Artif Intell Rev (2016) 45:235–269
DOI 10.1007/s10462-015-9444-8

Novelty detection in data streams

Elaine R. Faria[1] · Isabel J. C. R. Gonçalves[2] · André C. P. L. F. de Carvalho[3] · João Gama[4]

Published online: 27 October 2015

Abstract In massive data analysis, data usually come in streams. In the last years, several studies have investigated novelty detection in these data streams. Different approaches have been proposed and validated in many application domains. A review of the main aspects of these studies can provide useful information to improve the performance of existing approaches, allow their adaptation to new applications and help to identify new important issues to be addresses in future studies. This article presents and analyses different aspects of novelty detection in data streams, like the offline and online phases, the number of classes considered at each phase, the use of ensemble versus a single classifier, supervised and unsupervised approaches for the learning task, information used for decision model update, forgetting mechanisms for outdated concepts, concept drift treatment, how to distinguish noise and outliers from novelty concepts, classification strategies for data with *unknown* label, and how to deal with recurring classes. This article also describes several applications of novelty detection in data streams investigated in the literature and discuss important challenges and future research directions.

Keywords Novelty detection · Data streams · Survey · Classification

✉ Elaine R. Faria
elaine@ufu.br

Isabel J. C. R. Gonçalves
isagoncalves@estg.ipvc.pt

André C. P. L. F. de Carvalho
andre@icmc.usp.br

João Gama
jgama@fep.up.pt

[1] Faculty of Computing, Federal University of Uberlândia, Uberlândia, Brazil

[2] Instituto Politécnico de Viana do Castelo, Viana do Castelo, Portugal

[3] Institute of Mathematics and Computer Science (ICMC), University of São Paulo, São Paulo, Brazil

[4] Laboratory of Artificial Intelligence and Decision Support (LIAAD-INESC TEC), University of Porto, Porto, Portugal

1 Introduction

Novelty detection (ND) is the ability to identify an unlabeled instance (or a set of them) that differs significantly from the known concepts. Since it is an important capacity for learning systems, it has received considerable attention from machine learning (ML) and data mining (DM) researchers. In the literature, it is possible to find many definitions for ND, such as:

- The recognition that an input differs in some respect from previous inputs (Perner 2008).
- Novelty detection is concerned with identifying abnormal system behaviours and abrupt changes from one regime to another (Lee and Roberts 2008).
- Novelty detection makes it possible to recognize novel concepts, which may indicate the appearance of a new concept, a change occurred in known concepts or the presence of noise (Gama 2010).

Several previous studies address ND in batch scenarios, where it is assumed that the whole set of data is available (Markou and Singh 2003a, b; Marsland et al. 2002; Schölkopf et al. 2000; Hoffmann 2007). However, nowadays, an important scenario, where ND represents an important challenge to be addressed, is data streams.

A data stream (DS) is a sequence of examples that arrive continuously. They are continuous, unbounded, flow at high speed and have a data distribution that may change over time (Silva et al. 2014). In DS scenarios, new concepts may appear and known concepts may disappear or evolve.

Since concepts are hardly ever constant, the application of ND techniques to DSs presents many challenges (Gama 2010), which include presence of:

- Concept drift, making difficult the distinction between new concepts and changes in known concepts;
- Noise and outliers, which can be confused with the occurrence of a new concept;
- Recurring concepts, which can be confused with the appearing of a new concept;
- Concept evolution, when the number of problem classes increases over time, and these novel classes need to be incorporated into the decision model.

Among the many DS applications in which the use of ND techniques is important, we can mention: intrusion detection (Coull et al. 2003; Spinosa et al. 2008), fault detection (Zhang et al. 2006), medical diagnosis (Perner 2008; Spinosa and Carvalho 2004), detection of interest regions in images (Singh and Markow 2004), fraud detection (Wang et al. 2003), forest cover type detection (Masud et al. 2011a), spam filter (Hayat et al. 2010), information retrieval (Gaughan and Smeaton 2005) and text classification (Li 2006).

The purpose of this study is to describe and analyze the main algorithms for ND in DS scenarios. There are important surveys about ND (Markou and Singh 2003a, b; Marsland 2003; Pimentel et al. 2014), anomaly detection (Chandola et al. 2009) and outlier detection (Hodge and Austin 2004; Gogoi et al. 2011). However, they only deal with batch scenarios, not addressing this task in the context of DSs. Besides, they focus on only one of the aspects of ND, the classification technique.

In the last years, several researchers have proposed different approaches for ND in DSs (Masud et al. 2011a; Spinosa et al. 2009; Hayat and Hashemi 2010; Faria et al. 2013a; de Faria et al. 2015b; Farid and Rahman 2012; Al-Khateeb et al. 2012a, b). A review of the main aspects of these studies may provide an useful inspiration for the development of new techniques and highlight the main issues to be addressed in future researches. For such, this survey covers different aspects of ND in DSs, like:

- Definition of the main approaches for DS analysis;

- Formalization of the ND in DSs, clearly distinguishing the *offline* and the *online* phases;
- Taxonomy of the main approaches for ND in DSs;
- Learning task;
- Number of classes and classifiers considered in the two stages;
- External feedback for model update;
- Forgetting mechanisms;
- Treatment of noise and outliers;
- Treatment of recurring classes;
- Evaluation measures.

This paper is structured as follows. Section 2.1 contextualize this article by pointing out the main differences between novelty, anomaly and outlier detection. Section 2.2 explains two important aspects of DSs, concept drift and concept evolution. Section 3 formalizes the ND task in DSs. Section 4 shows the taxonomy used to survey the algorithms for ND in DSs. Section 5 presents the details about the *offline* (or learning) phase. Section 6 presents in details the *online* phase discussing about the three task performed in this phase, classification of new instances, detection of novelty patterns, and update of the decision model. Section 7 discusses other relevant aspects about ND in DSs as treatment of noise and outliers, detection of recurring contexts and evaluation measures. Section 8 discusses the main limitations and strengths of the works presented in this article. Section 9 describes the main applications for ND in DSs. Finally Sect. 10 discusses the main challenges to be addressed as future works.

2 Important definitions

This section presents what we understand to be the main differences among novelty, anomaly and outlier detection. Besides, important concepts regarding ND, such as, concept drift and concept evolution, are discussed.

2.1 Novelty detection, anomaly detection and outlier detection

Novelty, anomaly and outlier detection are correlated terms. While in some contexts these three terms seem to have the same meaning and are used indistinctly, here these terms will be distinguished. In general, the latter two terms are more similar and frequently used to express very close problems.

In fact, novelty, anomaly and outlier detection are terms related to find patterns that are different from the normal, usual, patterns. While the terms anomaly and outliers give the idea of an undesired pattern, the term novelty indicates an emergent or a new concept that needs to be incorporated to the normal pattern.

According to Chandola et al. (2009), anomaly detection refers to the task of finding patterns in data that do not conform to expected behavior. These patterns are also referred to as anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants. According to Chandola et al. (2009), ND also aims to detect unobserved patterns (emergent, novel) in data. However this term is distinguished from anomaly detection, since, in the first, the novel patterns are typically incorporated into the normal model after their detection.

In Aggarwal (2013), the author defines outlier as a data point, which could be considered either an abnormality or a noise, whereas an anomaly refers to a special kind of outlier, which is of interest to an analyst. According to Gogoi et al. (2011), outliers can be candidates for aberrant data that may affect systems adversely, such as by producing incorrect results, misspecification of models, and biased estimation of parameters. Some of the causes for outliers are malicious activity, instrumentation error, change in the environment, human error (Chandola et al. 2009). For Gama (2010), a concise group of examples should be required as an evidence of the appearance of a novel concept, or novelty. On the other hand, sparse independent examples, whose characteristics largely differ from those defined as normal, should be seen simply as outliers, since there is no guarantee that they represent concepts.

According to Marsland et al. (2002), ND is useful in cases where an important class is under-represented in the training set. For Markou and Singh (2003a), it is an important task, since, for many problems, we never know if the currently available training data include on all possible object classes. According to Gama (2010), ND allows the recognition of novel profiles (concepts) in unlabeled data.

In this study, we treat novelty as a cohesive and representative group of examples representing a new concept to be incorporated to the decision model. This new concept is different from the known problem concepts and represents an evolution, e.g., the appearing of a new class. Isolated examples, non-representative and non-cohesive groups of elements, not explained by the normal concept, are considered as outliers. Thus, they need to be identified, but not added to the current model.

2.2 Concept drift and concept evolution

In concept learning, concept is the function to be learned by a learning algorithm, defined over the set of training examples, mapping input values to their corresponding output values (Mitchell 1997). In DS scenarios, which represent a non-stationary environment, the concepts are not static, but they evolve over time. Thus, two important phenomena, called concept drift and concept evolution, may occur.

According to Dries and Rückert (2009), concept drift is an important problem in ML and DM, and can be described as a significant change in the data distribution. The concept of interest may depend on some hidden context, and changes in the hidden context may induce changes in the target concept, producing concept drift (Widmer and Kubat 1996). There are different types of concept drift, named sudden, incremental, gradual, and recurring (Gama et al. 2013).

For Tsymbal (2004), a difficult problem in handling concept drift is to distinguish between true concept drift and noise. For example, if the concepts are viewed as images in a representation space, then they can change their shapes, sizes, and locations (Kolter and Maloof 2007). Examples of concept drift are changes in the pattern of a disease, changes in the weather and changes in the purchasing profile over years.

Two approaches, named blind and informed, are used to adapt the decision model in order to address concept drift (Gama et al. 2013). Blind approaches update the decision model at regular time intervals without verifying if changes really occurred. In general, the decision model is retrained using the latest stream labeled examples. Informed approaches modify the decision model when a change is detected. Most of the ND algorithms discussed in this survey use blind approaches to address concept drift.

Another aspect of DSs is concept evolution. Traditional ML algorithms assume that the number of classes is previously defined and that each example belongs to one of these classes (Park and Shim 2010). However, in real DSs this assumption is not valid, since all classes may not be known in the *offline* training phase and new examples may not fit in the existing classes. According to Masud et al. (2011a), a concept-evolution occurs when a new class emerges in a stream. Concept-evolution, also named emerging classes, is seen in tasks like network intrusion detection, spam identification and text classification.

It is important to highlight that ND must deal with concept drift, new emergent classes and outliers. Algorithms for ND should be able to detect concept drift and update the decision model in order to represent the changes in the known concepts, detect emergent classes and update the decision model with these new classes, and identify and discard noisy examples and outliers.

3 Formalization of the problem

A data stream is an infinite sequence of examples that flow rapidly in a continuous way, and whose distribution may vary over time.

Definition 1 (*Data Stream*) A data stream $\mathcal{S}$ is a massive sequence of multi-dimensional examples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$,..., which is potentially unbounded, arriving in time stamps $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_N$,...Each example x_i is described by an n-dimensional attribute vector (Aggarwal et al. 2003).

Data streams, as well as time-series, are sequence data sets. According to Han (2005), a sequence data set is a sequence of ordered events, with or without concrete notion of time. Data streams are typically multi-variate sequences, in which the notion of time is not always present. Web page traversal sequences and customer shopping transactions sequences are examples of sequence data, but they may not be time-series data (Han 2005). A time-series is a numeric sequence of observations of a random variable taken sequentially in time, which presents an intrinsic feature of dependence between adjacent examples (Box and Jenkins 1990). In data streams, this dependence does not always occurs and in several situations adjacent examples are completely independent. However, time-series in which the data flows continuously and in high speed, can be considered data streams.

In general, algorithms for ND in data streams work in two phases, namely *offline* and *online*. In the *offline* phase, a set of labeled examples is used to induce a classifier. These labeled examples represent the known concepts about the problem. From now on, we refer to known concepts about the problem as simply known concepts. Usually, the known concept is composed by examples from only one class, named *normal class*. In the *online* phase, whenever a new example arrives, it is classified in the normal class or it is rejected (or classified as abnormal, anomaly or novelty). This is the classical setting in one-class-classification (Spinosa et al. 2009; Hayat and Hashemi 2010; Tax and Duin 2008; Denis et al. 2005; Liu et al. 2003; Yeung and Ding 2003; Yeung and Chow 2002; Aregui and Denœux 2007; Tan et al. 2011).

