model \cup h_1$
12:   else
13:     $k$ = update()
14:   end if
15: end for
16: return $Model$
```

In the online phase, each cluster stores the arrival time stamp of the last example clustered by it. The clusters that do not receive new examples for a given period are moved to a *sleep-memory*, allowing the model to forget outdated clusters. When a new cluster is validated, the *sleep-memory* is consulted to decide between a new concept or an extension. If the distance between the centroid of the new cluster and the centroid of one of the clusters in the *sleep-memory* is lower than a threshold, this new cluster represents an extension of the concept represent by the cluster in the *sleep-memory.* This may indicate the reoccurrence of a concept.

3.6 Heuristics used to select the value T

This study uses different strategies to select the best value of T. The first uses an absolute threshold value, given by the user. However, a good value can vary according to the data set. Additionally, it is difficult to find a threshold able to separate novelties from extensions properly. In general, small values of T produce many novelties and few extensions; high values of T, on the other hand, produce many extensions and few novelties. The second strategy is based on cluster cohesion. In a cluster with high cohesion, small values of T must be used to distinguish extensions from novelties. High values of T must be used in a cluster with low cohesion. The cohesion measure employed here is the sum of the squared distances between the examples and the centroid divided by the number of examples. The threshold value used is the cohesion value multiplied by a factor f. The value of f varies according to the data set. In the experiments carried out, we used the value 1.1.

4. EXPERIMENTAL EVALUATION

4.1 Data sets and Test Setup

MINAS was implemented in Java. The experiments were run on a PC with Intel Core 2 Duo processor with 2GHz and 3GB RAM. The code for k-means and Clustream were obtained from MOA [3]. In Table 1, we describe the data sets used in the experiments (all from UCI repository [6]). The experiments analyze the performance of MINAS to detect new concepts in a multi-class scenario. The first experiments compare the performance of MINAS with OLINDDA using classical data sets (rows 1 to 4 of the Table 1). After, we used artificial data sets generated by the MOA framework [3]. This framework allows deciding when new concepts will appear/disappear. Finally, we evaluate MINAS in a real multi-class scenario using the Forest Cover Type data set [6].

Table 1: Data sets used in the experiments

Data set	#Att.	#Exa.	Classes (#Exa.)	
			Normal	Others
Balance-scale	4	625	L (288)	R (288) B (49)
Biomed	5	194	normal (127)	carrier (67)
Breast-wisconsin	9	683	benign (444)	malignant (239)
Iris	4	150	Setosa (50)	Versicolor (50) Virgínica (50)
MOA	4	100,000	C1 (36,002) C2 (35,831)	C3 (18,180) C4 (99,87)
Forest Cover	54	581,012	Spruce-Fir (211,840) Lodgepole Pine (283,301)	Pond. Pine (35,754) Cotton/Willow (2,747) Aspen (9,493) Douglas-fir (17,367) Krummholz (20,510)

The proposed methodology considers a subset of the problem classes as the normal concept, represented in bold in the result tables, and uses the remaining classes as novel concepts (see Table 1) that will be identified by the algorithm. Each data set is divided into two subsets: training and test. The training set contains only examples from the normal concept classes. This set is used in the *offline* phase. The test set contains examples from all classes, including the normal concept. This set is used in the *online* phase. the test set contains the true class of each example and this information is used only to compute the accuracy measures. The training and test data sets were normalized.

For the initial decision model learning, we evaluated two options. The first uses the *k-Means* algorithm in each set that represents each normal concept class, creating the initial decision model. The second uses the *CluStream* algorithm to extract a data statistic summary for each class – *micro-clusters* – and use the *k-Means* algorithm in these *micro-clusters* to obtain the cluster set that composed the initial model. The last option is the fastest and may help to prevent overfitting problems, once it uses only representative elements. The first option was used in small data sets and the second in large data sets.

