An Efficient Approach to Detecting Concept-Evolution in Network Data Streams

Sarah M. Erfani[1], Sutharshan Rajasegarar[2], Christopher Leckie[1]

[1]*Department of Computer Science and Software Engineering*
[2]*Department of Electrical and Electronic Engineering*
The University of Melbourne
Parkville, Victoria, Australia

Email: sarahme@student.unimelb.edu.au
r.sutharshan@ee.unimelb.edu.au
caleckie@csse.unimelb.edu.au

Abstract—An important challenge in network management and intrusion detection is the problem of data stream classification to identify new and abnormal traffic flows. An open research issue in this context is concept-evolution, which involves the emergence of a new class in the data stream. Most traditional data classification techniques are based on the assumption that the number of classes does not change over time. However, that is not the case in real world networks, and existing methods generally do not have the capability of identifying the evolution of a new class in the data stream. In this paper, we present a novel approach to the detection of novel classes in data streams that exhibit concept-evolution. In particular, our approach is able to improve both accuracy and computational efficiency by eliminating "noise" clusters in the analysis of concept evolution. Through an evaluation on simulated and benchmark data sets, we demonstrate that our approach achieves comparable accuracy to an existing scheme from the literature with a significant reduction in computational complexity.

Keywords*: anomaly detection; concept-evolution; concept-drift; novel class detection*

1 INTRODUCTION

Data stream mining is the process of extracting hidden knowledge or patterns from continuous sequences of data records. It is mostly used for analyzing records in sensor networks, financial transactions, fraud detection, web searches and computer network traffic. For example, in computer networks there is a need to choose the main types of traffic flows, so that we can detect abnormal and malicious traffic. The challenge in this context is how to model "normal" behavior, given that the traffic patterns are constantly evolving over time.

Specifically, in data stream classification, the temporal behavior and dynamic nature of streams pose a number of open challenges. Two well-studied and challenging problems are "infinite length" and "concept-drift". As far as the first problem is concerned, traditional multi-pass algorithms are not practical, since they require infinite storage to keep all records for training purposes. To tackle this problem, several solutions have been proposed [3], [4]. The second problem, concept-drift, occurs when over time the underlying concepts and statistical properties of data change in an unforeseen way. Therefore, classification models need to be constantly modified and updated. Several solutions to this problem have been proposed [5] [7] [15].

However, another important characteristic of data streams, "concept-evolution", has rarely been addressed in previous studies [1][6][13]. Concept evolution occurs when new classes emerge in records. Most traditional data stream classifiers assume that the number of classes remains constant over time. In the real world, however, evolution of a new class may happen frequently in data patterns. Determining and resolving this problem is of vital importance to certain high security systems such as network traffic analysis, intrusion detection or fault detection as they are very sensitive to any slight change. However, user behavior is constantly changing in networks.

In this paper, we build on the previous work of *Masud et al*. [1] to address these data stream problems. In our study, we address the problems of infinite length, concept-drift and concept evolution. We introduce a step-by-step novel class detection technique, which greatly enhances the performance of previous works and decreases the risk of false-alarms. The technique consists of the following stages. First, in the training procedure, a set of cluster based decision boundaries are built as the basis for anomaly detection. Second, any outliers are detected, based on these decision boundaries. Finally, in three steps we eliminate the outliers to make an informed decision about any novel classes that have emerged in the data stream.

We introduce a new measure, to detect false negative instances and purify the outliers, known as *P-outliers*. This reduces the error rate and helps overcome the problem of concept-drift. In the next step, we employ a measure called the discrete Silhouette Coefficient to determine the cohesion among *P-outliers,* and the separation from other records in the training data. This measure allows us to differentiate the *P-outliers* meeting the requirements of novel instances. Finally, we apply a graph-based technique to identify multiple novel classes. To improve this approach, we introduce a complimentary technique that is able to detect and eliminate any unneeded clusters that arise from the clustering step.

We evaluated our algorithm with a benchmark network dataset. The training and testing data were both selected from the KDD Cup 1999. In comparison to an existing method [1], our results show that our method gives improved accuracy with up to a 25% reduction in computation time.

The rest of the paper is structured as follows. Section 2 gives a general survey of related work in data stream classification and novel class detection. Section 3 gives an overview of our approach. Section 4 describes and illustrates our

algorithm in detail. Section 5 presents our experiments and results. Section 6 concludes with directions for further research.

2 Related Work

Our approach addresses both data stream classification and novel class detection. In the following we discuss several previous studies in these areas.

