Addressing Concept-Evolution in Concept-Drifting Data Streams

Mohammad M. Masud [†1], Qing Chen [†2], Latifur Khan [†3], Charu Aggarwal [⋆4]
Jing Gao [‡5], Jiawei Han [‡6] and Bhavani Thuraisingham [†7]

[†] *Department of Computer Science, University of Texas at Dallas*
{[1] *mehedy,* [2] *qingch,* [3] *lkhan,* [7] *bhavani.thuraisingham*}*@utdallas.edu*

[⋆] *IBM T. J. Watson Research Center*
Yorktown Heights, New York
[4] *charu@us.ibm.com*

[‡] *Department of Computer Science*
University of Illinois at Urbana-Champaign
{[5] *jinggao3,* [6] *hanj*}*@cs.uiuc.edu*

***Abstract*—The problem of data stream classification is challenging because of many practical aspects associated with efficient processing and temporal behavior of the stream. Two such well studied aspects are infinite length and concept-drift. Since a data stream may be considered a continuous process, which is theoretically infinite in length, it is impractical to store and use all the historical data for training. Data streams also frequently experience concept-drift as a result of changes in the underlying concepts. However, another important characteristic of data streams, namely, concept-evolution is rarely addressed in the literature. Concept-evolution occurs as a result of new classes evolving in the stream. This paper addresses concept-evolution in addition to the existing challenges of infinite-length and concept-drift. In this paper, the concept-evolution phenomenon is studied, and the insights are used to construct superior novel class detection techniques. First, we propose an adaptive threshold for outlier detection, which is a vital part of novel class detection. Second, we propose a probabilistic approach for novel class detection using discrete Gini Coefficient, and prove its effectiveness both theoretically and empirically. Finally, we address the issue of simultaneous multiple novel class occurrence, and provide an elegant solution to detect more than one novel classes at the same time. We also consider feature-evolution in text data streams, which occurs because new features (i.e., words) evolve in the stream. Comparison with state-of-the-art data stream classification techniques establishes the effectiveness of the proposed approach.**

***Keywords*-data stream; concept-evolution; novel class; outlier**

I. Introduction

The problem of data stream classification has been widely studied in the literature over the last decade. The dynamic and evolving nature of data streams pose special challenges to the development of effective and efficient algorithms. Two of the most challenging characteristics of data streams are its *infinite length* and *concept-drift*. Since a data stream is a high volume phenomenon, which can be considered infinite in length, it is impractical to store and use all the historical data for training. Several incremental learners have been proposed to address this problem [4], [10]. In addition, concept-drift occurs in the stream when the underlying concepts of the stream change over time. A variety of techniques have also been proposed in the literature for addressing concept-drift [5], [9] in data stream classification.

However, another significant characteristic of data stream is that of *concept-evolution*. This occurs when new classes evolve in the data. For example, consider the problem of intrusion detection in a network traffic stream. If we consider each type of attack as a class label, then concept-evolution occurs when a completely new kind of attack occurs in the traffic. Another example is the case of a text data stream, such as that occurring in Twitter. In this case, new topics (classes) may frequently emerge in the underlying stream of text messages. The problem of concept-evolution is addressed in only a very limited way by the currently available data stream classification techniques. We investigate this problem in this paper, and propose improved solutions. Our current work also addresses the feature-evolution problem in data streams, such as text streams, where new features (words) emerge and old features fade away.

Our previous work [6] addresses the novel class detection problem in the presence of concept-drift and infinite length. In this technique, an ensemble of models is used to classify the unlabeled data, and detect novel classes. The novel class detection process consists of three steps. First, a *decision boundary* is built during training. Second, test points falling outside the decision boundary are declared as filtered outliers, or F-$outlier$s. Finally, the F-$outlier$s are analyzed to see if there is enough *cohesion* among themselves (i.e., among the F-$outlier$s) and *separation* from the training instances. If such cohesion and separation are found, then the F-$outlier$s are identified as novel class instances. However, this approach observed high false alarm rates on some datasets. Besides, it did not distinguish among more than one novel classes. In this paper, we propose an improved technique to reduce both false alarm rate and increase detection rate. Our framework also allows for methods to distinguish among two or more novel classes.