Recently, several authors extended this framework to a multiclass context (Faria et al. 2013a; de Faria et al. 2015b; Farid and Rahman 2012; Masud et al. 2010a, 2011a; Al-Khateeb et al. 2012a, b). In this case, the previous formalization must be generalized to the multiclass context. In this new generalization, in the *offline* phase, each example from the training set has a label (y_i), where $y_i \in Y^{tr}$, with $Y^{tr} = \{C_{knw_1}, C_{knw_2}, \ldots, C_{knw_L}\}$, where C_{knw_i} represents the ith known class of the problem and L is the number of known classes. In the *online* phase, as new data arrive, new *novel classes* may be detected, expanding the set of class labels to $Y^{all} = \{C_{knw_1}, C_{knw_2}, \ldots, C_{knw_L}, C_{nov_1} \ldots, C_{nov_K}\}$, where C_{nov_i} represents the ith *novel class* and K is the number of *novel classes*, which is previously unknown.

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

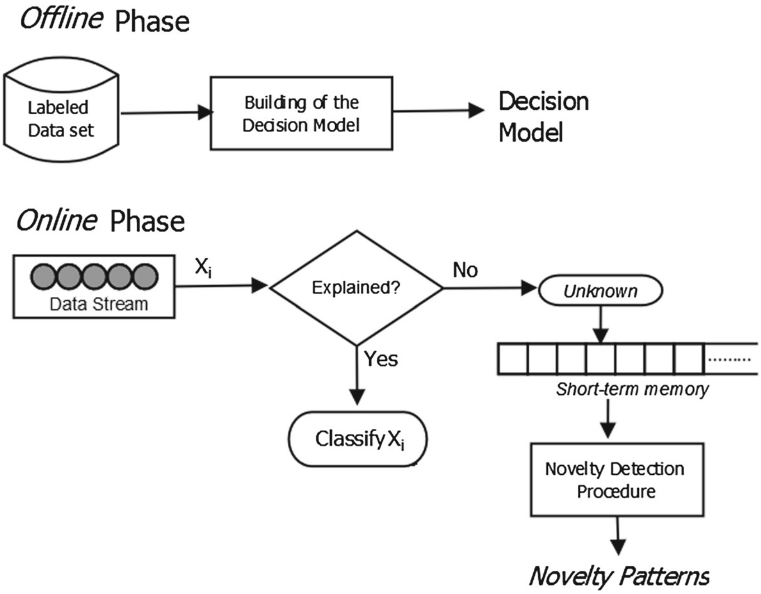

Fig. 1 Overview of the novelty detection task

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

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

The *unknown* examples are submitted to a ND task in order to produce different novelty patterns. Figure 1 shows an overview of the ND task.

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

It is important to highlight that, in a data stream classification task, besides the emergence of novel classes, known as concept evolution, another phenomenon must be considered, the concept drift. In this case, the known concept can change over time.

Definition 5 (*Concept drift*) Every instance $\mathbf{x}_t$ is generated from a source that corresponds to a certain 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. 2013).

Distinguish occurrence of new classes from changes in the known classes (concept drift) is an important issue to be addressed in DS research. In order to address each change that can occur in the data, ND algorithms, instead of identify when a concept drift occurs and, next, update the decision model, usually update the decision model constantly. If there is no treatment for concept drift, every change in the known concept can be detected as a novelty pattern. In contrast, it also necessary to forget outdated concepts.

Besides, noise or outlier cannot be considered as a *novelty pattern*, since they represent isolated or groups of non-cohesive examples.

The next section provides an overview of the main studies found in the literature investigating ND in data streams.

4 Overview of novelty detection in data streams

This section proposes a taxonomy to classify the ND algorithm for DSs. This taxonomy covers the main studies found in the literature, which are surveyed in this paper. This taxonomy characterize this work according to:

- *Offline* phase:
  - Number of classes (one or more-than-one)
  - Number of classifiers (single or ensemble)
  - Learning task (supervised or unsupervised)
- *Online* phase:
  - Classification:
    - Classification with *unknown* label option (yes or no)
    - Number of classifiers (single or ensemble)
  - Detection of novelty patterns:
    - Number of novel classes (one or more-than-one)
  - Decision model update:
    - External feedback (yes or no)
    - Number of classifiers (single or ensemble)
    - Forgetting mechanism (yes or no)
- Other aspects
  - Treatment of outliers
  - Treatment of recurring contexts
  - Evaluation measures

The algorithms to be analysed are presented in Table 1.

In general, the ND techniques work in two phases, the *offline* and the *online* phases. The *offline* phase, detailed in Sect. 5, receives a training set and induces a decision model, like in traditional ML tasks. Three important aspect of this phase are the type of learning used (supervised or unsupervised), the number of classifiers built (single or ensemble), and the number of classes that composes the known concept (one or more-than-one).

The *online* phase, detailed in the Sect. 6, receives an unlabeled DS and execute three task: classification of new examples, detection of novelty patterns and update of the decision model. The classification task uses the current decision model to classify new examples that continuously arrive over the stream. Some approaches classify every new example, while others mark as *unknown* the examples classified with low confidence. In addition, the classification task can be based on either a single classifier or an ensemble of classifiers. Besides, a few approaches consider that the novelty concept is composed by only one class, while others distribute the novelty concept into more-than-one class.

Another difference between the existing approaches is whether they use feedback to update the decision model. Some techniques also use forgetting mechanisms, which discard previous concepts that do not represent to current behavior of the stream.

Table 1 Algorithms for ND in DSs

Algorithms	References
1- ECSMiner - *Enhanced Classifier for Data Streams with novel class Miner*	Masud et al. (2011a)
2- MCM - *Multi Class Miner in Data Streams*	Masud et al. (2010a)
3- CLAM - *CLAss-based Micro classifier ensemble*	Al-Khateeb et al. (2012a)
4- MINAS - *MultI-class learNing Algorithm for data Streams*	Faria et al. (2013a); de Faria et al. (2015b)
5- OLINDDA - *OnLine Novelty and Drfit Detection Algorithm*	Spinosa et al. (2009)
6- DETECTNOD - *DiscrETE Cosine Transform based NOvelty and Drift detection*	Hayat and Hashemi (2010)
7- Tree for ND	Farid and Rahman (2012)
8- Ensemble Tree for ND	Farid et al. (2013)
9- HS-Trees - *Streaming Half-Space-Trees*	Tan et al. (2011)
10- SONDE - *Self-Organizing Novelty Detection*	Albertini and de Mello (2007)
11- Neural Network (NN) for ND	Rusiecki (2012)
12- Adaptive WOCSVM - *Weighted One-Class Support Vector Machine*	Krawczyk and Michal (2013)

Other relevant aspects of ND are discussed in Sect. 7. The first aspect is the treatment of outliers, which is not considered by all algorithms. The treatment of recurring contexts is also an important issue to be addressed by ND algorithms, but treated by few algorithms. Finally, the evaluation measures used by the data stream ND algorithms must also be highlighted, since there few studies addressing this task.

The Table 2 classify each one of the algorithms surveyed in this paper according to the previous taxonomy.

5 *Offline* phase

The *offline* phase represents the static learning phase of a ND algorithm. In this phase, the known concepts are learned, using a labeled data set. This phase involves three aspects (see Fig. 2), number of classes associated with the known concepts, number of classifiers, and learning task.

Several approaches proposed in the literature consider that the known concept is composed by only one class, named normal class or normal concept. They usually employ one-class classification techniques to induce a decision model, named normal model. Thus, only examples from one class (normal class) are used to induce the classifier. The classification output is binary, thus each example is classified as either normal, if explained by the decision model, or not normal, otherwise. This strategy is adopted by algorithms like Spinosa et al. (2009), Hayat and Hashemi (2010), Tan et al. (2011), Albertini and de Mello (2007), Rusiecki (2012) and Krawczyk and Michal (2013).

Other approaches consider the known concepts to be composed by more-than-one class and use multiclass classification approaches. Thus, the induced decision model is able to distinguish between three or more classes. This strategy is adopted by algorithms like Faria et al. (2013a), de Faria et al. (2015b), Al-Khateeb et al. (2012a), Masud et al. (2011a), Masud et al. (2010a), Farid and Rahman (2012) and Farid et al. (2013).

The second aspect is the number of classifiers. The techniques proposed in Faria et al. (2013a), de Faria et al. (2015b), Spinosa et al. (2009), Hayat and Hashemi (2010), Albertini and de Mello (2007), Rusiecki (2012), Farid and Rahman (2012) and Krawczyk and Michal (2013) create only one classifier in the *offline* phase, which will be update over the stream incrementally. Techniques, found in Masud et al. (2010a), Masud et al. (2011a), Tan et al.

Table 2 Taxonomy of the algorithms for ND in DSs

Algorithm	*Offline* phase		
	Number of classes	Number of classifiers	Learning task
1- ECSMiner	More-than-one	Ensemble	Supervised
2- MCM	More-than-one	Ensemble	Supervised
3- CLAM	More-than-one	Ensemble	Supervised
4- MINAS	More-than-one	Single	Supervised
5- OLINDDA	One	Single	Unsupervised
6- DETECTNOD	One	Single	Unsupervised
7- Tree for ND	More-than-one	Single	Supervised
8- Ensemble Tree	More-than-one	Ensemble	Supervised
9- HS-Trees	One	Ensemble	Unsupervised
10- SONDE	One	Single	Unsupervised
11- NN for ND	One	Single	Unsupervised
12- Adaptive WOCSVM	One	Single	Unsupervised
Algorithm	Other aspects		
	Treat recurring contexts	Treat outliers	Evaluation measures
1- ECSMiner	No	Yes	Fnew, Mnew, Err
2- MCM	No	Yes	Fnew, Mnew, Err, AUC
3- CLAM	Yes	Yes	Fnew, Mnew, Err
4- MINAS	Yes	Yes	Confusion matrix not squared
5- OLINDDA	No	Yes	Fnew, Mnew, Err
6- DETECTNOD	No	No	Fnew, Mnew, Err
7- Tree for ND	No	No	Fnew, Mnew, Err
8- Ensemble Tree	No	No	Fnew, Mnew, Err
9- HS-Trees	No	No	AUC
10- SONDE	No	No	Precision, recall, f-measure
11- NN for ND	No	No	2D-plot (signal × ND result)
12- Adaptive WOCSVM	No	No	Accuracy

Table 2 continued

Algorithm	*Online* phase		
	Classification		Novelty detection
	Classification with unknown label option	Number of classifiers	Number of novel classes
1- ECSMiner	Yes	Ensemble	One
2- MCM	Yes	Ensemble	More-than-one
3- CLAM	Yes	Ensemble	One
4- MINAS	Yes	Single	More-than-one
5- OLINDDA	Yes	Single	One
6- DETECTNOD	Yes	Single	One
7- Tree for ND	No	Single	One
8- Ensemble tree	Yes	Ensemble	One
9- HS-Trees	No	Ensemble	One
10- SONDE	No	Single	One
11- NN for ND	No	Single	One
12- Adaptive WOCSVM	No	Single	One

Algorithm			
	Decision model update		
	External feedback	Number of classifiers	Forgetting mechanism
1- ECSMiner	Yes	Ensemble	Yes
2- MCM	Yes	Ensemble	Yes
3- CLAM	Yes	Ensemble	Yes
4- MINAS	No	Single	Yes
5- OLINDDA	No	Single	No
6- DETECTNOD	No	Single	No
7- Tree for ND	Yes	Single	No
8- Ensemble tree	Yes	Ensemble	Yes
9- HS-Trees	No	Ensemble	Yes
10- SONDE	No	Single	Yes
11- NN for ND	Yes	Single	No
12- Adaptive WOCSVM	Yes	Single	Yes

(2011), Al-Khateeb et al. (2012a), Al-Khateeb et al. (2012b) and Farid et al. (2013) are based on an ensemble of classifiers. From these studies, Al-Khateeb et al. (2012a), Al-Khateeb et al. (2012b) use an ensemble per class. In all these studies, a chunk of labeled examples is used for the induction of each classifier.

5.1 Learning task

Although the labels of the training examples are known, several algorithms do not use this information when building a decision model. They suppose that all examples from the training

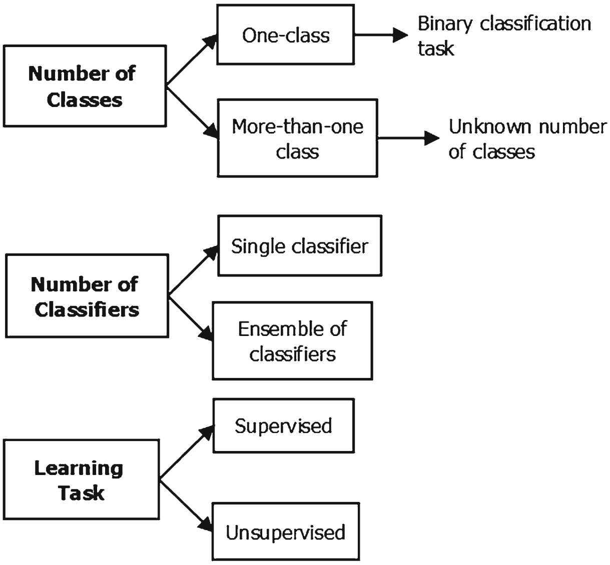

Fig. 2 Taxonomy used in the *offline* phase

set belong to the normal concept and they do not distinguish among their different classes. For these algorithms, the learning task is unsupervised. In opposite, several algorithms use supervised approaches in the building of the decision model, that is, they take into account the labels of the training examples. Thus, the decision model can classify examples in different known classes. These approaches are detailed next.