2.1 Data Stream Classification

Data stream mining has been an active research topic in recent years, and the resulting techniques can generally be classified into two main categories: single model and ensemble model classification. The first category includes studies that, to address concept-drift, incrementally modify and update a single classification model. An example of the single method study is [8]; it proposes a framework, which can adapt micro-clusters quickly to changes in the underlying data stream. Other studies include [7], [11], [15], [16]. The second category includes studies that replace an older model with a high error-rate with a newer, more accurate model. One such study, for example, evaluates the newly trained classifiers in the ensemble based on their expected classification accuracy [5]. Since the ensemble model is more accurate and efficient in handling concept-drift and can efficiently update the current model, we have chosen this approach. Our approach, however, differs by making a more practical method of outlier detection. We also consider novel class detection, which has not been addressed in these previous studies.

2.2 Novel Class Detection

Markou and Singh [10] classify novelty detection approaches into two main categories: parametric and non-parametric. In parametric approaches, a given data distribution is considered and the distribution parameters from the normal data are measured. If any test data does not follow this distribution, it is assumed to be a case of a novel instance [12]. Our approach is non-parametric, meaning it is not constrained to any data distribution. Some examples of non-parametric approaches include kernel-based methods [9] and k-nearest neighbor (k-NN) based approaches [16].

There are two main differences between our approach and the above approaches. First, most of these approaches are based on the assumption that the normal model is static. This indicates, for example, that concept-drift does not occur in the dataset. Our approach, however, is able to identify novel classes even when there is concept-drift. Second, the other approaches only test if there is a major difference between a test instance and the training data. However, in addition to estimating the difference between test data and the normal data, we consider the similarities among the test data.

Spinosa et al. [14] apply a cluster-based technique to data streams to detect novel classes. This technique considers only one normal class and assumes all others as novel classes. As a result, it is not suitable for multi-class detection, where multiple classes are considered as normal. Our approach, however, is more efficient as it does not depend on the number of existing classes. What is more, *Spinosa et al.* [14] consider novel classes as having convex topological shapes, which is not always true in real datasets. A few studies have used a multi-class classifier for detecting novel classes [1][13]. While [13] has not examined the occurrence of multiple novel classes, [1] propose a classification and novel class detection technique. This study not only tackles the challenge of infinite length, concept-drift and concept evolution in data streams, but can also handle the simultaneous appearance of more than one novel class as well.

Masud et al. [1] chose K-means as their clustering method. Although, K-means clustering is one of the popular clustering methods in data-mining and intrusion detection [2], it has several potential limitations in this context. One of the main disadvantages of K-means clustering that significantly affects the performance of these proposed approach is the predefined number of clusters. In this clustering method, the number of clusters should be defined by the user. The value K plays a determining role in producing the final result. Opting for a lower value of K, respective to the operation of K-means, generates larger clusters to cover all the potential novel classes. The larger the sizes of clusters, the higher the probability of covering outliers. On the other hand, a high value of K results in the generation of unneeded clusters that only contain outliers. Finally, both approaches can increase the false-positive rate of the algorithm. In this study, we follow the approach of [1], but improve it in a number of ways.

To alleviate the problems of K-means, we introduce a technique to remove any superfluous generated clusters. In our approach we prefer to define the value of K slightly higher to prevent losing novel instances and decreasing the detection rate. In contrast, we identify those clusters with a low density and high intra-instance distances. These clusters are not genuine novel classes and they are mislabeled as novel classes. By doing so, we give sufficient flexibility to the K-means algorithm to recognize the higher density classes, maximizing novel instance identification. In addition, we remove the extra clusters to minimize the error rate.

Note that identifying ineligible clusters not only reduces the false-positive rate, but also improves the performance of the algorithm and reduces the overhead. The computational complexity of the algorithm is also reduced substantially by eliminating inappropriate clusters.

3 Preliminaries

We use the ensemble model for data stream classification where our stream classifier is an ensemble of M classifiers, $C=\{C_1,...,C_M\}$. The stream is divided into equal size chunks, and each chunk is used to train a *k-NN* based classification model. This input for k-NN is a set of training instances or data records. We employed a semi-supervised K-means clustering algorithm to build K clusters for each chunk. These clusters represent the classification models. A summary of each cluster is saved, comprising the centroid, radius and the number of data instances in the cluster. The centroid of a cluster is the center or mean of all its instances. The radius of a cluster is the distance from the centroid of the cluster to its farthest data-point (belonging to the cluster). However, although the cluster summaries are kept, the raw data-points cannot be discarded since they are required for more accurate novel class detection. In Section

4.2, we examine the trade off between memory space and accuracy.