We claim three major contributions in novel class detection for data streams. First, we propose a flexible decision boundary for outlier detection by allowing a slack space

outside the decision boundary. This space is controlled by a threshold, and the threshold is adapted continuously to reduce the risk of false alarms and missed novel classes. Second, we apply a probabilistic approach to detect novel class instances using the discrete Gini Coefficient. With this approach, we are able to distinguish different causes for the appearance of the outliers, namely, noise, concept-drift, or concept-evolution. Finally, we apply a graph-based approach to detect the appearance of more than one novel classes simultaneously, and separate the instances of one novel class from the others. To the best of our knowledge, this is the first work that proposes these advanced techniques for novel class detection and classification in data streams. We apply our technique on a number of benchmark data streams including Twitter messages, and outperform the state-of-the-art classification and novel class detection techniques.

The rest of the paper is organized as follows. Section II discusses the related works in data stream classification and novel class detection. Section III describes the proposed technique. Section IV then reports the datasets and experimental results, and Section V concludes with directions to future works.

II. Related work

Most of the existing data stream classification techniques are designed to handle the efficiency and concept-drift aspects of the classification process [1], [3]–[5], [9]–[11]. Each of these techniques follow some sort of incremental learning approach to tackle the infinite-length and concept-drift problems. There are two variations for this incremental approach. The first approach is a single-model incremental approach, where a single model is dynamically maintained with the new data. For example, [4] incrementally updates a decision tree with incoming data, and the method in [1] incrementally updates micro-clusters in the model with the new data. The other approach is a hybrid batch-incremental approach, in which each model is built using a batch learning technique. However, older models are replaced by newer models when the older models become obsolete ([2], [5], [9], [10]). Some of these hybrid approaches use a single model to classify the unlabeled data (e.g. [10]), whereas others use an ensemble of models (e.g. [5], [9]). The advantage of the hybrid approaches over the single model incremental approach is that the hybrid approaches require much simpler operations to update a model.

The other category of data stream classification technique deals with concept-evolution, in addition to addressing infinite-length and concept-drift. Spinosa et al. [7] apply a cluster-based technique to detect novel classes in data streams. However, this approach assumes only one "normal" class, and considers all other classes as "novel". Therefore, it is not directly applicable to multi-class data stream classification, since it corresponds to a "one-class" classifier. Furthermore, this technique assumes that the topological shape of the normal class instances in the feature space is convex. This may not be true in real data. Our previous work [6] proposes a classification and novel class detection technique that is a multi-class classifier and does not require classes to have convex shape. In this paper, we extend this work by proposing flexible and dynamically adaptive decision boundary for outlier detection, as well as methods for distinguishing more than one novel classes. Experiments with real datasets prove the effectiveness of our approach.

III. Novel class detection: proposed approach

In this paper, we propose three different improvements over the existing novel class detection technique, i.e.: i) *Outlier detection using adaptive threshold*, ii) *Novel class detection using Gini Coefficient*, and iii) *Simultaneous multiple novel class detection*. These are discussed in the following subsections. First, we briefly discuss the existing novel class detection technique.

Our stream classifier is an ensemble of L classification models, $M = \{M_1, ..., M_L\}$. A class c is defined as a *novel class* if none of the classifiers M_i has been trained with c. Otherwise, if one of the classifiers M_i has been trained with c, then it is an *existing class*. The data stream is divided into equal sized chunks. We train a k-NN based classifier with each labeled chunk. Here, K clusters are built using a semi-supervised K-means clustering, and the cluster summaries (mentioned as *pseudopoint*s) of each cluster are saved. The summary contains the *centroid*, *radius*, and *frequencies* of data points belonging to each class. The radius of a pseudopoint is equal to the distance between the centroid and the farthest data point in the cluster. The raw data points are discarded after creating the summary. These pseudopoints constitute the *classification model*. This classifier replaces one of the existing classifiers (usually the highest error classifier) in the ensemble. Besides, each pseudopoint corresponds to a hypersphere having center at the centroid and radius equal to the radius of the psuedopoint. The union of the hyperspheres constitute the *decision boundary* for the classifier. If a test instance x is inside the decision boundary of any model in the ensemble, then it is classified as an existing class instance using majority voting. Otherwise, if x is outside the decision boundary of all the models, it is considered an $F\text{-}outlier$, and it is temporarily stored in a buffer Buf. When there are enough outliers in Buf, we invoke a novel class detection procedure to check whether the outliers actually belong to a novel class. If a novel class is found, the $F\text{-}outlier$s are tagged as novel class instance.