4.2 Experimental Methodology

This study proposes a new experimental methodology to evaluate the performance of algorithms for novelty detection in multi-class problems. It observes how the examples from each class are classified: belonging to a class of the decision model, learned in the *offline*/*online* phase, or *unknown.* The label of the classes learned in the *offline* phase is available in the training set and classes learned in the *online* phase are represented by N_i, where i is a sequential number. In contrast to the methodology used in the literature, the number of examples classified to the class C_i also includes the examples classified to an extension of the class C_i. The examples classified as *unknown* include those not explained by the current model and those used to build the clusters representing the novelties and extensions. Differently, the methodology used in the literature does not consider the examples used to build the novelties and extensions clusters as *unknown.*

In the proposed methodology, when the *online* process

Table 2: MINAS performance for the Iris data set

Class	Final Class Distribution (variance)			
	Set.	N1	N2	Unk
Set.	0.94(0.04)	0.00	0.00	0.06(0.04)
Ver.	0.00	0.55(0.05)	0.06(0.01)	0.39(0.02)
Vir.	0.0	0.51(0.01)	0.00	0.49(0.01)

starts, only the classes learned in the *offline* phase compose the decision model. Thus, it is possible to classify a new example as either belonging to one of the classes of the model or as *unknown*. After a valid set of examples is processed and a new concept is discovered, the decision model is updated. Whenever MINAS detects a new concept, it adds a new column to the confusion matrix to represent this new concept. In this matrix, the rows represent the classes observed in the stream and the columns the classes learned so far by MINAS (used to classify new instances). The number of columns increases as new concepts are discovered. Each cell m_{ij} of the matrix represents the number of examples of the class i classified by MINAS to the class j.

This methodology allows the evaluation of the classification process over the stream. In contrast to the batch-mode, where the evaluation measures are presented at the end of the classification process. In a data streams scenario, it is important to use measures that evolve over the stream. Specially, when a new class is detected, it is important to evaluate the behavior of the system until this moment and to increment the confusion matrix to support this new class.

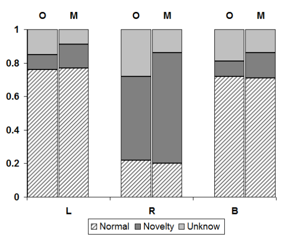
4.3 OLINDDA vs MINAS

We analyzed the performance of OLINDDA (O) and MINAS (N) in the following data sets: Balancescale, Biomed, Breastwisconsin, and Iris. In the experiments, only one class represents the normal concept (training set). The other classes were used in the test set. The 10-fold-cross-validation was used and the examples from each test set were shuffled to prevent many consecutive examples from the same class. The folds and the examples order are the same in all executions. Figure 2 shows the experimental results for OLINDDA (O) and MINAS (N) for these data sets. The horizontal axis, X, represents the problem classes and the vertical axis, Y, represents the percentage of examples classified into the classes normal, novelty and *unknown* by each algorithm. Because the training set contains a small number of examples, only the *k-means* algorithm was used to produce the clusters representing the classes.

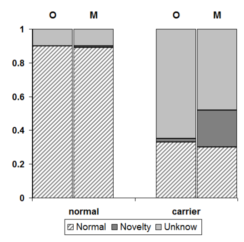
MINAS classified more examples from the new concept classes as novelty than OLINDDA. For such, MINAS used a different approach to identify novelties/extensions. On the other hand, the two algorithms classified a similar proportion of examples as normal, probably because both use the k-means algorithm to represent the normal concept. Table 2 shows the detailed accuracy results for the Iris data set, where $N1$ and $N2$ represent two new classes (novelties) detected by MINAS. Although MINAS found two new classes, most of the examples from Ver. and Vir. were classified as belonging to the class $N1$. According to these results, the threshold value was not able to separate these two classes.