Before describing our approach, we formally define the key concept of a novel class.

Definition 1 (Decision Boundary). The union of the classification models constitutes the decision boundary, which is used as the basis for outlier detection.

Definition 2 (Novel class and Existing class). A class ς is identified as a novel class if and only if it does not occur in any of the M classifiers. Otherwise, it is an existing class. In other words, when any of the classifiers C_i has been trained with the instances of class ς, the class is indicated as an existing class.

4 Proposed Approach

In this paper we propose our enhanced and time efficient approach to novel class detection in four main stages, as can be seen in Figure 1. At the first stage, the outliers on the fringe of clusters, such as instance t in Figure 2 (Stage 1), are detected and labeled as belonging to existing classes. At the second stage, the regions with a high density of outliers are identified. Next, the approximate number of novel clusters present in the test data is determined, and any ineligible clusters are identified and eliminated. Finally, any closely connected clusters are merged.

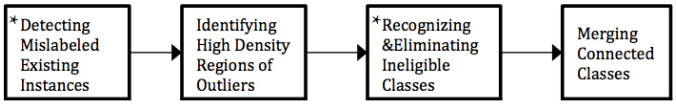

Figure 1. Illustration of the stages of proposed approach. (Steps marked with an asterisk have been introduced by us, while the two other steps were introduced in [1])

The open problem that we address is how to identify new clusters in the test data, given that we do not know a priori how many new clusters are present. As a result, clustering methods such as K-means may lead to unneeded clusters that are artefacts of the algorithm. To detect these unneeded clusters, we have developed a new technique to refine the set of new clusters. In Figure 2, for example, five clusters *A, B, C, D* and *E* are generated at this level. But only *A, B, C* and *D* meet the eligibility condition of novel classes. In contrast, cluster *E* is generated as an artefact from the difficulty in knowing the correct number of clusters a priori. As discussed in Section 4.3 in more detail, this extra cluster is identified as an artefact and removed. In the last stage, once only the eligible novel classes remain, we construct a graph of connected (close) clusters. This graph, as shown in Figure 2 (Stage 4), classifies clusters *A, B* and *C* as a group and that can be merged into a single cluster, whereas the identified cluster D remains as an independent cluster.

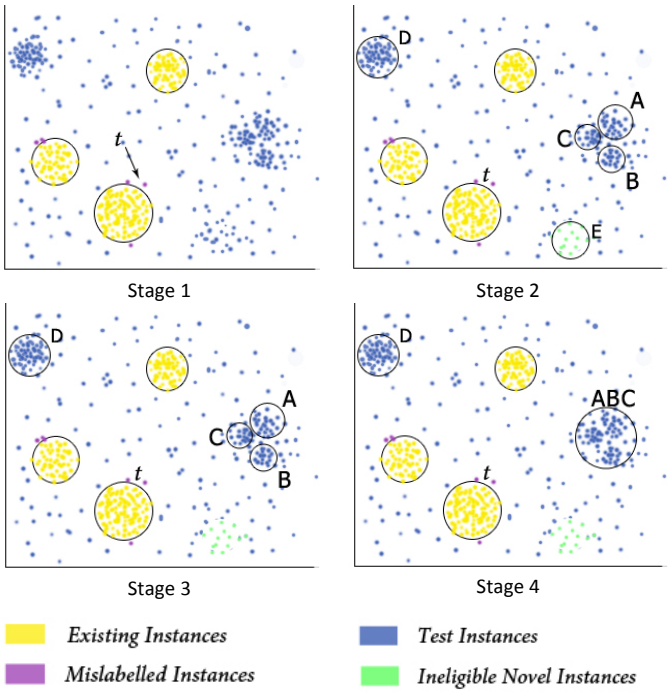

Figure 2. Illustration of the main four stages of the approach

4.1 Identifying and Purifying Outliers

Definition 3 (Existing Instance and Outlier). In order to detect outliers, we require a decision boundary that defines the border between the outliers and the existing classes. The test instances placed inside (or on) the decision boundary are classified as existing class instances. On the other hand, the test data placed outside the decision boundaries of all classes are classified as outliers.

Similarly, a test data record is defined as an outlier if it is outside the radius of all the existing clusters. As a result, the instances that are outside the cluster surface but still very close to them are considered as outliers. For example, see data point t in Figure 2. However, this might occur frequently because of noise or concept-drift, causing normal instances to lie just outside the fringe of a cluster. This results in an increase in the false alarm rate, due to existing instances being labeled as novel data. To address this problem, we define the concept of the *weight* of a test instance.