The main assumption in novel class detection is that any class of data follows the *property* that "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*)" [6]. The novel class detection procedure measures the cohesion among the $F\text{-}outlier$s in Buf, and separation of the $F\text{-}outlier$s from the existing class instances by computing

a unified measure of cohesion and separation, which we call q-Neighborhood Silhouette Coefficient or q-NSC, for short, as follows: $q\text{-}NSC(x) = \frac{\bar{D}_{c_{min},q}(x) - \bar{D}_{c_{out},q}(x)}{max(\bar{D}_{c_{min},q}(x), \bar{D}_{c_{out},q}(x))}$ where $\bar{D}_{c_{out},q}(x)$ is the mean distance from $F\text{-}outlier$ x to its q nearest $F\text{-}outlier$ instances, and $\bar{D}_{c_{min},q}(x)$ is the mean distance from x to its q-nearest existing class instances. The expression q-NSC yields a value between -1 and +1. A positive value indicates that x is closer to the $F\text{-}outlier$ instances (more cohesion) and farther away from existing class instances (more separation), and vice versa. The q-NSC(x) value of an $F\text{-}outlier$ x is computed separately for each classifier $M_i \in M$. A *new class* is declared if there are at least q' ($> q$) $F\text{-}outlier$s having positive q-NSC for all classifiers $M_i \in M$.

A. Outlier detection using adaptive threshold

A test instance is identified as an $F\text{-}outlier$ if it is outside the radius of all the pseudopoints in the ensemble of models. Therefore, if a test instance is outside the hypersphere of a pseudopoint, but very close to its surface, it will still be an outlier. However, this case might be frequent due to concept-drift or noise, i.e., existing class instances may be outside and *near* to the surface of the hypersphere. As a result, the false alarm rate (i.e., detecting existing classes as novel) would be high. In order to solve this problem, we follow an adaptive approach for detecting the outliers. We allow a slack space beyond the surface of each hypersphere. If any test instance falls within this slack space, it is considered as existing class instance. This slack space is defined by a threshold, OUTTH. We apply an adaptive technique to adjust the threshold. First, we explain how the threshold is used.

Using OUTTH: Let x be a test instance, and h be the nearest pseudopoint of x in model M_i, with radius r. Let d be the distance from x to the centroid of h. We define weight(x) as follows: $weight(x) = e^{r-d}$. If $r >= d$, then x is inside (or on) the hypersphere and weight(x) $>= 1$. Otherwise, x is outside the hypersphere and weight(x) < 1. Note that if x is outside the hypersphere, then weight(x) is within the range [0,1). The main reason for using this exponential function is that the function produces values within the range [0,1), which provides a convenient normalized value. The value of OUTTH is also within [0,1). Now, if weight(x) $>=$ OUTTH, then we consider x as existing class instance, otherwise, x is considered as an outlier. If x is identified as an outlier for all models $M_i \in M$, then x is considered as an $F\text{-}outlier$. *Adjusting OUTTH:* Initially, OUTTH is initialized with OUTTH_INIT value. We set OUTTH_INIT to 0.7 in our experiments. To adjust OUTTH, we examine the latest labeled instance x. If x had been a false-novel instance (i.e. existing class but misclassified as novel class), then it must have been an outlier. Therefore, weight(x) $<$ OUTTH. If the difference OUTTH - weight(x) is less than a small constant ϵ, then we consider x as a *marginal false-novel* instance. If x is a marginal false-novel instance, then we increase the slack space so that future instances like this do not fall outside the decision boundary. Therefore, OUTTH is decreased by a small value (ϵ), which effectively increases the slack space. Conversely, if x is a *marginal false-existing* instance , then x is a novel class instance but was falsely identified as an existing class instance by a narrow margin. Therefore, we need to decrease the slack space (increase OUTTH). This is done by increasing OUTTH by ϵ. The *marginal* constraint is imposed to avoid drastic changes in OUTTH value. Figure 1 illustrates the concept of OUTTH, marginal false-novel and marginal false-existing instances.