4.4 Artificial Data Sets

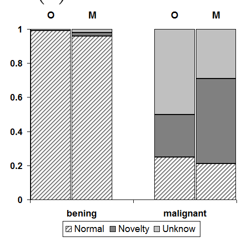
We generate the artificial data using the radial basis function (RBF) generator from MOA (*Massive Online Analysis*)



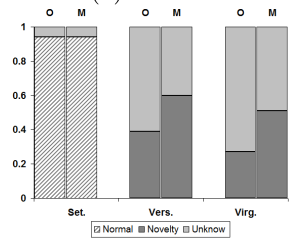
(a) Balancescale



(b) Biomed



(c) Breastwisconsin



(d) Iris

Figure 2: Experimental results for the OLINDDA and MINAS algorithms.

Table 3: MINAS performance for the MOA data set

Class	Final Class Distribution				
	C1	**C2**	N1	N2	Unk
C1	0.99	0.00	0.00	0.00	0.01
C2	0.00	0.99	0.00	0.00	0.01
C3	0.0	0.00	0.99	0.00	0.01
C4	0.00	0.00	0.00	0.99	0.01

toolkit [3]. A fixed number of random centroids was generated, each with a random position and a class label. All centers share the same initial standard deviation value (i.e., the centroid radius), which can vary with time. This creates a normally distributed hypersphere of examples surrounding each center with (possibly) varying densities. Drift is introduced by moving the centroids with constant speed. We tuned the generator settings to introduce at most two new classes along the time. The new event (class appearance/disappearance) frequency was set to 30,000 and the clusters move a distance of 0.01 at every 1,500 examples. This data set was divided into two subsets, training (first 10%) and test (the remaining). The training set contains only examples from the C_1 and C_2 classes. The test set contains examples from all classes (C_1, C_2, C_3, C_4).

Table 3 shows the accuracy results obtained by MINAS for this data set (here, k-fold-cross-validation was not used). It shows that 99% of the examples from the classes $C3$ and $C4$ were classified as novelty (represented by the classes N_1 and N_2 in the MINAS algorithm) . Thus, only 1% of the examples were classified as *unknown*, and as a result used to model the novelties. Here, the *CluStream* algorithm could be used because the training data set contains 3,000 examples.

4.5 Forest Cover data set

The Forest Cover data set [6] contains information about seven different types of forests. Each example has 54 numeric attributes. All attributes were normalized and used. As proposed in [11], the *offline* phase used 6,000 examples and the rest of examples were used in the *online* phase. The

Table 4: MINAS performance for the Forest Cover data set

Class	**Final Class Distribution**							
	C1	**C2**	**N1**	**N2**	**N3**	**N4**	**N5**	**Unk**
C1	0.62	0.38	0.00	0.00	0.00	0.00	0.00	0.00
C2	0.28	0.71	0.01	0.00	0.00	0.00	0.00	0.00
C3	0.0	0.40	0.59	0.01	0.01	0.01	0.00	0.01
C4	0.00	0.0	0.97	0.03	0.01	0.01	0.00	0.02
C5	0.29	0.69	0.02	0.00	0.00	0.00	0.00	0.00
C6	0.01	0.41	0.57	0.01	0.00	0.00	0.00	0.00
C7	0.92	0.07	0.0	0.03	0.02	0.00	0.00	0.01

offline phase contains examples from two classes, Spruce-Fir and Lodgpole Pine, here called $C1$ and $C2$, respectively. The *online* phase contains examples from all classes, including $C1$ and $C2$. The classes Pond Pine, Cotton/Willow, Aspen, Douglas-Fir and Krummholz are named here $C3$, $C4$, $C5$, $C6$ and $C7$, respectively.

Table 4 shows the accuracy performance of MINAS for this data set. It can be seen that MINAS identified examples from the classes $C3$, $C4$, and $C6$ as novelty. MINAS obtained the best results for the class $C4$, where 97% of the examples were classified as novelty. However, MINAS was not able to distinguish between different novelties, suggesting that the threshold value was not adequate to separate different classes. Moreover, the examples from the classes $C5$ and $C6$ were incorrectly classified as belonging to the normal concept, indicating that the decision model needs to be improved. An alternative to deal with this is to use other clustering techniques, especially non-spherical techniques.