Definition 4 (Weight). Let t be a test instance, and ς be the nearest cluster with a radius of Cr. Let Td be the distance from instance t to the centroid of the cluster, as illustrated in Figure 3. The *weight* of t is defined as: $weight(t)=e^{Cr-Td}$. If the instance is inside the cluster then $Cr \geq Td$, and $weight(t) \geq 1$ (t_2 in Figure 3). Otherwise, t is an outlier and $weight(t)<1$ (t_1 in Figure 3). The reason for employing this exponential function is that it generates values ranging from 0 to 1 for outlier instances. The outliers farther from the decision boundary have a lower value of *weight* (closer to 0).

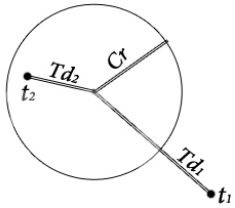

Figure 3. Illustration of centroid distance of two test instances for weight calculation

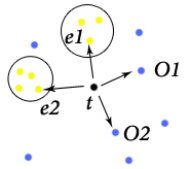

Figure 4. Illustrates the two nearest existing and outlier instances to the instance *t*

In order to identify outliers, first, all the test data is checked for records that are outside the decision boundary. The detected outliers are temporarily stored in a buffer, *TempBuf*. However, as discussed earlier, some of the *TempBuf* instances may be genuine existing instances misclassified as outliers, i.e, because of noise or concept-drift. Then we use the outlier-purifying technique to remove genuine data instances from the buffer, calling the remaining outliers *P-outliers.*

Definition 5 (Outlier-Purifying). The purifying process is based on the *weight*, discussed in Definition 4, of each data instance in the *TempBuf*. We define a threshold *wth=0.7*, where if the weight of a *TempBuf* instance is higher than the *wth* (0.7<weight<1), we label this data as an existing class instance. Otherwise, if its weight is less than or equal to the threshold, the data is identified as a *P-outlier*. Algorithm 1 describes this step in detail:

Algorithm 1. Outlier Purifying

```
for i=1 to size of TempBuf
  calculate weight for instance TempBuf (i)
  if  0<weight(TempBuf (i))≤0.7
    add TempBuf (i) to POutliers
  end if
end for
```

4.2 NOVEL CLASS DETECTION

In data streams, a major source of outliers can be the presence of new clusters due to concept-evolution. Once the outliers in the test data have been detected, we perform several computations on *P-outliers* to recognize the occurrence of novel classes. As the first step, the regions of *P-outliers* with a high density are recognized. As mentioned in [13] "a data-point should be closer to the data-points of its own class (cohesion) and farther apart from the data-points of other classes (separation)". In other words, a novel class is constructed if there is sufficient cohesion among the outliers and enough separation from the existing classes, presented in Figure 2 (Stage 2). To acquire this measure, we use the n-nearest neighbor Silhouette Coefficient (n-SC) [1].

Definition 6 (n-Nearest Neighbors). The *n* nearest *P-outlier* neighbors of an instance, *t*, refers to a set of *n* data-points, among all the outliers, having minimum distance to *t*. Similarly, the nearest *n* existing instances to *t* refers to the *n* closest existing data among all the existing instances of all classes. The nearest outliers and existing instances are illustrated with *o* and *e* in Figure 4, respectively.

Definition 7 (n-SC). Let $D_{np}(x)$ be the average distance from a *P-outlier* *x,* to its *n*-nearest *P-outliers*. $D_{np}(x)$ represents the cohesion among the outlier, *x*, and the *n* closest outlier neighbors. In contrast, $D_{nclass}(x)$ is the average distance of *x* to the *n* nearest existing class instances, and indicates the separation of *x* from them. We define the n-Silhouette Coefficient n-SC as:

$$n\_SC(x) = \frac{D_{np}(\mathrm{x}) - D_{nclass}(\mathrm{x})}{\max\left(D_{np}(\mathrm{x}), D_{nclass}(\mathrm{x})\right)}$$

The definition n-SC returns a value between -1 and 1. A positive result shows that *x* has high cohesion with *P-outliers* and high separation from class instances. It is closer to other outliers and farther away from existing class instances. The outliers having negative n-SC, can be removed from the *P-outliers*, since they are regarded as existing data.