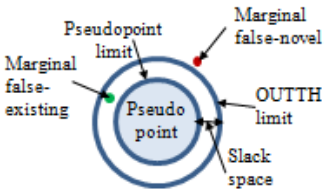

Figure 1. Illustration of a slack space outside the decision boundary

B. Novel class detection using Gini Coefficient

The $F\text{-}outlier$s detected during the outlier detection phase may occur because of one or more of the three different reasons: noise, concept-drift, or concept-evolution. In order to distinguish the $F\text{-}outlier$s that occur because of concept-evolution only, we compute a metric called discrete Gini Coefficient of the $F\text{-}outlier$ instances. We show that if the Gini Coefficient is higher than a particular threshold, then we can be confident of the concept-evolution scenario. After detecting the $F\text{-}outlier$ instances using the OUTTH value discussed in the previous section, we compute the q-NSC(x) value for each $F\text{-}outlier$ instance x. If the q-NSC(x) value is negative, we remove x from consideration, i.e., x is regarded as an existing class instance. For the remaining $F\text{-}outlier$s, q-NSC(.) is within the range [0,1]. Now, we compute a compound measure for each such $F\text{-}outlier$, called *Novelty score* or *Nscore*, as follows: $Nscore(x) = \frac{1-weight(x)}{1-minweight} q\text{-}NSC(x)$, where weight($x$) is defined in Section III-A, and minweight is the minimum weight among all $F\text{-}outlier$s having positive q-NSC. $Nscore$ contains two parts: The first part measures how far the outlier is away from its nearest existing class pseudopoint (higher value - greater distance) The second part measures the cohesion of the $F\text{-}outlier$ with other $F\text{-}outlier$s, and the separation of the $F\text{-}outlier$ from the existing class instances. Note that the value of $Nscore(x)$ is within [0,1]. A higher value indicates a greater likelihood of being a novel class instance. The distribution of $Nscore(x)$ can be characterized by the actual class of $F\text{-}outlier$ instances. In other words, by examining the distribution of $Nscore(x)$, we can decide about the novelty of the $F\text{-}outlier$ instances, as follows.

We discretize the $Nscore(x)$ values into n equal intervals (or bins), and construct a cumulative distribution function

(CDF) of $Nscore$. Let y_i be the value of the CDF for the ith interval. We compute the *discrete Gini Coefficient* G(s), for a random sample of y_i, as follows: $G(s) = \frac{1}{n}(n + 1 - 2(\frac{\sum_{i=1}^{n}(n+1-i)y_i}{\sum_{i=1}^{n} y_i}))$. Let us consider three different cases and examine the behavior of G(s) in each case.

Case 1: All $Nscore(x)$ are very low, and fall in the first interval. Therefore, y_i = 1 for all i. So G(s) becomes (after simplification): $G(s) = \frac{1}{n}(n + 1 - 2(\frac{\sum_{i=1}^{n}(n+1-i)1}{\sum_{i=1}^{n} 1})) = 0$. Note that this case occurs when all F-$outlier$s actually belong to the existing classes.

Case 2: All $Nscore(x)$ are very high, and fall in the last interval. Therefore, y_n = 1 and y_i=0 for all $i < n$. So G(s) becomes (after simplification): $G(s) = \frac{1}{n}(n + 1 - 2(\frac{1}{1})) = \frac{n-1}{n}$. Note that this case occurs when all F-$outlier$s actually belong to the novel class.

Case 3: $Nscore(x)$ is evenly distributed across all the intervals. In this case $y_i = i/n$ for all i. So G(s) becomes (after simplification): $G(s) = \frac{1}{n}(n+1-2(\frac{\sum_{i=1}^{n}(n+1-i)i}{\sum_{j=1}^{n} i})) = \frac{n-1}{3n}$. Note that this case occurs if the distribution is mixed, i.e., noise, concept-drift and possibly some novel class instances.

By examining the three cases, we can come up with a threshold for Gini Coefficient to identify a novel class. If $G(s) > \frac{n-1}{3n}$, we declare a novel class and tag the F-$outlier$s as novel class instances. If $G(s) = 0$, we classify the F-$outlier$s as existing class instances. If $G(s) \in (0, \frac{n-1}{3n})$, we filter out the F-$outlier$s falling in the first interval, and consider rest of the F-$outlier$s as novel class. Note that if $n \to \infty$, $\frac{n-1}{3n} \to \frac{1}{3}$. But for any value of $n < \infty$, $\frac{n-1}{3n} < 1/3$. For example, if n=10, then $\frac{n-1}{3n} = 0.3$. We use n=10 in our experiments.