5.1.1 Normal concept composed by a set of classes

The techniques discussed in this section consider that the normal concept can be composed by a set of classes. They use the label of the training examples for the induction of a decision model able to represent the known problem classes. These techniques follow distinct approaches to induce the decision model.

MINAS (Faria et al. 2013a; de Faria et al. 2015b) associates a set of clusters to each one of the problem classes. Each cluster is represented by a center, a radius and a label, that indicates to each class it belongs. The training data set is divided into subsets, each one representing one class from the data set. Algorithms like $k-$means (MacQueen 1967; Lloyd 1982) or CluStream (Aggarwal 2013) are used to model each one of the problem classes by a set of clusters. The decision model is composed by the union of the k clusters obtained for each class.

CLAM (Al-Khateeb et al. 2012a) creates an ensemble of M classifiers for each problem class. As a result, c ensembles are built, where c represents the number of known classes. Each classifier in the ensemble is represented by a set of clusters, where each cluster is summarized by its radius, center and number of elements. Each classifier is built from a different chunk of data. Each chunk is divided into subsets, each one representing one of the classes present in the chunk. For each class, a set of clusters is built, using the $k-$means algorithm.

MCM (Masud et al. 2010a) and ECSMiner (Masud et al. 2011a) use only one ensemble to represent the known classes. While in Masud et al. (2010a), the ensemble is composed by

M KNN classifiers, in Masud et al. (2011a), the ensemble is composed by M decision trees. Each classifier is built using data from a chunk and consider all the classes present in this chunk. The size of each chunk, as well as the number of ensembles M, are input parameters defined by the user. In Masud et al. (2010a), each classifier is represented by k clusters, obtained using the $k-$means algorithm. The summary of each cluster (centroid, radius and frequencies of examples belonging to each class) is stored. In Masud et al. (2011a), after inducing the decision tree, a clustering algorithm is applied to each leaf node of the tree, in order to identify examples not explained by the current model (*unknown*). In Masud et al. (2011a), the authors also propose to use an ensemble of KNN classifiers.

In Farid et al. (2013), an ensemble of three decision trees is proposed. The first step is to initialize a weight for each example of the training set D, using the highest posterior probability of a Naive Bayes classifier. The first tree is built using a data set, named D_{new} obtained from a selection and replacement technique in D. For the two other trees, the examples with higher weights are selected, creating a new training set. The weight of the examples is updated, according to their classification by the latest decision tree (correctly classified or misclassified). Each leaf node of the tree is also clustered and has the percentage its examples regarding the training set size calculated. A weight T is associated to the tree, based on its classification accuracy rate to classify the examples of the original training set. The work is an extension of Farid and Rahman (2012), which used only one decision tree.

5.1.2 Normal concept composed by one class

Several techniques do not use the class label of the examples to induce the decision model. They assume that all training examples belong to one class, named normal class. Some of these techniques are based on clustering algorithms or neural networks (NN).

OLINDDA (Spinosa et al. 2009) uses a clustering algorithm to create a set of clusters to represent the normal concept, without distinction between different normal classes. The clustering algorithm (e.g. $k-$means) produces a set of k clusters, which are represented by their center and radius. In addition, a macro-hypersphere is created around the clusters that compose the normal model, whose center is the centroid of the clusters and the radius is the distance to the farthest center. This macro-hypersphere is used to separate novelty from extensions in the *online* phase. Thus, the normal concept is composed by only one class, represented by a set of clusters. DETECTNOD (Hayat and Hashemi 2010) also uses a clustering algorithm to create a set of k clusters. However, afterwards, a clustering algorithm is applied to each cluster, producing sub-clusters. By using interpolation, the number of examples in each sub-cluster is set to the same value. In order to decrease the memory usage, a Discrete Cosine Transform (DCT) is applied to the sub-clusters.

In Albertini and de Mello (2007), the authors proposed an unsupervised NN, named Self-Organizing Novelty Detection (SONDE), to represent the normal class. Though the authors do not propose its use for DS environments, SONDE can be easily used in DSs. SONDE has three layers: input layer, which normalizes the data, competitive layer, where neurons compete to provide an output to the normalized data, and the Best Matching Unit (BMU), where the winner neuron is used to represent a new received pattern. Neurons are used to represent the input data as well to detect novelties. Each neuron is represented by a centroid, an average radius, and a similarity degree to recognize new examples. Similar input data are assigned to the same neuron. The examples used in its training represent the normal concept.

In Rusiecki (2012), two autoregressive feedforward NN induce the decision model. The first NN is trained with a robust learning algorithm, which tries to minimize the error by decreasing the influence of outliers in the training set. The second uses the traditional backpropagation algorithm to minimize the least squares error. A time window of a predefined length which moves in discrete periods is used to represent the recent data. The training set, whose size is the window length, is used to train the NNs.

In Krawczyk and Michal (2013), the authors use a technique based on SVMs (Vapnik 1998), named Weighted One-Class Support Vector Machine (WOCSVM) (Bicego and Figueiredo 2009) to represent the normal class. Based on One-Class Support Vector Machine (OCSVM) (Schölkopf et al. 2001), an adaptation of SVMs to one-class classification, WOCSVM adds one weight value to each example from the training set. The weights are used to minimize the hypersphere volume and still encompass all the training data. In the training phase, the first chunk is used to induce a WOCSVM classifier. The minimization function is presented in Eq. 1, subject to the constraint given by Eq. 2, and the weight associated to each example in Eq. 3. In these Equations, C represents the center of the hypersphere, R is its radius, w_i ($0 \leq w_i \leq 1$) and ξ_i ($\xi_i \geq 0$) are the weight and slack variable respectively, associated to the ith example, N is the number of examples, O is a parameter that controls the optimization process (larger O, less outliers), and δ is a parameter that, when larger then 0, avoids the division by 0 in Eq. 3.

$$\Theta(C, R) = R^2 + O \sum_{i=1}^{N} w_i \xi_i \tag{1}$$

$$\forall_{1 \leq i \leq N} : \|x_i - C\|^2 \leq R^2 + \xi_i \tag{2}$$

$$w_i = \frac{|x_i - x_{mean}|}{R + \delta} \tag{3}$$

In Tan et al. (2011), an ensemble of Half-Space-Trees (HS-Tree) (Ting et al. 2009) is used to represent the normal class. HS-Trees are binary trees that are induced without using the training examples, but using the data space dimensions. Each node of the HS-Tree contains the number of examples (mass) that reaches this node in the reference chunk (mass r) and in the latest chunk (mass l). After building the M first HS-Tree, the system uses the first chunk of examples to update the mass r of each tree. For this, each example descends each one of the M HS-Trees, updating the mass r in its corresponding node.

6 *Online* phase

In the *online* phase, new unlabeled data arrive continuously (in general in high speed). In this phase three tasks are executed, classification of new examples, detection of novelty patterns, and adaptation of the decision model. Figure 3 shows an overview of the tasks performed in this phase.

6.1 Classification

The classification task verifies if a new example can be explained by the current decision model. Figure 4 summarizes the taxonomy used in this task.

Some approaches, like Albertini and de Mello (2007), Krawczyk and Michal (2013), Tan et al. (2011), Farid and Rahman (2012) and Rusiecki (2012), classify every new example as either normal, if explained by the current decision model, or novelty (anomalous or abnormal), if not explained. In these approaches, a novelty is detected by the presence of a single new example not explained by the current model. Other approaches (Faria et al. 2013a; de Faria et al. 2015b; Spinosa et al. 2009; Hayat and Hashemi 2010; Farid et al. 2013; Masud et al.

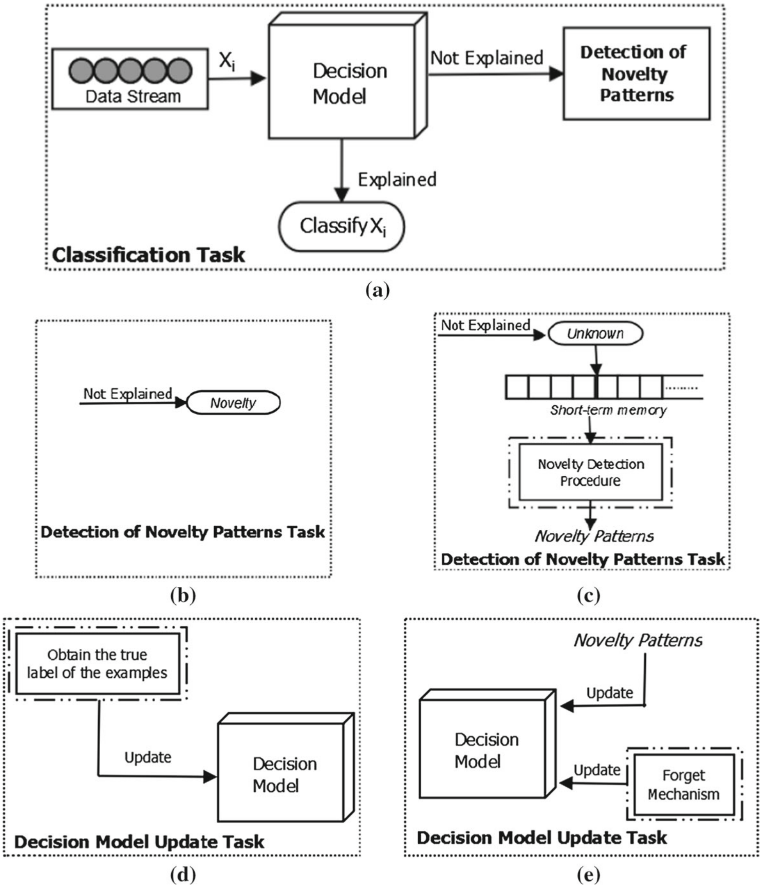

Fig. 3 Tasks performed in the *online* phase **a** classification, **b** detection of novelty patterns without *unknown* label option, **c** detection of novelty patterns with *unknown* label option, **d** decision model update with feedback **e**, decision model update without feedback

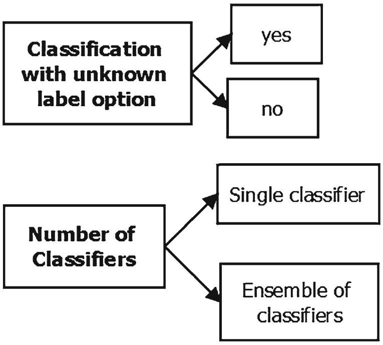

Fig. 4 Taxonomy used in the classification task

2010a, 2011a; Al-Khateeb et al. 2012a) do not immediately classify every new example, but, instead, assign an *unknown* profile to those not explained by the current decision model. These examples are placed in a short-term memory for future analysis and can be later used to model novelty patterns. In these approaches, a novelty pattern is composed by a set of cohesive and representative set of examples not explained by the current model. In Faria et al. (2013a), de Faria et al. (2015b), Spinosa et al. (2009) and Hayat and Hashemi (2010) the *unknown* examples are also used to model extensions of the known concepts.

Additionally, two different approaches have been used to classify new examples, based on a single classifier and based on an ensemble of classifiers.

6.1.1 Single classifier

The techniques from (Faria et al. 2013a; de Faria et al. 2015b; Spinosa et al. 2009; Hayat and Hashemi 2010) use a decision model represented by a set of clusters and allow the classification with *unknown* label option. In Spinosa et al. (2009), for the classification of a new example, the distance between the new example and one of the clusters is measured. If the distance is smaller than the radius of the cluster, the example is explained by the decision model. Otherwise, it is marked as *unknown*. In this case, as the decision model is composed by the normal, extension and novelty submodels, each one of these submodels is checked. The example is assigned to the class associated with the submodel of the cluster it falls within, allowing three possibles classifications: normal, extension or novelty. In Hayat and Hashemi (2010), the closest sub-cluster of the normal model is selected to classify a new example and a unknown score is computed. This score represents the difference between the DCT representation of the sub-cluster before and after the inclusion of the new example. If this score is above a threshold, the example is labeled as *unknown*. In Faria et al. (2013a) and de Faria et al. (2015b), there is only one decision model, without submodels, composed by the clusters learned *offline* and *online*. The closest cluster is used to classify a new example, which receives the label associated with the cluster. In this case, in the *online* phase, a new example is classified either in one of the classes learned in the *offline* phase, which can be extended in the *online* phase

Two studies, Albertini and de Mello (2007) and Rusiecki (2012), propose the use of a single classifier based on NNs, without the *unknown* label option. In Albertini and de Mello (2007), classifying a new example means finding the neuron that best represent this example. For such, the distance between the new example and each neuron is measured and later used in the definition of the activation value of the neuron. The neuron with the highest activation value is chosen to classify the example. However, this neuron has to present a minimum similarity degree. Otherwise, it cannot be used to explain this new example, indicating the presence of a novelty. In Rusiecki (2012), a new example is submitted to two NN. If the difference between the results produced by these two NN is smaller than a threshold, the example is classified as belonging to the normal concept. Otherwise, it is labeled as novelty.