It is to be noted that choosing a higher value of *n* gives us greater certainty and confidence in deciding whether an instance belongs to a novel class. However, if *n* is too large, the algorithm fails to detect a novel class when the number of novel class instances is less than the value of *n*. We find a value for n empirically, as explained in Section 5.

Algorithm 2. Novel Class Detection

```
for i=1 to the size of POutliers
  find n nearest outliers to POutliers (i)
  find n nearest existing instances to POutliers (i)
    if  0<n-SC (POutliers (i))≤ 1
      calculate N-rate for POutliers (i)
    end if
end for
```

In the next step of novel class detection, to distinguish the *P-outlier*s occurring due to the concept-evolution of data streams, we employ a metric called the discrete Gini Coeffcient. We have found that if the result of this metric is higher that a given value then we can determine the emergence of concept-evolution in our stream.

Definition 8 (N-rate). Novelty rate (N-rate) is a metric to determine the likelihood of an outlier belonging to a novel class. It consists of two parts. In the first part the distance of an outlier to its nearest existing class instance is measured. The higher value, as expected, indicates a greater separation. The second part calculates the distance of the outlier to its closest outliers. It should be noted that the N-rate is only computed for instances with positive n-SC value, i.e, high cohesion with *P-outliers* and high separation from existing

instances. The N-rate for each *P-outlier* is obtained by using the calculated weight and n-SC of that instance in Definitions 4 and 7 respectively. The term *minweight* in the N-rate equation indicates the minimum weight of all the *P-outliers*, which have positive n-SC. The value of N-rate(x) is within the range of [0,1], and a higher value implies a greater likelihood of *x* being a novel data instance.

$$N_{rate(x)} = \frac{1 - \text{weight(x)}}{1 - \text{minweight}} \times n\_SC(x)$$

Analyzing the distribution of N-rate values provides a clearer image of the novelty of the *P-outliers*. We discretise the values of N-rate(x) into *l* equal intervals, and generate a cumulative distribution function (CDF) denoted as *s* of the N-rate values where $\{y_i ; i=1,\ldots,m\}$. Now we apply the discrete Gini Coefficient metric *GC(s),* for a random sample of y_i, to make the final decision about the novelty of each *P-outlier*.

$$GC(s) = \frac{1}{m} + (m + 1 - 2(\frac{\sum_{i=1}^{m}(m + 1 - i)y_i}{\sum_{i=1}^{m} y_i})$$

Let y_i be the CDF value for the *i*th interval ($1 \leq i \leq m$). We categorize the behavior of GC(s) into three main cases:

I. $GC(\text{s}) = 0$, this occurs when all N-rate values are generally very low and lie in the first interval. Then y_i would be equal to 1 for all the intervals. This shows that all the outliers belong to the existing class but are misclassified as outliers.

$$GC(\text{s}) = \frac{1}{m} + (m + 1 - 2\left(\frac{\sum_{i=1}^{m}(m+1-i)1}{\sum_{i=1}^{m}1}\right) = 0$$

II. $GC(\text{s}) = \frac{m-1}{m}$, in contrast with case I, this occurs when all N-rate values are generally high and lie in the last interval. Therefore, all the outliers belong to the novel class. Then y_m=1 and y_i=0 for all $i < m$. This implies that all the outliers belong to the novel class. After simplification, GC(s) becomes:

$$GC(\text{s}) = \frac{1}{m} + (m + 1 - 2\left(\frac{1}{1}\right) = \frac{m-1}{m}$$

III. $G(\text{s}) = \frac{m-1}{3m}$, this is occurs when the distribution is a mixture of, for example, some novel class instances and noise or concept-drift. Therefore, the distribution of N-rate values is relatively uniform across all the intervals. The value of y_i for all intervals is $y_i = i/m$. After simplification, GC(s) becomes:

$$GC(\text{s}) = \frac{1}{m} + (m + 1 - 2\left(\frac{\sum_{i=1}^{m}(m+1-i)i}{\sum_{i=1}^{m} i}\right) = \frac{m-1}{3m}$$

To summarize, by examining the three possible cases with the aim of identifying the evolution of a novel class, we came up with a threshold for the Gini Coefficient. If $GC(s) > \frac{m-1}{3m}$ then a novel class is declared and the *P-outlier* instances are labeled as novel class instances [1].

4.3 Detection of Multiple Simultaneous Novel Classes

One of the challenges of novel class detection is recognizing the evolution of more than one class at the same time. The occurrence of multiple new classes in data streams, such as Twitter [1], is a common scenario. This poses problems since it needs to be performed in an unsupervised environment. To solve this problem, we construct a graph to define the connected components (nodes), which correspond to a subset of closely related novel clusters.