C. Simultaneous multiple novel class detection

It is possible that more than one novel class may arrive at the same time (in the same chunk). This is a common scenario in text streams, such as Twitter messages. Note that determining whether there are more than one novel classes is a challenging problem, since we must execute it in an unsupervised fashion. In order to detect multiple novel classes, we construct a graph, and identify the connected components in the graph. The number of connected components determines the number of novel classes. The basic assumption in determining the multiple novel classes follows from the *cohesion* and *separation* property. For example, if there are two novel classes, then the separation among the different novel class instances should be higher than the cohesion among the same-class instances.

At first we use N_list, the collection of novel class instances detected using the novel class detection technique, to create K_v pseudopoints using K-Means clustering and summarizing the clusters. Here $K_v = K|N\_List|/S$ (S being the chunk size). Then we build a graph $G = (V, E)$. Each psudopoint is considered a vertex of G. For each pseudopoint h, we find its nearest neighbor $h.nn$ based on centroid distances, and compute the silhouette coefficient of h using the following formula: $h.sc = \frac{dist(h,h.nn)-h.\mu}{max(dist(h,h.nn),h.\mu)}$, where $dist(h, h.nn)$ is the centroid distance between h to $h.nn$, and $h.\mu$ is the mean distance from the centroid of h to all instances belonging to h. If $h.sc$ is high (close to 1), it indicates h is a tight cluster and it is far from its nearest cluster. On the other hand, if $h.sc$ is low, then h is not so tight, and close to its nearest cluster. We add an edge (h,$h.nn$) to G if $h.sc$ is less than a threshold th_{sc}, which indicates h and $h.nn$ are not so separable. We use th_{sc}=0.8 in all experiments. Once we have the graph G, we find the connected components, and mark each pseudopoint with the corresponding component number.

For each connected component $g_i \in G$, we first compute its global centroid $C(g_i)$, i.e., the center of gravity of all pseudopoints in g_i, and $\mu d(g_i)$, i.e., the mean distance of all the pseudopoints in g_i from $C(g_i)$. For each pair of components $(g_1, g_2) \in G$, we merge them if $\mu d(g_1)+\mu d(g_2)$ is greater than twice the distance between $C(g_1)$ and $C(g_2)$. In other words, two components are merged if the mean intra-component distance is higher than the inter-component distance, i.e., the components are less dense and less separable from each other. Finally, we assign class labels to each novel class instance, which is equal to the component number to which the instance belongs.

IV. Experiments

A. Dataset

We have done extensive experiments on the Twitter, Forest Cover, KDD, and synthetic data sets. Due to space limitation, we report only for Twitter and Forest Cover datasets. The descriptions of these datasets may be found in [8].

B. Experimental setup

Baseline techniques: ***MineClass:*** This is the existing approach proposed in [6]. ***MCM:*** This is the proposed approach, which stands for Multi Class Miner in Data Streams. ***OW:*** This is the combination of two approaches, namely, OLINDDA [7], and weighted classifier ensemble (WCE) [9]. OLINDDA works as a novel class detector, and WCE performs the classification. Similar baseline has been used in [6], with two variations - parallel and single. Here we use only the parallel model since it was the better of the two. In all experiments, the ensemble size and chunk-size are kept the same for both these techniques. Besides, the same base learner (i.e., k-NN) is used for all three methods.

Parameters settings: Feature set size = 30 for Twitter dataset. For other datasets, all the numeric features are used. K (number of pseudopoints per chunk) = 50, S (chunk size) = 1000, L (ensemble size) = 6, q (minimum number of F-$outlier$s required to declare a novel class) = 50. For OLINDDA, we use the default parameter values [7].

C. Evaluation

1) Overall novel class detection: ***Evaluation approach:*** We use the following performance metrics for evaluation: M_{new} = % of novel class instances Misclassified as existing class, F_{new} = % of existing class instances Falsely identified as novel class, **ERR** = Total misclassification error (%)(including M_{new} and F_{new}). We build the initial models in each method with the first $InitNumber$ chunks. From the $InitNumber + 1$st chunk onward, we first evaluate the performances of each method on that chunk, then use that chunk to update the existing models. We use $InitNumber$=3 for all experiments. The performance metrics for each chunk are saved and aggregated for producing the summary result.