In Krawczyk and Michal (2013), a single classifier is also used, without the *unknown* label option. In the work, the classification of a new example during the *online* phase checks whether it falls within a hypersphere, created in the *offline* phase by using a single SVM classifier. If it is true, the new object is labeled as belonging to normal concept.

In another study, (Farid and Rahman 2012), a decision tree is used to classify new examples from a stream, without considering the *unknown* label option. When a new example reaches a leaf node, the percentage of examples classified by this leaf node is updated. An increase in this value may be an indicative of a novel class.

6.1.2 Ensemble of classifiers

In the approaches that use an ensemble of classifiers, each classifier corresponds to one-vote and the final decision is made by a rule that combines the outputs of multiple classifiers (Menahem et al. 2013). Some of the rules found in the literature are majority voting, mean vote, mean weighted vote, max rule, etc. (Menahem et al. 2013; Tax et al. 2001; Juszczak and Duin 2004).

In Masud et al. (2011a), Masud et al. (2010a), Al-Khateeb et al. (2012a) and Farid et al. (2013), an ensemble is used to classify a new example using the *unknown* label option. In Masud et al. (2011a) and Farid et al. (2013), the first task is to verify whether the example is *unknown* by using an ensemble of decision trees. If the example is outside all the clusters present in the leaf node reached, for all classifiers in the ensemble, it is labeled as *unknown*. Otherwise, it is classified using the majority vote of the ensemble. The authors also propose to use an ensemble of KNN classifiers. In this variation, to classify a new example, it is initially necessary to verify whether it should be marked as *unknown*. This occurs when the example is outside the radius of its closest cluster for all classifiers of the ensemble. Otherwise, it receives the label with the highest frequency in the closest cluster. However, the final decision is given by the majority vote of the classifiers in the ensemble.

A similar approach is used in Masud et al. (2010a). However, a slack space is created for each cluster, where a cluster is represented by a hypersphere (center and radius). If an example is outside the radius of the hypersphere, but inside the slack space, it is considered as belonging to this hypersphere. The slack space is defined by a threshold set by the user.

Another technique that uses an ensemble of classifiers, named CLAM (Al-Khateeb et al. 2012a), employs one ensemble of classifiers per class, where each classifier is represented by a set of clusters. When CLAM receives a new example, each ensemble verifies if the example is *unknown*. An example is considered *unknown* by an ensemble if it is outside the decision boundary of the majority of its classifiers. A new example is labelled as *unknown* if all ensembles considered it *unknown*. Otherwise, the example is classified in one of the known classes. For such, for each ensemble, the distances between the example and the clusters that compose each classifier in the ensemble are measured. The example is classified in the class of the ensemble with the minimum distance.

In Tan et al. (2011), an ensemble of HS-Trees without the *unknown* label option is used. When a new example needs to be classified, it is assigned by each one of the t HS-Trees to a leaf node and a score is computed for each tree, using the mass r (an information computed in the training phase for each node). The sum of the scores of a new example, obtained for each tree, defines an anomaly score, which is used to decide if a new example is anomalous or not.

6.2 Detection of novelty patterns

In general, this task uses unlabeled examples not explained by the current decision model to identify novelty patterns. In anomaly detection systems, the presence of only one example not explained by the current model is sufficient to identify an anomaly behavior. However, several approaches consider that a novelty is composed by a set of cohesive and representative examples not explained by the model. The taxonomy adopted in this task can be seen in Fig. 5.

Most of the approaches from the literature (Masud et al. 2011a; Al-Khateeb et al. 2012a; Farid and Rahman 2012; Farid et al. 2013; Albertini and de Mello 2007; Rusiecki 2012; Tan et al. 2011; Krawczyk and Michal 2013) considers the novelty concept to be composed by only one class. In this case, an example, or a set of examples, not explained by the current

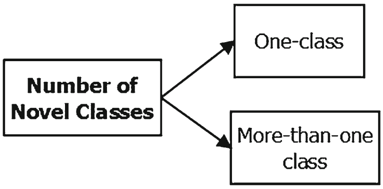

Fig. 5 Taxonomy used in the detection of novelty patterns

decision model are classified as novelty. Differently, Masud et al. (2010a); and de Faria et al. (2015b) consider that different novel classes, composed by one or more novelty patterns, may appear over time. Thus, the different novelty patterns should be distinguished when they appear. As a result, it is not sufficient to classify a new example only as novelty, but it must be labeled with the novelty pattern to which it belongs.

Several approaches have been proposed to identify novelty patterns from unlabeled examples. Some of them use clustering algorithms to find groups of similar examples in order to identify novelties. In OLINDDA (Spinosa et al. 2009), every time a new example is labelled as *unknown*, the $k-$means algorithm is executed, producing k clusters. Each cluster is checked in order to verify if it is valid. Otherwise, it is discarded. A cluster is considered valid if it is cohesive and representative. Cohesiveness evaluates the degree of similarity between the examples in a cluster. Representativeness establishes a minimum number of examples in a valid cluster. Threshold values are used to eliminate clusters that are not representative (user threshold) and cohesive (threshold based in the normal model). When a new valid cluster is identified, it is verified if it represents a novelty or an extension of the normal model. If the centroid of the new valid cluster is inside the macro-hypersphere, created using the clusters of the normal class, it is considered an extension of the normal concept. Otherwise it is considered a novelty. Thus, the novelty concept is composed by a set of clusters, but there is no distinction between different novelty patterns.

DETECTNOD (Hayat and Hashemi 2010) applies a clustering algorithm to the *unknown* data. When deciding between a extension or a novelty, DETECTNOD uses the distance between the new cluster and its nearest sub-cluster in the normal model. If this distance is lower than a threshold, the new cluster is considered an extension, otherwise it is considered a novelty. The novelty concept is also composed by a set of clusters, but there is no distinction between different novelty patterns.

The MINAS (Faria et al. 2013a, de Faria et al. 2015b) algorithm also groups *unknown* examples by using a clustering algorithm. Like OLINDDA, MINAS eliminates invalid clusters. When a new cluster is created, it is evaluated to decide if it represents a novelty pattern or an extension of the known classes. If the distance between the centroid of a new cluster and the centroid of its nearest cluster is less than a threshold, the new valid cluster is considered an extension of a known concept, and its label is the label of its nearest cluster. Otherwise, it is considered a novelty pattern and a sequential label is associated to it ($N1$, $N2$, $N3$, $\ldots$). Thus, a novelty pattern can be composed by one or more clusters and different novelty patterns can be detected over the stream.

After clustering the examples in a short-term memory, ECSMiner (Masud et al. 2011a) calculates a clustering validation internal measure, named q-NSC measure, for each cluster in each classifier of the ensemble (see Eq. 4). This measure is a combination of cohesion and separation and produce values in the range [−1,+1]. In this Equation, $\overline{D}_{c_{out},q(x)}$ is the mean distance from an *unknown* example x to $\lambda_{c_{out},q}(x)$, $\lambda_{c_{out},q(x)}$ is the set of q-nearest neighbors of x regarding the *unknown* example, and $\overline{D}_{c_{min},q(x)}$ is the minimum among all $\overline{D}_{c,q(x)}$, where c is an existing class. A positive value indicates that the *unknown examples* are more cohesive and more distant from the existing classes. If the number of clusters in a classifier with positive q-NSC value is larger than a threshold, the classifier votes for the detection of a new class. If all classifiers vote for the detection of a new class, a novelty is identified. For each chunk, if more than two novel classes appear simultaneously, they are considered only one novelty. CLAM (Al-Khateeb et al. 2012a) uses a similar procedure to ND, based on the q-NSC measure. CLAM, like ECSMiner, does not distinguish between different novel patterns in the same chunk.

$$q - NSC(x) = \frac{\overline{D}_{c_{min},q(x)} - \overline{D}_{c_{out},q(x)}}{max(\overline{D}_{c_{min},q(x)}, \overline{D}_{c_{out},q(x)})} \tag{4}$$

Another technique that uses the q-NSC metric is MCM (Masud et al. 2010a). For each example marked as *unknown*, MCM calculates its Gini Coefficient, whose value to separate the examples that represent a concept evolution from the noisy or concept-drift examples. If the Gini Coefficient is higher than a threshold, the example represents a concept evolution. Next, the q-NSC is computed for each example (see Eq. 4). If the q-NSC value for an *unknown* example is negative, the example is removed. Afterwards, the Nscore measure (see Eq. 6) is calculated for each example that has a positive value for q-NSC. In this Equation, r is the radius of the closest cluster to the example, d is the distance between the example and the center of the closest cluster and $minweight$ is the minimum weight among all the *unknown* examples with positive q-NSC. A larger value of Nscore indicates a higher probability for the example to be considered a novelty. At the end, the Nscore(x) is discretized in n intervals, a cumulative distribution function is computed (CDF), and the discrete Gini Coefficient is computed. If the Gini Coefficient value is larger than $\frac{n-1}{3n}$, a novel class is detected. MCM distinguish different novel classes by computing the connected components of a graph, built from the clusters of the examples marked as novelty.

$$weight(x) = e^{r-d} \tag{5}$$

$$Nscore(x) = \frac{1 - weight(x)}{1 - minweight} q - NSC(x) \tag{6}$$

Anomaly detection systems were proposed in Albertini and de Mello (2007), Rusiecki (2012), Farid and Rahman (2012), Farid et al. (2013), Tan et al. (2011) and Krawczyk and Michal (2013). In these systems, the presence of only one *unknown* example indicates an anomaly behavior. Although most of these studies use the term novelty detection to define this anomalous behavior, we believe that a more adequate definition for this task is anomaly detection. We believe that a novelty should be composed by a cohesive and representative set of examples not explained by the current model.

In Albertini and de Mello (2007), when no neuron is able to classify a new example a new neuron is created, indicating that a novelty was detected. The centroid of the new neuron centroid is equal to the attribute vector of the new example, the minimum similarity degree is set to a predefined value α, and the radius is set to $-\ln(\alpha)$.

In Rusiecki (2012), when the difference between the results produced by the two NNs for a new example is larger than a given threshold value T, a novelty is detected. In order to find the best value for this threshold, an approach based in the standard deviation of differences between NN outputs is employed, see Eq. 7. In this Equation, $y_{mse}(x_i)$ is the output of the traditional neural network and $y_{mls}(x_i)$ is the output of the robust network for the i^{th} example, where k is a constant defined by the user.

$$T = k * Std(|y_{mse}(x_i) - y_{mls}(x_i)|) \tag{7}$$

The works from Farid and Rahman (2012), Farid et al. (2013), Tan et al. (2011) use a decision tree structure to detect novelties. In Farid and Rahman (2012), the proposed ND algorithm checks if the addition of a new example in a leaf node increases a percentage p (proportion of examples in the leaf node considering the number of examples in the training set) already computed. An increase indicates the presence of a novel class. If this new example is outside all the clusters of this leaf, it belongs to a novel class arrived. In Farid et al. (2013), if a new example does not belong to any of the clusters in its corresponding leaf, it is placed in a short-term memory and a novel-flag is set to 1. If the number of examples classified by a leaf node increases or decreases (in comparison with the previous calculated value) and the novel-flag value is 1, the example belongs to a novel class.

In Tan et al. (2011), an accumulated score is computed for each new example (described in Sect. 6.1). If this value is higher than a threshold, the example is considered anomalous. In Krawczyk and Michal (2013), if a new example is labeled as not belonging to the normal concept, it is considered an anomalous example.

6.3 Update of the decision model

Algorithms for ND in DSs should update their decision model continuously, in order to address concept drift and concept evolution. Figure 6 shows three aspects to be considered in this phase:

- Type of update, which can be carried out with or without feedback;
- Number of classifiers;
- Use of forgetting mechanisms to remove outdated concepts.