5. CONCLUSIONS

This study presented a new novelty detection algorithm for data streams multi-class problem, called MINAS. This algorithm builds only one model to represent the classes and their extensions. Each class is represented by a set of clusters that are later used to classify new examples. The examples not explained by the current model are classified as *unknown*. A cohesive and representative set of *unknown* examples is used to discover new concepts or extensions to the known concepts. The differentiation between novelty and extension is made by a threshold value. We also proposed a new experimental methodology to evaluate novelty detection algorithms in multi-class problems.

Experimental results show that MINAS presented better results than OLINDDA for five UCI data sets. Using an artificial data set, MINAS was capable to differentiate two new concepts in the *online* phase and to correctly classify new examples with accuracy of 0.99. Results for the Cover Forest data set were promising. Examples of the classes learned in the *online* phase were classified as novelty and different novelties were detected. The next steps include the investigation of non-spherical clustering techniques to better represent the classes, and the development new approaches for the automatic choice of the threshold value.

6. ACKNOWLEDGMENTS

The authors acknowledge the financial support given by the project FCT-KDUS(PTDC/EIA/098355/2008), the ERDF through the COMPETE Programme, by National Funds through FCT within the project FCOMP - 01-0124-FEDER-022701, and CAPES, CNPq and FAPESP, Brazilian funding agencies.

7. REFERENCES

[1] C. C. Aggarwal, J. W. J. Han, and P. S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases*, volume 29, pages 81–92, 2003.

[2] T. Ahmed and M. Coates. Multivariate online anomaly detection using kernel recursive least squares. In *Proc. IEEE Infocom*, pages 625–633, Anchorage, Alaska, 2007.

[3] A. Bifet, G. Holmes, B.Pfahringer, P. Kranen, H.Kremer, T. Jansen, and T. Seidl. MOA: Massive online analysis, a framework for stream classification and clustering. 11:44–50, 2010.

[4] R. Casimir, E. Boutleux, G. Clerc, and A. Yahoui. The use of features selection and nearest neighbors rule for faults diagnostic in induction motors. *Eng. Appl. Artif. Intell.*, 19(2):169–177, 2006.

[5] L. A. Clifton, H. Yin, and Y. Zhang. Support vector machine in novelty detection for multi-channel combustion data. In *Proceedings of the Third international conference on Advances in Neural Networks - Volume Part III*, ISNN'06, pages 836–843, Berlin, Heidelberg, 2006. Springer-Verlag.

[6] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[7] J. Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, 1 edition, 2010.

[8] M. Z. Hayat and M. R. Hashemi. A dct based approach for detecting novelty and concept drift in data streams. In *International Conference on Soft Computing and Pattern Recognition (SoCPaR)*, pages 373–378. IEEE, 2010.

[9] M. Markou and S. Singh. Novelty detection: a review part 1: statistical approaches. *Signal Process.*, 83(12):2481–2497, 2003.

[10] S. Marsland, J. Shapiro, and U. Nehmzow. A self-organising network that grows when required. *Neural Network*, 15:1041–1058, 2002.

[11] 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. on Knowl. and Data Eng.*, 23(6):859–874, 2011.

[12] M. M. Masud, T. M. Al-Khateeb, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham. Detecting recurring and novel classes in concept-drifting data streams. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 1176–1181, Washington, DC, USA, 2011. IEEE Computer Society.

[13] M. Naldi, R. Campello, E. Hruschka, and A. Carvalho. Efficiency issues of evolutionary k-means. *App. Soft Comp.*, 11:1938–1952, 2011.

[14] E. J. Spinosa, A. C. P. L. F. Carvalho, and J. Gama. Novelty detection with application to data streams. *Intelligent Data Analysis*, 13(3):405–422, 2009.

[15] D. Y. Yan and C. Chow. Parzen-window network intrusion detectors. In *International Conference on Pattern Recognition*, ICPR '02, pages 385–388, 2002.