Figures 2(a), and 2(b) show the ERR rates for each approach throughout the stream in the Twitter, and Forest datasets respectively. For example, in figure 2(a) at X axis = 200, the Y values show the average ERR of each approach from the beginning of the stream to chunk 200 in Twitter dataset. At this point, the ERR of MineClass, MCM, and OW are 17.2%, 1.3%, and 3.3%, respectively. Figures 2(d), and 2(e) show the total number of novel instances missed for each of the baseline approaches for Twitter and Forest dataset, respectively. For example, in figure 2(e), at the same value of the X axis (=200), the Y values show the total novel instances missed (i.e., misclassified as existing class) for each approach from the beginning of the stream to chunk 200 in the Twitter dataset. At this point, the number of novel instances missed by MineClass, MCM, and OW are 929, 0, and 3533, respectively. The ROC curves for the Twitter, and Forest datasets are generated by plotting false novel class detection rate (false positive rate if we consider novel class as positive class and existing classes as negative class) against the true novel class detection rate (true positive rate). Figure 2(c) shows the ROC curves for the Twitter dataset.

Table I
SUMMARY OF THE RESULTS

Dataset	Method	ERR	M_{new}	F_{new}	AUC
Twitter	MineClass	17.0	24.3	15.1	0.88
	MCM	**1.8**	**0.7**	**0.6**	**0.94**
	OW	3.1	100	1.4	0.56
Forest	MineClass	3.6	8.4	1.3	0.97
	MCM	**3.1**	**4.0**	**0.68**	**0.99**
	OW	5.9	20.6	1.1	0.74

Table I summarizes the results of overall classification and novel class detection error i.e., error in classification and detecting novel class only (not distinguishing multiple novel classes). For example, the column headed by M_{new} reports the M_{new} rates of each approach in different datasets for the entire stream. In Twitter dataset, the M_{new} rates are 24.3%, 0.7%, and 100% for MineClass, MCM, and OW, respectively. The column AUC reports the area under the ROC curves for each dataset. To summarize the results, MCM outperforms MineClass and OW in ERR, F_{new} and F_{new} rates. This is because of the enhanced mechanism of MCM in detecting novel classes. Recall that MCM applies an adaptive threshold for outlier detection, and also employs a probabilistic approach in recognizing the novel class instances. The net effect is that the overall F_{new} and M_{new} rates drop significantly and the ERR rate also drops.

Table II
SUMMARY OF MULTI-NOVEL CLASS DETECTION RESULTS

Dataset	Occurrence	1	2	3	4	Total
Twitter	Type 1 as Type 1	360	394	508	447	1709
	Type 1 as Type 2	0	0	0	0	0
	Type 2 as Type 2	518	568	453	500	2039
	Type 2 as Type 1	35	0	55	19	109
	Precision	1.0	1.0	1.0	1.0	1.0
	Recall	0.91	1.0	0.9	0.96	0.94
	F-measure	0.95	1.0	0.94	0.98	0.97
Forest	Type 1 as Type 1	371	583	—	—	954
	Type 1 as Type 2	183	444	—	—	627
	Type 2 as Type 2	300	550	—	—	850
	Type 2 as Type 1	113	411	—	—	524
	Precision	0.67	0.57	—	—	0.60
	Recall	0.77	0.59	–	—	0.64
	F-measure	0.71	0.58	–	–	0.62

2) Multi novel class detection: Table II reports the multiple novel class detection results. There are 4 and 2 occurrences of two novel classes in Twitter, and Forest datasets, respectively. In other words, two novel classes appear simultaneously in 4 different data chunks in Twitter dataset, and two novel classes appear simultaneously in 2 different data chunks in Forest dataset. For each occurrence of multiple novel classes, we report the confusion matrix in a single column. The entries in the rows headed by 'Type 1 as Type 1' report the number of type 1 novel class instances correctly detected as type 1, the rows headed by 'Type 1 as Type 2' report the number of type 1 novel class instances incorrectly detected as type 2, and so on. For example, in the Twitter dataset, and in the first occurrence of two novel classes (under column '1'), all of the 360 instances of type 1 novel class are identified correctly as type 1; none of the type 1 novel class instances are incorrectly identified as type 2; 518 of the type 2 novel class instances are correctly identified as type 2; and 35 of the type 2 novel class instances are incorrectly identified as type 1. Note that the numbering of type 1 and 2 are relative. We also summarize our findings by reporting the precision, recall, and F-measure for each occurrence for each dataset, based on the mis-classification of type 1 novel class instance into the other kind. For example, the table cell corresponding to the column headed by '1' and the row headed by 'Twitter F-measure' reports the F-measure of multiple novel class detection on the first occurrence of two novel classes in Twitter dataset, which is 0.95. The F-measure is computed by considering type 1 instances as positive, and the other as negative class. Considering the fact that we apply an an unsupervised approach, the results are very promising, especially in the Twitter dataset, where the F-measure is 0.97. For the Forest dataset, the F-measure