The use of single classifiers (Faria et al. 2013a; de Faria et al. 2015b; Spinosa et al. 2009; Albertini and de Mello 2007; Hayat and Hashemi 2010; Farid and Rahman 2012; Rusiecki 2012; Krawczyk and Michal 2013) in contrast to an ensemble of classifiers (Farid et al. 2013; Masud et al. 2010a, 2011a; Al-Khateeb et al. 2012a; Tan et al. 2011) affects the update of

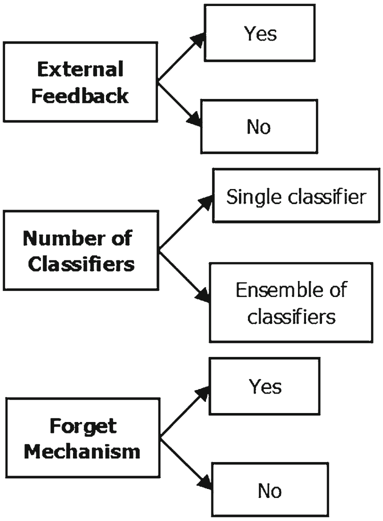

Fig. 6 Taxonomy used in the decision model update task

the decision model. While single classifiers usually are induced by incremental algorithms, for ensembles, whenever a new classifier is trained, an old, outdated classifier is usually removed.

The use of external feedback in the update of the decision model is also an important aspect to be considered. Techniques that use feedback assume that the true label of all the examples will be available after a delay. Besides, there are techniques that ask the user the label of a subset of the examples, the important examples, in the stream (active learning). In contrast, in techniques that do not use feedback, the decision model is updated without information about the true label of the examples. It is important to observe that obtain the true label of all examples is a time consuming and expensive task, and in several real problems can be impracticable.

6.3.1 Number of classifiers and external feedback

Single classifiers without external feedback: In some algorithms, like Faria et al. (2013a), de Faria et al. (2015b), Spinosa et al. (2009), Hayat and Hashemi (2010) and Albertini and de Mello (2007) the decision model is composed by a single classifier, which is updated incrementally. These algorithm do not use feedback to update their decision model.

In MINAS (Faria et al. 2013a; de Faria et al. 2015b), whenever a new valid cluster is obtained using the *unknown* examples, it is labeled as either a novelty pattern or an extension. In both cases, this new cluster is included in the current decision model and used to classify new examples. In Spinosa et al. (2009), the new valid cluster is also included in the decision model, however there is one decision model to represent the normal concept, one to the extension concept and one to the novelty concept. In Hayat and Hashemi (2010), the normal model is updated with the clusters that represents a drift. The strategy followed replaces an old cluster by a new one. In Albertini and de Mello (2007), when a new example is classified by a neuron, it is considered to represent this example, updating its radius and centroid. When none of the existing neurons can explain a new example, a new neuron is created.

Single classifiers with external feedback: The works proposed in Farid and Rahman (2012), Krawczyk and Michal (2013) and Rusiecki (2012) also use only a single classifier, however they are updated with feedback information. In Farid and Rahman (2012), a new example is considered a novelty when none of the clusters present in the leaf node reached by the example can explain it. In this case, the example is added to the training data set and the decision model is updated. In Krawczyk and Michal (2013), the authors propose a single classifier based on SVMs. A passive incremental learning technique is used to update the decision model. This technique adds every new example to the training data with an associate weight. Two approaches are proposed to calculate the weight for each example. The first uses the Eq. 3, the second assigns the highest weight values to the most recent examples. As a result, all examples in the latest chunk have their weight set to 1. In Rusiecki (2012), the authors argues that update the network every time a new example arrives is computationally expensive. According to him, a better approach is to train a network only once for a given period of time. Training examples are accumulated until a given size is achieved, determined by a parameter named window length. Next, the network weights are updated by using these examples. Another parameter defines the distance between two time windows, since not all examples are used for the training.

Ensemble of classifiers with external feedback: Some works that use an ensemble of classifiers are Masud et al. (2010a), Masud et al. (2011a), Al-Khateeb et al. (2012a), Farid et al.

(2013) and Tan et al. (2011). Most of them assume that, after a delay, the true label for all examples is available. Thus, from time to time, they update offline their decision model. These algorithms are able to address concept drift and incorporate new classes to the decision model using the feedback information. However, if the true label of the examples is not available or the delay takes a long time, these algorithms have their performance decreased.

In Masud et al. (2010a) and Masud et al. (2011a) in order to handle concept drift, the ensemble is continuously updated, by substituting in the ensemble the model with the highest classification error by a new model. In Farid et al. (2013), each decision tree in the ensemble has an associated weight. A small weight value is associated with a low accuracy classifier, which can be replaced by a new one. The ensemble of the classifiers is updated, if a new decision tree has a weight higher than every other decision trees in the ensemble. In Al-Khateeb et al. (2012a), whenever a new micro-classifier is trained, it replaces one of the micro-classifiers from its corresponding ensemble. The micro-classifiers with the highest prediction error, considering the corresponding subset in the last training chunk, is removed.

Finally, some ensemble approaches assume that the labeled examples are limited, once to label all the examples is very time consuming. In Masud et al. (2010b), after classify each example from a chunk in one of the known classes or as novelty, it selects a set of examples weakly classified to be labeled by a specialist. These examples are placed in a buffer of training examples. When there is a sufficient number of examples in this buffer, a new learning supervised phase updates the current classification model.

Ensemble of classifiers without external feedback: In the ensemble of classifiers used in Tan et al. (2011), a classifier can be updated, but there is no replacement of classifiers in the ensemble. In addition, the tree structure is not modified during the update, only the attributes mass l and r are updated. Thus, the external feedback is not used to update the ensemble of classifiers.

6.3.2 Forgetting mechanism

In addition to the update of the decision model, some works use a mechanism to forget previous, outdated, concepts. The approaches based in an ensemble, identify and remove these concepts every time a new classifier is trained, replacing an old model. In approaches that use a single classifier, a specific mechanisms for such is included.

The approach adopted by MINAS (Faria et al. 2013a; de Faria et al. 2015b) to forget previous concepts, actually previous clusters, is based on the assumptions that clusters that not received new examples for a long time should be removed. If a cluster does not receive a new example for a long period, it goes to a *sleep memory*. In Albertini and de Mello (2007), as known concepts can change, neurons adapt, forgetting past information. When the neuron is updated due to its use to classify a new example, its centroid and radius are updated following the exponential weighted moving averages (EWMA), which uses parameters that define the influence of the old examples in the current behavior of the stream.

In Krawczyk and Michal (2013), an incremental forgetting reduces the weights of the examples from the previous chunk. After reading some chunks, old examples have their weight set to 0 and are removed from the training data, assuming that they are not important in the current stream. Two approaches are proposed to calculate the weight of the examples from the previous chunks. The first decreases the weight values gradually, taking into account their initial importance. Thus, objects with different initial weights will be removed in different timestamps. The second uses an aligned decrease that does not consider the initial importance of the weights. Thus, all the objects in the same chunk will be removed at the same timestamp.

7 Other relevant aspects of the novelty detection algorithms

In this section, we discuss some aspects of ND that, although important for this task, are addressed only by few studies. The first aspect is the detection of recurring contexts, which can lead to high false positive rates when a recurring class is confused with the emergence of a new class. The second is the treatment of noise and outliers, which can be confused with concept drift or concept evolution. Finally, the adaptation of classical classification measures and the design of new evaluation measures for ND in DS scenarios.

7.1 Detection of recurring contexts in novelty detection

Recurring contexts are an important phenomenon observed in many real world applications, like climate change, electricity demand, buyer habits and detection of new topic in a text. In these applications, the class definitions may change when previous situations recur, in periodic or random way, after some period of time (Elwell and Polikar 2011). According to Katakis et al. (2010), recurring contexts are a special type of concept drift where concepts that appeared in the past may recur in the future.

In domains where concepts reappear, it would be a waste of effort to relearn an old concept from scratch for each recurrence (Widmer and Kubat 1996). Even in anomaly detection scenarios, the treatment of recurrence contexts is an important issue (Srivastava 2006). In recurring contexts, instead of forgetting outdated concepts, these concepts should be saved and reexamined at some later time when they can improve the prediction performance in a cost-effective way.

For Masud et al. (2011b) and Al-Khateeb et al. (2012a), a recurring class is a special case of concept-evolution where a class appears in the stream, then disappears for a long time, and appears again. A common error committed by systems that do not address recurring classes is to treat them as a novelty. According to Katakis et al. (2010), this strategy creates undesirable effects, like the increase in the false alarm rate, in the human effort in analyzing the false alarms, additional computational efforts in executing a ND task and in learning a new class that was already learned.

Although the phenomenon of reappearing concepts is very common in real world problems, only few stream-based classification methodologies take it into account (Katakis et al. 2010). In the context of ND in DSs, these problems are addressed by Masud et al. (2011b), Al-Khateeb et al. (2012a), Faria et al. (2013a), de Faria et al. (2015b) and Katakis et al. (2010).

In Masud et al. (2011b), a chunk based approach is proposed to classify new examples in the recurring class context. For such, an ensemble of M classifiers is used to classify new examples and an auxiliary ensemble of size M^A detects recurring classes. When a new example arrives, if it is not labelled as *unknown* by the first ensemble, it is assigned by this ensemble to a known class. If the example is considered *unknown* (i.e. not explained by the first ensemble), the auxiliary ensemble is consulted. If the auxiliary ensemble can explain the example, it is classified by this second ensemble in a recurring class. Otherwise, the example is stored in a buffer and can be later evaluated in a ND task.

According to Al-Khateeb et al. (2012a) chunk-based techniques for ND cannot detect recurrent classes. This happens because if a class disappears for a long time, the ensemble discards all models trained with examples from this class. Thus, when examples of this class reappear, none of the models in the ensemble can classify them, and probably these examples will be identified as novelty patterns.

In order to address this problem, in Al-Khateeb et al. (2012a), the authors propose a new ensemble technique, named class-based ensemble, to classify examples in scenarios involving recurring classes and thus overcome the drawbacks of the chunk based approach. The authors propose the use of c ensembles, one per class, each ensemble is composed by M micro-classifiers (models). If c is the number of classes seen so far in the stream, there are c ensembles. Whenever a new model is trained, it replaces an old model. However, if a class disappears with time, its corresponding ensemble is not deleted. In this case, a class is always considered active, i.e., it never disappears. By doing so, this technique identifies recurring classes as existing classes.

Another algorithm able to deal with ND in DS multiclass classification with recurring contexts is proposed in Faria et al. (2013a) and de Faria et al. (2015b). It represents each problem class by a set of clusters. The clusters that not classify new examples for a long period of time are put in a *sleep memory*, indicating that the concept, which they represent, disappeared during a time period. However, when new valid clusters are detected from examples not explained by the current model, these clusters are assigned to one of three categories: novel concept, extension of a known concept, or recurrence of a concept. The last category occurs when the new cluster is close to a cluster in the *sleep memory*.

A ND application to email filtering using ensemble of classifiers able to deal with recurring contexts is found in Katakis et al. (2010). In the work, the stream is divided into batches of examples. Each batch is used by a transformation function to produce a new conceptual representation model, named conceptual vector. Conceptual vectors describe the concepts present in a particular batch of examples. If two batches are conceptually similar, their conceptual vectors are close. A clustering algorithm groups batches of examples, represented by conceptual vectors, into concepts, which allows the identification of recurring contexts. For every concept detected by the algorithm over the stream, an incremental classifier is created and added to the ensemble. For each cluster, there is an associated classifier. Thus, new examples are always classified using the classifier that corresponds to the cluster of the previous conceptual vector.

7.2 Treatment of outliers

Outliers are data that are isolated, sparse and not present in a representative number (Spinosa et al. 2009). Outlier detection techniques look for examples not following the normal behavior, which may represent undesired patterns. ND algorithms, on the other hand, look for a cohesive and representative set of examples that describe a new emergent concept. ND techniques must address an important issue, the treatment of noise or outliers examples, which can be confused with the appearing of a new concept or a change in the known concepts. According to Rusiecki (2012), outlier detection is more complex in DSs, once they may be noise, which needs to be removed from the data, or produced by a concept drift, representing important changes in the system behavior. Some algorithms address this issue by developing specific techniques to identify noise or outliers.

An approach used by techniques like OLINDDA (Spinosa et al. 2009), ECSMiner (Masud et al. 2011a), MCM (Masud et al. 2010a) and MINAS (Faria et al. 2013a; de Faria et al. 2015b) is to store the examples that are not explained the current model into a temporary memory and label them as *unknown*. The *unknown* examples are later analyzed to decide if they represent a noise or outlier or a novelty pattern.