Definition 9 (Connected components). A connected component represents a set of novel classes, such that the cohesion between all novel class instances is higher than the distance between each pair of the classes (separation).

Prior to identifying the connected components, the novel instances, detected in Section 4.2, should be classified as different novel classes. Since defining the required number of novel classes (clusters) has a significant impact on the accuracy of the final result, we use a combination of two methods. First, we use K-means clustering to generate the clusters, and opt for an a choice of K value that over estimates the true number of clusters, which is defined as follows:

$$K_n = \frac{K \times NI}{ChSize}$$

where *NI* is the number of novel instances, *K* is the initial number of clusters and *ChSize* the chunk size. K_n indicates the number of required clusters. After constructing the novel classes using K-means with K_n, we measure the density of each cluster. The density of a novel class can be measured by dividing the number of cluster instances by the mean distance among them.

$$NDensity = \frac{NSize}{mean\ D_{NI}}$$

In general, K_n may not be precise and may not return the exact number of novel clusters. Inevitably, some extra clusters are generated containing no or only a few novel instances along with some outliers or existing instances. This leads to an increase in the false alarm rate. To tackle this problem, we define a threshold for the cluster density. If the cluster density is lower than the threshold, $th_{Dens}=0.5$ and the number of its instances is less than 10, we eliminate that cluster and its instances from the novel classes. Only the novel clusters with high density remain, which we call eligible classes or *e-classes*.

Once the eligible novel classes have been identified, we build a graph G= (V, E) to detect the connected components, where each of the *e-classes* is considered as a vertex of G and is connected to the nearest neighbor having a minimum centroid distance. To do so, we use the Silhouette Coefficient equation:

$$c\_SC = \frac{d.nn - \mu.inst}{\max(d.nn, \mu.inst)}$$

where *d.nn* represents the centroid distance of an *e-class* from its nearest neighbor, and $\mu.inst$ is the mean distance of all the instances belonging to the *e-class* from its centroid. A low value of c-SC (closer to zero) indicates that the *e-class* is not a tight cluster and is close to its nearest neighbor. On the other hand, a higher c-SC (close to 1) indicates that the *e-class* is a tight cluster and far from its closest cluster. We introduce a threshold th_c=0.8 such that if the value of c-SC is less than th_c we add an edge to G, showing that the two clusters are not separable.

Once the graph G is constructed with all the connected nodes, the centroid distance between each pair of nodes is measured. Let η_1 and η_2 be two nodes of the graph G, where $D_{\eta 1}$ and $D_{\eta 2}$ are the mean distance of their instances. We merge the two nodes if the sum of $D_{\eta 1}$ and $D_{\eta 2}$ is twice as much as the distance between the centers of the two nodes. Finally, the final number of novel classes is equal to the number of merged clusters plus the unmerged clusters. For example, in Figure 2 (Stage 4), clusters A, B and C are merged into one novel class and cluster D is defined as a separate class. Algorithm 3 describes our method in detail:

Algorithm 3. Novel Class Detection

```
Calculate the number of required novel clusters, K_n
for i=1 to size of K_n
  if NDensity of cluster(i)>th_Dens & # cluster instances >10
    define cluster(i) as an e-class
  end if
end for
for j=1 to the number of e-classes
  find the connected components to class(j)
end for
for p=1 to the number of e-classes
  for q=1 to the number of connected components to the e-
  classes(p)
    if D_η(q)+ D_η(p)>2×(centroid distance between class(p)
    and class(q))
      merge the clusters
    end if
  end for
end for
```

5 Evaluation

The aim of our evaluation is to measure the accuracy and efficiency of our proposed scheme in terms of detecting multiple simultaneous novel classes. In our experiment we use synthetic datasets to measure the accuracy of our algorithm and visualize its performance. In addition, we apply a benchmark dataset to evaluate the applicability of our approach to a real-world networking scenario.

We implemented our novel class detection algorithm in Matlab and used the following parameter settings: K (number of initial clusters)=3, C (number of classifiers)=3, *n* (minimum number nearest neighbors)=12. With these settings we examined our approach using various amounts of data. For each of the experiments, the false positive and false negative rates were calculated.

Since the aim of the algorithm is to detect novel clusters, a 'positive' is anything labeled by the algorithm as a member of a novel class. A 'negative' is anything labeled by the algorithm as outlier or a member of an existing class. Then, a false positive (FP) occurs when an outlier or a member of an existing class is labeled as a member of a novel class. A false negative (FN) occurs if a member of a novel class is labeled by the algorithm as an outlier or an existing class instance.