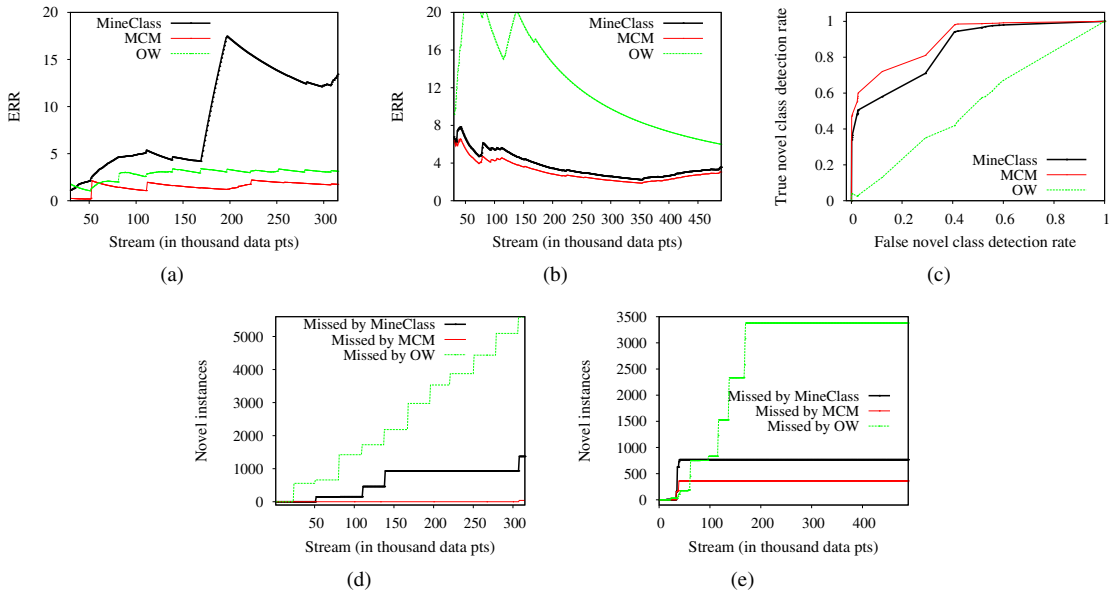

Figure 2. ERR rates in (a) Twitter, and (b) Forest dataset; (c) ROC curves in Twitter dataset; Novel classes missed in (d) Twitter , and (e) Forest dataset

is lower because the novel classes in Twitter dataset are relatively well separated than that of the Forest dataset.

V. Conclusions

We propose several improvements over the existing classification and novel class detection technique. First, we propose an improved technique for outlier detection by defining a dynamic *slack space* outside the *decision boundary* of each classification model. Second, we propose a better alternative for identifying novel class instances using discrete Gini Coefficient. Finally, we propose a graph-based approach for distinguishing among multiple novel classes. We apply our technique on several real data streams that experience concept-drift and concept-evolution, and achieve significant performance improvements over the existing techniques. In the future, we would like to extend our technique to text and multi-label stream classification problems.

Acknowledgement

Research was sponsored in part by AFOSR MURI award FA9550-0810265, NASA grant NNX08AC35A, and the Army Research Laboratory (ARL) under Cooperative Agreement No.W911NF-0920053 (NS-CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ARL or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] 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.

[2] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavald. New ensemble methods for evolving data streams. In *SIGKDD*, pages 139–148, 2009.

[3] S. Hashemi, Y. Yang, Z. Mirzamomen, and M. Kangavari. Adapted one-versus-all decision trees for data stream classification. *IEEE TKDE*, 21(5):624–637, 2009.

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

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

[6] 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.

[7] 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.

[8] University of Texas at Dallas Data Mining Tools Repository. http://dml.utdallas.edu/Mehedy.

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

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

[11] P. Zhang, X. Zhu, and L. Guo. Mining data streams with labeled and unlabeled training examples. In *ICDM*, pages 627–636, 2009.