In OLINDDA (Spinosa et al. 2009), a clustering algorithm is applied to the *unknown* examples. The clusters obtained are evaluated to decide if they represent novelty patterns. Each cluster has its representativeness and cohesiveness evaluated by validation criterion. Clusters with a low evaluation, named invalid clusters, are removed. However their examples stay in the temporary memory, which has a limited capacity. If a new example must be stored but there is no space available, the oldest example is removed. The removed examples have not been used lately. There is a high chance that these examples are noise or outliers. MINAS (Faria et al. 2013a; de Faria et al. 2015b) uses a similar strategy to eliminate noise or outliers. Clusters that are neither cohesive nor representative are removed from the temporary memory.

ECSminer (Masud et al. 2011a), MCM (Masud et al. 2010a) and CLAM (Al-Khateeb et al. 2012a) also applies a clustering algorithm to the *unknown* examples and use the clusters obtained to identify novelty patterns. For each cluster, a q-NSC measure, a combination of cohesion and separation, is computed. If the number of clusters with positive q-NSC value is greater than a threshold, its associated classifier vote for the detection of a new class. If all classifiers votes for the detection of a new class, a novelty is identified. Otherwise, the examples probably represent either noise or outliers or drifts in the known concepts.

7.3 Evaluation in novelty detection

Although several algorithms have been proposed to deal with ND in DS scenarios, little attention has been devoted to the evaluation of their predictive performance.

7.3.1 Experimental methodology

In batch scenarios, the evaluation procedure uses well-known methodologies, like fold-cross-validation, leave-one-out and bootstrap, among others. In DS scenarios, these methodologies must be adapted or new methodologies must be developed. Conventional batch sampling methodologies, like cross-validation, are not applicable to DSs due to the constraints present in this scenario, such as data potentially infinite and non-stationary data distribution (Gama et al. 2013).

For DS classification tasks, two sampling methods have been often used: holdout and prequential (predictive sequential) (Dawid 1984). In the holdout method, the data are divided into training and test sets. The decision model induced by the training set is applied to the test set at regular time intervals. In the prequential method, the induced decision model is applied to each example from the stream. The error rate is computed by the accumulated sum of a loss function between the prediction and true values.

Most of the studies found in the literature for ND in DSs do not discuss the experimental methodology used. Some of these studies use the same methodologies used in batch scenarios, which we believe are not direct applied to DSs, while others create new ad-hoc methodologies, but do not discussed their motivations.

In OLINDDA (Spinosa et al. 2009), the authors use 10-fold-cross-validation. The folds used for training are composed by examples from only one class, the normal class. The test set is composed by examples from the novel classes plus the remaining examples from the normal class. DETECTNOD (Hayat and Hashemi 2010) uses a similar method.

In ECSMiner (Masud et al. 2011a), MCM (Masud et al. 2010a) and CLAM (Al-Khateeb et al. 2012a), the training set for the *offline* phase is composed by the first $init$ windows of data. The remaining elements are used in the *online* phase. MINAS (Faria et al. 2013a, de Faria et al. 2015b) employs the first p elements of the stream for the *offline* training, and the remaining for test. In these four studies, all examples from the test set are used in the computation of the evaluation measures.

7.3.2 Evaluation measures

A few studies employ classical classification measures to evaluate the predictive performance of the investigated classifiers. In Tan et al. (2011), e.g., the Area Under Curve (AUC) measure is used. The instances of the test set are ranked according to their anomaly score. The AUC measure is calculated by using this score and the ground truth. In Krawczyk and Michal (2013), the authors use the accuracy measure, but it is not clear how this measure is computed in the ND context.

In Albertini and de Mello (2007), the evaluation measures precision, recall, and f-measure are used. In the ND context, these measures are calculated according to Eqs. 8, 9, and 10. In the first two equations, $TNov$ is the number of true detected novelties, $DNov$ is the total number of examples detected as novelty and Nc is the total number of examples from the novel classes in the stream.

$$Precision = \frac{TNov}{DNov} \tag{8}$$

$$Recall = \frac{TNov}{Nc} \tag{9}$$

$$FMeasure = 2 * \frac{precision \times recall}{precision + recall} \tag{10}$$

The experiments in Spinosa et al. (2009), Hayat and Hashemi (2010), Masud et al. (2010a), Masud et al. (2011a), Farid et al. (2013), Al-Khateeb et al. (2012a), Al-Khateeb et al. (2012b), Faria et al. (2013a) and de Faria et al. (2015b) use evaluation measures specifically developed for the ND task. However, these measures are more adequate to static scenarios and contexts where ND is considered a one-class classification task. Besides, these measures cannot use the *unknown* label option in the evaluation of classifiers.

The works proposed by Spinosa et al. (2009) and Hayat and Hashemi (2010) treat ND as one-class classification task and use binary classification evaluation measures. Measures like percentage of novel class examples misclassified in the normal class (*Mnew*) and the percentage of normal class examples wrongly labeled as belonging to the novelty or extension classes (*Fnew*), defined by the Eqs. 11 and 12, are used. In these equations, FP is the total number of elements from the normal class wrongly classified as novelty, extension or *unknown*, FN is the total number of examples from the novel classes classified in the normal classes, Nc is the total number of examples from the novel classes in the stream, and N is the total number of examples in the stream. Another measure commonly used is the percent of total misclassification, see Eq. 13. This measure includes, besides FP and FN, the FE, which is the total number of existing class examples misclassified (other than FP).

$$Mnew = \frac{FN * 100}{Nc} \tag{11}$$

$$Fnew = \frac{FP * 100}{N - Nc} \tag{12}$$

$$Err = \frac{(FP + FN + FE) * 100}{N} \tag{13}$$

In Masud et al. (2010a), Masud et al. (2011a), Farid et al. (2013), Al-Khateeb et al. (2012a) and Al-Khateeb et al. (2012b), although ND is treated as a multiclass classification task, binary evaluation measures are used. Binary classification measures are not able to properly evaluate scenarios where the novelty concept is multiclass. In these scenarios, it is not sufficient to classify an example from a novel class as novelty. An error should also computed when examples from different novel classes are classified in the same novelty pattern.

The experiments in Rusiecki (2012) use only artificial, 2D data sets and the classifier is evaluated without the use of evaluation measures. Instead, the authors propose the use of a 2D-graphic with two curves, one representing the signal (the data set) and the other representing the results from the ND task (the timestamps where a novelty was detected).

As another limitation, most of the algorithms for ND found in the literature initially classify the examples not explained by the current model as *unknown*. Later, they can label these examples as novelties. Thus, an important issue to be addressed is how to incorporate these *unknown* examples in the evaluation measures. A similar situation is seen in classifiers with the rejection option (Pillai et al. 2011; Nadeem et al. 2010; Marrocco et al. 2007; Tax and Duin 2008). In these classifiers, an example is rejected if its true class cannot be reliably predicted (Nadeem et al. 2010). It is considered to be better to reject an example than to misclassify it. For these situations, the accuracy and error rate could be calculated regarding either all the examples or only the examples accepted by the classifier.

Regarding the evaluation of ND by unsupervised methods, Faria et al. (2013b) and de Faria et al. (2015a) proposes an evaluation methodology for multiclass ND in DSs. The methodology proposed in Faria et al. (2013b) and de Faria et al. (2015a) is able to associate novelty patterns with problem classes and after that apply multiclass evaluation measures. Also, this methodology proposes to compute a measure to evaluate the *unknown* examples, separately from the other examples.

In Faria et al. (2013a) and de Faria et al. (2015b), the authors proposed that the confusion matrix (see Fig. 7) for ND should not be square. They argued that a new column should be added to the matrix whenever a new novelty pattern is discovered. Each row of this confusion matrix represents one of the problem classes (known and novel classes) and each column represents one of the classes predicted by the ND algorithm. The columns $C_{knw_1}, C_{knw_2}, \ldots, C_{knw_L}$ correspond to the classes learned during the *offline* phase, the columns $N_1, N_2, \ldots$ correspond to the novelty patterns learned in the *online* phase and the last column is the *unknown* (*Unk*) label. For unsupervised algorithms, the novelty patterns detected over time do not have a direct matching with the problem classes. The novelty patterns are sequentially labeled as N_1, N_2, etc. Besides, a given problem class can be associated with one or more novelty patterns and some classes may not be detected by the algorithm.

In order to evaluate this confusion matrix, five aspects should be observed:

1. One class can be represented by two or more novelty patterns, thus it is possible to have more novelty patterns than problem classes;

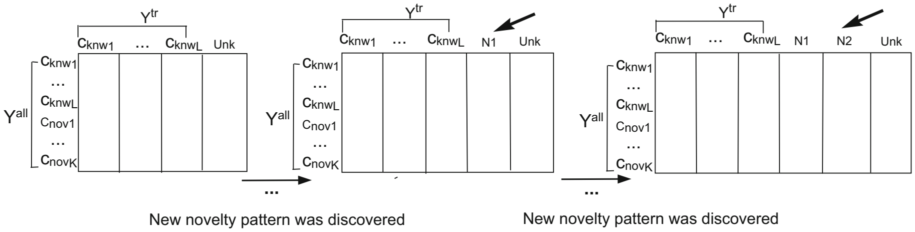

Fig. 7 Evolution of the confusion matrix over time (adapted from Faria et al. (2013b), de Faria et al. (2015a))

2. The algorithm can detect less novelty patterns than the number of novel classes. This may happen if the algorithm did not properly distinguish the examples from all the novel classes;
3. Examples not explained by the current model are labeled by the algorithm as *unknown*;
4. The matrix represents a multiclass scenario, i.e., the computation of accuracy or error measures must consider the different classes learned in the *offline* and *online* phases, which is usually more difficult than to distinguish between the normal and novel concepts;
5. It is necessary to carry out a periodic evaluation of the confusion matrix in order to analyse scenarios that vary over time.

8 Discussion

Although several studies have investigated ND in DSs, they usually present limitations that make their use in real world applications unfeasible. Next, the main limitations and strengths of these studies are presented.

A first limitation is the assumption that all examples used by the model update procedure are labeled. This assumption, adopted in Masud et al. (2010a), Masud et al. (2011a), Al-Khateeb et al. (2012a), Farid and Rahman (2012), Farid et al. (2013), Rusiecki (2012), Krawczyk and Michal (2013), and Tan et al. (2011) is unrealistic. In a DS, data usually flow in very high speed, therefore it is impractical to label all examples. In opposite, in Faria et al. (2013a), de Faria et al. (2015b), Spinosa et al. (2009), and Hayat and Hashemi (2010), the decision model is updated using unlabeled examples. Thus, the novelty patterns detected over the stream are not associated with the real problem class. A more realistic assumption is to consider that the true label of some example are available, and the model update use labeled and unlabeled examples. Despite of the potentiality of this third approach, few studies of the literature adopt this approach (Masud et al. 2010b, 2011c; de Faria et al. 2015b).

A second limitation is to consider ND as a binary classification problem, in which two classes are available: normal and not normal (or novelty). This simplistic view has been adopted by works like Spinosa et al. (2009), Hayat and Hashemi (2010), Albertini and de Mello (2007), Rusiecki (2012), Krawczyk and Michal (2013) and Tan et al. (2011). However, real world problems are composed by different classes and new classes can be detected continuously. For scenarios where the problem known concepts are composed by different classes, works such as Faria et al. (2013a), de Faria et al. (2015b), Masud et al. (2010a), Masud et al. (2011a), Al-Khateeb et al. (2012a), Farid and Rahman (2012), and Farid et al. (2013) can be used. However, Masud et al. (2011a), Al-Khateeb et al. (2012a), Farid and Rahman (2012) and Farid et al. (2013) assume that only one new class appear at time.

A third limitation is to consider every change in the data as concept evolution. Several studies, such as Farid and Rahman (2012), Albertini and de Mello (2007), Rusiecki (2012), Krawczyk and Michal (2013), and Tan et al. (2011) assume as concept evolution the simply presence of one example not explained by the current decision model. However, in some situations, the known concepts are continuously evolving, thus it is important to distinguish between changes in the known concepts and appearing of new concepts. In addition, one example not explained by the current decision model can also indicate the presence of noise or outliers. Thus, approaches as in Masud et al. (2011a) and Al-Khateeb et al. (2012a) work with a time constraint, in which the examples can be classified up to Tl time units after their arrival, in order to detect if these examples can be considered as a novelty pattern or not. Other works, like Spinosa et al. (2009), Faria et al. (2013a), de Faria et al. (2015b) and Hayat and Hashemi (2010), classify as *unknown* the examples not explained by the current decision model. When there are sufficient statistics about these examples, they are classified as concept drift or novelty pattern. Approaches like those proposed in Faria et al. (2013a), de Faria et al. (2015b) and Hayat and Hashemi (2010), which try to find a threshold to separate concept drift from concept evolution, suffer from the difficult of automatically determine this value from the data set.