True positive (TP) and true negative (TN) represent the number of correctly detected novel instances, and outlier or existing class members, respectively.

The false positive rate (FP-rate) is the ratio of the number of false positive instances generated by the algorithm to the total number of true negatives and false positives.

- FP-rate = FP/(TN+FP)

Similarly, false negative rate (FN-rate) is the ratio of the number of false negative data instances to the total number of true positives and false negatives.

- FN-rate = FN/(TP+FN)

The detection rate is:

- DR=1- FN-rate

5.1 Synthetic Dataset

Our synthetic dataset consists of three classes. The first class is used to build some initial (existing) classes, as can be seen in Figure 2. Another set of data is used as outliers, and is a low-density dataset dispersed over the data range. The last dataset is defined as 'normal' instances, and includes some classes with different densities and distances. To evaluate our approach, first the initial instances are applied to generate the classifiers and decision boundary. Next, the algorithm is applied to the unlabeled outlier and normal data. We applied both our approach and the method of *Masud et al.*[1] to the data, and the resulting FP, DR and processing time are shown in Table 1. The ROC curves in terms of the resulting FN and DR of the two methods are presented in Figure 5. The value of the Area Under the ROC curve (AUC) obtained from the approach of *Masud et al.*[1] is 0.79, while, the AUC of our approach is 0.88. What is more, in comparison, the average of the computation time of the two approaches reveals an almost 25% improvement when using our approach.

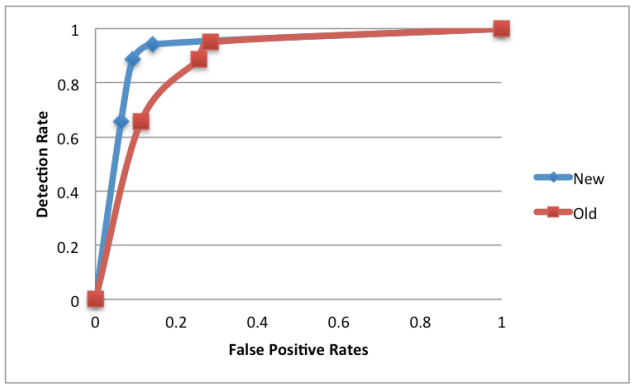

Figure 5. ROC curves of ourapproach (New) and approach of [1] (Old) on synthetic data set.

5.2 KDD CUP Dataset

The KDD Cup 1999 dataset was derived from the 1998 DARPA Intrusion Detection Evaluation Program organized by the MIT Lincoln Labs. In this data, a military LAN was

emulated along with various types of attacks over a period of nine weeks. The first seven weeks of traffic generate the training dataset with about 4.9 million connections. The two other weeks consist of a test dataset with about 300,000 connections, which also includes several different types of attacks compared with the training data. The entire dataset was gathered from the binary TCP dump and converted into sets of connection records. We have extracted a subset of this data set for our evaluation. Each of the records contains 82 various features and a label, defining whether the data is normal or an attack.

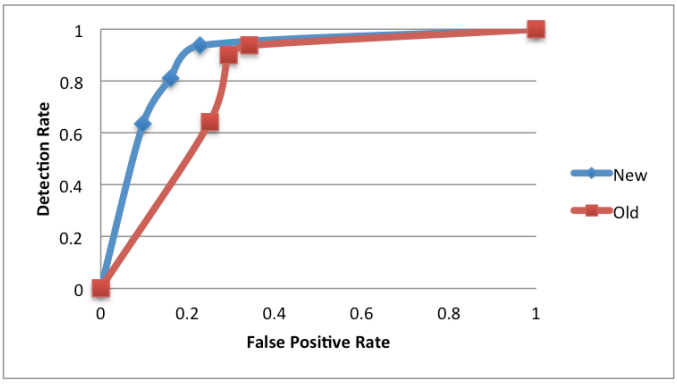

Figure 6. ROC curves of our approach (New) and approach of [1] (Old) on KDD Cup dataset.

The ROC curves resulting from the KDD Cup dataset for the two approaches are presented in Figure 6. The value of the AUC obtained from the approach of *Masud et al.*[1] is 0.87, whereas the AUC resulting from our approach is 0.92. Similar to the synthetic datasets, the average computational time of the two approaches reveals an almost 28% improvement in our approach.