Still considering the difference between concept drift and concept evolution, a strong challenge is to determine when a ND procedure must be applied. While applying this process whenever a new example arrives is time consuming, to apply it at time intervals requires the definition of the size of the time interval. In Masud et al. (2010a), Masud et al. (2011a), Al-Khateeb et al. (2012a), Faria et al. (2013a), de Faria et al. (2015b) and Spinosa et al. (2009), the technique either asks the user the time interval to execute a ND procedure, or assigns a fixed value to this interval. However, the automatic definition of this value is still an open issue.

A fourth limitation is related to the algorithm used to induce the decision model. While supervised algorithms, such as decision trees (or ensemble of trees) Masud et al. (2010a), Masud et al. (2011a), Farid and Rahman (2012) and Farid et al. (2013), have presented good predictive accuracy, they can be updated only in the presence of labeled examples. On the other hand, the use of unsupervised approaches (Faria et al. 2013a; de Faria et al. 2015b; Spinosa et al. 2009; Masud et al. 2010a; Al-Khateeb et al. 2012a, b; Hayat and Hashemi 2010) based on k-means, present two main drawbacks. It is assumed that the classes are represented by hyperspheres and the number of clusters needs to be defined a priori (input parameter).

A fifth limitation is to ignore recurring contexts. Several approaches developed mechanisms to forget outdated concepts, but few of them treat recurring concepts. In the literature, a common approach is to consider as NP an old concept that reappears. Works that deal with this issue include Faria et al. (2013a), de Faria et al. (2015b), Al-Khateeb et al. (2012a) and Masud et al. (2011b). However, in general, the identification of when a concept needs to be forgotten or a new concept is reappearing involves the definition of user parameters, which can vary according to the used data set.

9 Applications

Recent advances in hardware and software have allowed the acquisition of data in a wide range of applications (Gaber et al. 2005). In general, these data are generated continuously, creating massive data sets. Models induced by DM techniques have been applied to these data to extract new and useful information. Additionally, the development of diverse and powerful sensors has allowed the monitoring many events in real time (Aggarwal 2007). Since these data sets present a dynamic behavior, it is necessary to build models that are not only able to represent these data, but can also evolve over time. In general, these models learn from known scenario about a particular problem, and they need react to changes, forget old concepts and learn new concepts. Among the large range of applications concerning ND in DSs, it is possible to mention anomaly/intrusion detection in networks, credit card fraud detection, fault detection in machinery, video surveillance, etc. Table 3 shows the main ND applications in DSs found in the literature.

The techniques proposed in the literature for ND in DSs are frequently compared by using data from real applications. One of the most common real applications used in these comparisons is the intrusion detection in networks data set. Several studies have used this data

Table 3 Applications of ND in DSs

Applications	References
Intrusion detection	Spinosa et al. (2008), Masud et al. (2011a), Al-Khateeb et al. (2012a), Spinosa et al. (2009), Shyu et al. (2005), Wang et al. (2008), Tan et al. (2011), Farid and Rahman (2012), Farid et al. (2013), de Faria et al. (2015b), Rios et al. (2011), Perdisci et al. (2006),
Forest cover type detection	Masud et al. (2011a) , Al-Khateeb et al. (2012a), Masud et al. (2010a), Tan et al. (2011), Faria et al. (2013a), de Faria et al. (2015b)
Video and image analysis	Singh and Markou (2005), Ramezani et al. (2008), Tavakkoli et al. (2006), Farid et al. (2013), Gaughan and Smeaton (2005)
Text mining	Yang et al. (2002), Masud et al. (2010a)

set, like Spinosa et al. (2008), Masud et al. (2011a), Al-Khateeb et al. (2012a), Spinosa et al. (2009), Tan et al. (2011), Farid and Rahman (2012), Farid et al. (2013), Wang et al. (2008), Rios et al. (2011) and Perdisci et al. (2006). The intrusion detection has been employed to induce classifiers able to identify unauthorized activities in a computer network, separating the normal access from the attacks. In this application, in the training phase, a classifier is induced by using examples from the normal access and a set of known attacks. In the application, *online* phase, the classifier should be able to recognize new types of attack, which may appear any time.

Most of the experiments reported in the literature for this application uses only numerical attributes. However, in Shyu et al. (2005) handle nominal is also used. The most popular public intrusion detection data set comes from the KDD Cup 99. It consists in a simulation of several types of attacks in an military network environment. It is used in Spinosa et al. (2008), Masud et al. (2011a), Al-Khateeb et al. (2012a), Spinosa et al. (2009), Tan et al. (2011) and Shyu et al. (2005). A subset of this data set, named NLS-KDD, is used in Farid and Rahman (2012) and Farid et al. (2013). The NLS-KDD data set was proposed by Tavallaee et al. (2009) to overcome some of the problems found in the KDD Cup 99 data set.

Another classical data set used to evaluate ND algorithms for DSs, available in the UCI repository (Frank and Asuncion 2010), is the Forest Cover Type. This data set represents a real problem of detecting new types of forest according to a set of geospatial attributes. In general, a classifier is induced from a set of forest cover types. The induced model can be later used to identify new types of forest cover. The following studies investigate the performance of ND algorithms in this data set (Faria et al. 2013a; de Faria et al. 2015b; Masud et al. 2010a, 2011a; Al-Khateeb et al. 2012a; Tan et al. 2011).

ND algorithms have also been applied to identify novelties in images from videos (Singh and Markou 2005; Ramezani et al. 2008; Tavakkoli et al. 2006) and image analysis (Farid et al. 2013). Some of these works, e.g. (Ramezani et al. 2008), focus on video-based surveillance, which is an important application for tasks like human security, counter-terrorism and traffic control. Another study, carried out by Gaughan and Smeaton (2005), uses dialogue or closed captions to organize broadcast news retrieval results based on the degree of newness of the topic.

Text mining is another important application for ND in DSs. In Yang et al. (2002), the authors investigate a new system able to classify online documents into a set of predefined topic categories. This system uses topic-conditioned ND to identify new topics. In Masud et al. (2010a), the authors apply the MCM system to ND in Twitter text DSs. For such, the system is trained with a predefined number of topics (classes). The system is used to detect new topics that may appear over the stream and incorporate them its the decision model.

10 Challenges and future work

Even though several ND algorithms have proposed and investigated for a wide range of applications, many issues still need to be addressed. Initially, although experimental methodology is well established in traditional, batch, ML and DM scenarios, in DSs, different methodologies have been adopted by several authors. This lack of standard makes the comparison of these different studies a difficult task. Besides, many important questions concerning experimental methodology in DSs scenarios need to addressed or better explored.

An important issue to be addressed is the adaptation and development of evaluation measures able to provide a meaningful assessment of the performance of ND techniques. Measures are needed for ND in classification and clustering tasks. In scenarios involving concept evolution and those where ND is considered a multiclass problem, the evaluation measures represent an important challenge, especially because the process to detect novelty uses unlabeled examples. Additionally, these evaluation measures need to deal with the presence of examples labeled as *unknown*.

Most of the techniques available on the literature consider that the true label of the training instances will always be available. However, this is not a realistic assumption, since the obtainance of the true label is a time consuming task, many times needing support from a domain specialist. A more realistic view is to ask to the true label of only a subset of the examples, which can be carried out by using active learning. Despite its importance, few studies have addressed this issue.

Several works have treated ND as a one-class problem, where the training examples belong only to a normal concept and whose goal is to discriminate examples from the normal and not normal concepts. In multiclass classification problems, the not normal concept may associated with two or more classes. This one-class approach needs to be adapted to be used in these situations. Some examples of multiclass classification problems fitting this situation are intrusion detection, fault detection, fraud detection, forest cover type detection, spam filter, and text classification.

New approaches able to deal with outliers and noise and a changing environment, where the concepts may evolve either gradually or abruptly, are also necessary. The ability to distinguish a noise or outliers from a novelty concept is crucial to a ND system. While a novelty concept is composed by a set of cohesive and representative examples, noise and outliers are sparse examples, which need to be removed. Moreover, the known concepts may evolve gradually or abruptly and a ND system needs to update its decision model to address these different situations.

Another important issue is the automatic definition of the main parameter settings of ND algorithms. In general, an algorithm proposed in the literature requires a set of parameters whose values should be defined by the user, such as number of classifiers, number of clusters, parameters related to the cluster validation, window size, number of examples for the training phase, threshold to separate novelties from extensions, to mention some. Most of these parameters are very important for the model induction and may have direct impacts on the model performance. In addition, these parameters may vary according to the data set, which makes even more difficult their definition by the user.

The development of new approaches to deal with predictive attributes whose values are not numerical, but categorical, ordinal, hierarchical or assuming other complex data structures, is also an important issue. Several approaches for ND are based on clustering techniques, which can only deal with numerical attributes. In addition, the abundance of data generated from different data sources may lead to complex data structures, which cannot be analyzed by using most of the algorithms from the literature.

Finally, new public domain data sets, real and artificial, presenting several alternative data configurations, would be very useful to assess the performance of the existing and new ND techniques. These new data sets would allow the comparison of these techniques and evaluate their strength and weakness for several data conformations.

Acknowledgments Thanks to European Commission through project MAESTRA (ICT-2013-612944), ERDF through the COMPETE Programme, National Funds through FCT within the project FCOMP - 01-0124-FEDER-022701, and CAPES, CNPq and FAPESP, Brazilian funding agencies.

11 Appendix

The Table 4 lists the principal public online data sets used in the works referred in this survey. They may be downloaded from the following repositories/sites:

- UCI Machine Learning Repository - is a large collection data sets that may be used in different kinds of machine learning tasks, such as clustering, classification, pattern recognition, with a wide variety of different application areas. Available at http://archive.ics.uci.edu/ml.
- KDD Cup Center - annual Data Mining and Knowledge Discovery competition organized by ACM Special Interest Group on Knowledge Discovery and Data Mining. Available at http://www.kdd.org/kddcup/index.php.
- The NSL-KDD Data Set - is a selected collection of records from the KDD Cup 99 which purpose is to overcome some of the problems of the KDD Cup 99 (Tavallaee et al. 2009). In this site, there are also available different sets for training and testing. Avalaible at http://nsl.cs.unb.ca/NSL-KDD/.
- Data Mining Tools Repository - tools developed at the UTD data mining lab, headed by Dr. Latifur Khan. Each tool is part of a project, where it is possible to download related

Table 4 Public data sets used in ND in DS

Data sets	Repository	References
Real		
KDD Cup 99	KDD Cup center	Masud et al. (2011a), Al-Khateeb et al. (2012a), Spinosa et al. (2009), Tan et al. (2011), Shyu et al. (2005), Spinosa et al. (2008), de Faria et al. (2015b)
Forest cover type	UCI machine learning	Masud et al. (2011a), Al-Khateeb et al. (2012a), Masud et al. (2010a), Tan et al. (2011), Faria et al. (2013a), de Faria et al. (2015b)
Image segmentation	UCI machine learning	Farid et al. (2013)
Statlog (Shuttle)	UCI machine learning	Tan et al. (2011)
Spambase	UCI machine learning	Minegishi and Niimi (2011)
NLS-KDD	NLS-KDD	Farid and Rahman (2012), Farid et al. (2013)
Synthetic		
SynC	Data mining tools	Masud et al. (2011a)
SynCN	Data mining tools	Masud et al. (2011a), Al-Khateeb et al. (2012a)
MOA	MOA	Faria et al. (2013a), de Faria et al. (2015b)

published papers, data sets used in the publications and source code of some of the authors algorithms . Available at http://dml.utdallas.edu/Mehedy/.
- MOA - is an open source framework for DS mining. It includes synthetic data generators for classification and clustering tasks.

References

Aggarwal CC (2007) Data streams: models and algorithms. Springer, Berlin

Aggarwal CC (2013) Outlier analysis. Springer, Berlin

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, 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: Proceddings of the IEEE 12th international conference on data mining (ICDM '12). IEEE Computer Society, Washington, DC, USA, pp 31–40

Al-Khateeb TM, Masud MM, Khan L, Thuraisingham B (2012) Cloud guided stream classification using class-based ensemble. In: Proceedings of the 2012 IEEE 5th international conference on computing (CLOUD'12). IEEE Computer Society, Washington, DC, USA, pp 694–701

Albertini MK, de Mello RF (2007) A self-organizing neural network for detecting novelties. In: Proceedings of the 2007 ACM symposium on applied computing (SAC '07), pp 462–466

Aregui A, Denœux T (2007) Fusion of one-class classifiers in the belief function framework. In: Proceedings of the 10th international conference on information fusion, pp 1–8

Bicego M, Figueiredo MAT (2009) Soft clustering using weighted one-class support vector machines. Pattern Recognit 42(1):27–32

Box GEP, Jenkins G (1990) Time series analysis: forecasting and control. Holden-Day, Incorporated, San Francisco

Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41(3):58

Coull S, Branch J, Szymanski B, Breimer E (2003) Intrusion detection: a bioinformatics approach. In: Proceedings of 19th international conference on computer security applications (ACSAC 2003). Nevada, USA, IEEE Computer Society, Las Vegas, pp 24–33

de Faria ER, Goncalves IR, Gama J, Carvalho ACPLF (2015a) Evaluation of multiclass novelty detection algorithms for data streams. Knowl Data Eng, IEEE Trans 27(11):2961–2973. doi:10.1109/TKDE.2015.2441713

de Faria ER, Carvalho ACPLF, Gama J (2015b) MINAS: multiclass learning algorithm for novelty detection in data streams. Data Min and Knowl Discov. doi:10.1007/s10618-015-0433-y

Dawid AP (1984) Statistical theory: the prequential approach (with discussion). J R Stat Soc A 147:278–292

Denis F, Gilleron R, Letouzey F (2005) Learning from positive and unlabeled examples. Theor Comput Sci 348(1):70–83

Dries A, Rückert U (2009) Adaptive concept drift detection. Stat Anal Data Min 2(56):311–327

Elwell R, Polikar R (2011) Incremental learning of concept drift in nonstationary environments. IEEE Trans Neural Netw 22(10):1517–1531

Faria ER, Gama J, Carvalho ACPLF (2013a) Novelty detection algorithm for data streams multi-class problems. In: Proceedings of the 28th symposium on applied computing (ACM SAC'13), pp 795–800

Faria ER, Gonçalves IR, Gama J, Carvalho ACPLF (2013b) Evaluation methodology for multiclass novelty detection algorithms. In: Proceedings of the 2nd Brazilian conference on intelligent systems (BRACIS'13), pp. 19–25

Farid DM, Rahman CM (2012) Novel class detection in concept-drifting data stream mining employing decision tree. In: Proceedings of the 7th international conference on electrical computer engineering (ICECE' 2012), pp 630–633

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. Expert Syst Appl 40(15):5895–5906

Frank A, Asuncion A (2010) UCI machine learning repository. http://archive.ics.uci.edu/ml

Gaber MM, Zaslavsky A, Krishnaswamy S (2005) Mining data streams: a review. SIGMOD Rec 34(2):18–26

Gama J (2010) Knowledge discovery from data streams, 1st edn. CRC Press Chapman Hall, Boca Raton

Gama J, Sebastião R, Rodrigues PP (2013) On evaluating stream learning algorithms. Mach Learn 90(3):317–346

Gaughan G, Smeaton AF (2005) Finding new news: novelty detection in broadcast news. In: Proceedings of the 2nd Asia conference on Asia information retrieval technology (AIRS'05), pp 583–588

Gogoi P, Bhattacharyya D, Borah B, Kalita JK (2011) A survey of outlier detection methods in network anomaly identification. Comput J 54(4):570–588

Han J (2005) Data mining: concepts and techniques. Morgan Kaufmann Publishers Inc., San Francisco

Hayat M, Basiri J, Seyedhossein L, Shakery A (2010) Content-based concept drift detection for email spam filtering. In: Proceedings of the 5th international symposium on telecommunications (IST'10), pp 531–536

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

Hodge V, Austin J (2004) A survey of outlier detection methodologies. Artif Intell Rev 22(2):85–126

Hoffmann H (2007) Kernel PCA for novelty detection. Pattern Recognit 40(3):863–874

Juszczak P, Duin RPW (2004) Combining one-class classifiers to classify missing data. In: Roli F, Kittler J, Windeatt T (eds) Multiple classifier systems. Springer, Berlin, pp 92–101

Katakis I, Tsoumakas G, Vlahavas I (2010) Tracking recurring contexts using ensemble classifiers: an application to email filtering. Knowl Inf Syst 22(3):371–391

Kolter JZ, Maloof MA (2007) Dynamic weighted majority: an ensemble method for drifting concepts. J Mach Learn Res 8:2755–2790

Krawczyk B, Michal W (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

Lee H, Roberts S (2008) On-line novelty detection using the kalman filter and extreme value theory. In: Proceedings of 19th international conference on pattern recognition (ICPR 2008). Tampa, Florida, USA, IEEE, pp 1–4

Li X (2006) Improving novelty detection for general topics using sentence level information patterns. In: Proceedings of the 15th ACM international conference on information and knowledge management (CIKM '06), ACM, pp 238–247

Liu B, Dai Y, Li X, Lee WS, Yu PS (2003) Building text classifiers using positive and unlabeled examples. In: Proceedings of the 3rd IEEE international conference on data mining (ICDM'03), pp 179–186

Lloyd SP (1982) Least squares quantization in PCM. IEEE Trans Inf Theory 28(2):129–137

MacQueen JB (1967) Some methods for classification and analysis of multivariate observations. In: Cam LML, Neyman J (eds) 5th Berkeley symposium on mathematical statistics and orobability, vol 1, pp 281–297

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

Markou M, Singh S (2003b) Novelty detection: a review part 2: neural network based approaches. Signal Process 83(12):2499–2521

Marrocco C, Simeone P, Tortorella F (2007) A framework for multiclass reject in ECOC classification systems. In: Proceedings of the 15th Scandinavian conference on image analysis (SCIA'07), pp 313–323

Marsland S (2003) Novelty detection in learning systems. Neural Comput Surv 3:157–195

Marsland S, Shapiro J, Nehmzow U (2002) A self-organising network that grows when required. Neural Netw 15:1041–1058

Masud M, Gao J, Khan L, Han J, Thuraisingham BM (2011a) 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 (2010a) Addressing concept-evolution in concept-drifting data streams. In: Proceedings of the 10th IEEE international conference on data mining (ICDM'10), pp 929–934

Masud MM, Gao J, Khan L, Han J, Thuraisingham B (2010b) Classification and novel class detection in data streams with active mining. In: Proceedings of the 14th Pacific-Asia conference on advances in knowledge discovery and data mining—volume Part II (PAKDD'10), pp 311–324

Masud MM, Al-Khateeb TM, Khan L, Aggarwal C, Gao J, Han J, Thuraisingham B (2011b) Detecting recurring and novel classes in concept-drifting data streams. In: Proceedings of the 11th IEEE international conference on data mining (ICDM '11), pp 1176–1181

Masud MM, Woolam C, Gao J, Khan L, Han J, Hamlen KW, Oza NC (2011c) Facing the reality of data stream classification: coping with scarcity of labeled data. Knowl Inf Syst 33(1):213–244

Menahem E, Rokach L, Elovici Y (2013) Combining one-class classifiers via meta-learning. In: ACM international conference on information and knowledge management (CIKM 2013), p to be appeared

Minegishi T, Niimi A (2011) Detection of fraud use of credit card by extended VFDT. In: World congress on internet security (WorldCIS'11), pp 152–159

Mitchell TM (1997) Machine learning, 1st edn. McGraw-Hill Inc, New York

Nadeem MSA, Zucker JD, Hanczar B (2010) Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option. In: Workshop and conference proceedings on machine learning in systems biology, vol 8, pp 65–81

Park CH, Shim H (2010) Detection of an emerging new class using statistical hypothesis testing and density estimation. Int J Pattern Recognit Artif Intell 24(1):1–14

Perdisci R, Gu G, Lee W (2006) Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In: Proceedings of the 6th international conference on data mining (ICDM '06), pp 488–498

Perner P (2008) Concepts for novelty detection and handling based on a case-based reasoning process scheme. Eng Appl Artif Intell 22:86–91

Pillai I, Fumera G, Roli F (2011) A classification approach with a reject option for multi-label problems. In: Proceedings of the 16th international conference on image analysis and processing: Part I (ICIAP'11), pp 98–107

Pimentel MA, Clifton DA, Clifton L, Tarassenko L (2014) A review of novelty detection. Signal Process 99:215–249

Ramezani R, Angelov P, Zhou X (2008) A fast approach to novelty detection in video streams using recursive density estimation. In: Proceedings of the 4th international IEEE conference on intelligent systems (IS '08), vol 2, pp 14–2–14–7

Rios G, FILHO RH, Coelho ALC (2011) An autonomic security mechanism based on novelty detection and concept drift. In: Proceeding of the 7th international conference on autonomic and autonomous systems

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

Schölkopf B, Williamson R, Smola A, Taylor JS, Platt J (2000) Support vector method for novelty detection. Adv Neural Inf Process Syst 12:582–588

Schölkopf B, Platt JC, Shawe-Taylor JC, Smola AJ, Williamson RC (2001) Estimating the support of a high-dimensional distribution. Neural Comput 13(7):1443–1471

Shyu ML, Sarinnapakorn K, Kuruppu-Appuhamilage I, Chen SC, Chang L, Goldring T (2005) Handling nominal features in anomaly intrusion detection problems. In: Proceedings of the 15th international workshop on research issues in data engineering: stream data mining and applications (RIDE '05), pp 55–62

Silva JA, Faria ER, Barros RC, Hruschka ER, Carvalho ACPLF, Gama J (2014) Data stream clustering: a survey. ACM Comput Surv 46(1):31

Singh S, Markou M (2005) A black hole novelty detector for video analysis. Pattern Anal Appl 8(1):102–114

Singh S, Markow M (2004) An approach to novelty detection applied to the classification of image regions. IEEE Trans Knowl Data Eng 16(4):396–407

Spinosa EJ, Carvalho ACPLF (2004) SVMs for novel class detection in bioinformatics. In: Proceedings of III Brasilian workshop on bioinformatics (WOB 2004), BrasÃlia, pp 81–88

Spinosa EJ, de A C P L F de Carvalho, Gama J (2008) Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks. In: Proceedings of the 2008 ACM symposium on applied computing (SAC '08), ACM, pp 976–980

Spinosa EJ, Carvalho ACPLF, Gama J (2009) Novelty detection with application to data streams. Intell Data Anal 13(3):405–422

Srivastava A (2006) Enabling the discovery of recurring anomalies in aerospace problem reports using high-dimensional clustering techniques. In: IEEE Aerospace conference

Tan SC, Ting KM, Liu TF (2011) Fast anomaly detection for streaming data. In: Proceedings of the 22th international joint conference on artificial intelligence—volume 2 (IJCAI'11), pp 1511–1516

Tavakkoli A, Nicolescu M, Bebis G (2006) A novelty detection approach for foreground region detection in videos with quasi-stationary backgrounds. In: Proceedings of the 2nd international symposium on visual computing

Tavallaee M, Bagheri E, Lu W, Ghorbani A (2009) A detailed analysis of the kdd cup 99 data set. In: IEEE symposium on computational intelligence for security and defense applications, 2009. CISDA 2009, pp 1–6

Tax DMJ, Duin RPW (2001) Combining one-class classifiers. In: Proceedings of the 2nd international workshop on multiple classifier systems (MCS '01), pp 299–308

Tax DMJ, Duin RPW (2008) Growing a multi-class classifier with a reject option. Pattern Recognit Lett 29(10):1565–1570

Ting KM, Tan SC, Liu FT (2009) Mass: a new ranking measure for anomaly detection. In: Technical report fa2386-09-1-4014, Gippsland School of Information Technology, Monash University

Tsymbal A (2004) The problem of concept drift: definitions and related work. In: Technical report TCD- CS-2004-15, Computer Science Department, Trinity College, Dublin

Vapnik VN (1998) Statistical learning theory, 1st edn. Wiley, New York

Wang H, Fan W, Yu PS, Han J (2003) Mining concept-drifting data streams using ensemble classifiers. In: Proceeding of the 9th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'03), pp 226–235

Wang W, Guan X, Zhang X (2008) Processing of massive audit data streams for real-time anomaly intrusion detection. Comput Commun 31(1):58–72

Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach Learn 23(1):69–101

Yang Y, Zhang J, Carbonell J, Jin C (2002) Topic-conditioned novelty detection. In: Proceedings of the 8th ACM SIGKDD international conference on knowledge discovery and data mining (KDD '02), pp 688–693

Yeung D, Chow C (2002) Parzen-window network intrusion detectors. In: Proceedings of the 16th international conference on pattern recognition, pp 385–388

Yeung D, Ding Y (2003) Host-based intrusion detection using dynamic and static behavioral models. Pattern Recognit 36:229–243

Zhang J, Yan Q, Zhang Y, Huang Z (2006) Novel fault class detection based on novelty detection methods.In: Intelligent computing in signal processing and pattern recognition. Lecture notes in control and information sciences, vol 345. Springer, Berlin, pp 982–987