		FP-rate		DR		Process Time*	
	Number of Test Data	New	Old	New	Old	New	Old
Synthetic	2200	0.067	0.112	0.657	0.657	394	448
	2200	0.091	0.198	0.774	0.801	487	563
	1200	0.092	0.254	0.887	0.887	56	96
	2400	0.141	0.283	0.942	0.95	434	612
Average						342.75	429.75
KDD Cup	2380	0.097	0.253	0.634	0.642	13,107	20,590
	2380	0.161	0.262	0.812	0.808	20,243	28,433
	1180	0.200	0.294	0.874	0.901	788	903
	1180	0.227	0.325	0.937	0.937	764	785
Average						8,726	12,678

Table 1. Results of our approach (New) and approach of [1] (Old) on synthetic and KDD Cup datasets. * Process time in seconds.

6 Conclusion

In this paper, we proposed a novel class detection scheme, which is also capable of identifying multiple classes simultaneously. We have evaluated the scheme on both simulated and benchmark datasets and shown that our approach is able to achieve a comparable detection rate, while reducing the false positive rate, compared to an existing approach. In addition, this accuracy was achieved with a reduction in the required computation time.

7 Acknowledgement

We thank the MIT Lincoln Lab for the use of the 1999 KDD Cup datasets. This work was supported in part by the Australian Research Council.

References

[1] M. M. Masud, J. Gao, L. Khan, J. Han, Q.Chen, C.Aggarwal and B. M. Thuraisingham. "Addressing concept-evolution in concept-drifting data streams". *IEEE International Conference on Data Mining*, 1550-4786/10,pages 929-934, Dec 2010.

[2] V. Chandola, A. Banerjee, and V. Kumar. "Anomaly Detection: A Survey". In *ACM Computing Surveys (CSUR),* Volume 41 Issue 3, July 2009.

[3] G. Hulten, L. Spencer, and P. Domingos. "Mining time-changing data streams". In *ACM SIGKDD*, pages 97–106, 2001.

[4] Y. Yang, X. Wu, and X. Zhu. "Combining proactive and reactive predictions for data streams". In *ACM SIGKDD*, pages 710–715, 2005.

[5] H. Wang, W. Fan, P. S. Yu, and J. Han. "Mining concept-drifting data streams using ensemble classifiers". In *ACM SIGKDD*, pages 226–235, 2003.

[6] M. M. Masud, J. Gao, L. Khan, Jiawei, H. and B. M. Thuraisingham. "Classification and Novel Class Detection inConcept-Drifting Data Streams under Time Constraints". *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol.23 no.6, pages 859-874, Jun 2011.

[7] M. Scholz and R. Klinkenberg. "An ensemble classifier for drifting concepts " . In *Proc. Second International Workshop on Knowledge Discovery in Data Streams (IWKDDS)*, pages 53–64, Porto, Portugal, Oct 2005.

[8] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. "A framework for on-demand classification of evolving data streams". *IEEE Trans. on Knowl.& Data Engg.(TKDE)*, 18(5):577–589, 2006.

[9] T. Ahmed, M. Coates, and A. Lakhina. " Multivariate online anomaly detection using kernel recursive least squares". *Proc. IEEE Infocom, Anchorage,* Alaska, May 2007.

[10] M. Markou and S. Singh. "Novelty detection: A review-part 1: Statistical approaches, part 2: Neural network based Approaches". In *Signal Processing*, volum 83, Issue 12, pages 2499-2521, Dec 2003.

[11] H. Wang, W. Fan, P. S. Yu, and J. Han. "Mining concept-drifting data streams using ensemble classifiers". In *ACM SIGKDD*, pages 226–235, 2003.

[12] A. Nairac, T. Corbett-Clark, R. Ripley, N. Townsend, and L. Tarassenko. "Choosing an appropriate model for novelty Detection" . In *Proc. International Conference on Artificial Neural Networks*, pages 117–122, 1997.

[13] M. M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham. "Integrating novel class detection with classification for concept-drifting data streams". In *ECML PKDD*, volume 2, pages 79–94, 2009.

[14] E. J. Spinosa, A. P. de Leon F. de Carvalho, and J. Gama. "Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks". In *ACM SAC*, pages 976–980, 2008.

[15] J. Kolter and M. Maloof. " Using additive expert ensembles to cope with concept drift". In *ICML*, pages 449–456, 2005.

[16] Y. Yang, J. Zhang, J. Carbonell, and C. Jin. "Topic-conditioned novelty detection". In *Proc. ACM SIGKDD*, pages 688–693, 2002.