Recurring and Novel Class Detection in Concept-Drifting Data Streams Using Class-Based Ensemble

Mohammad Raihanul Islam

Bangladesh University of Engineering and Technology

Abstract. Over the recent years concept-evolution has received a lot of attention because of its importance in the context of mining data streams. Mining data stream has become an important task due to its wide range of applications such as network intrusion detection, credit card fraud protection, identifying trends in the social networks etc. Concept-evolution means introduction of novel class in the data stream. Many recent works address this phenomenon. In addition, a class may appear in the stream, disappears for a while and then reemerges. This scenario is known as recurring classes and remained unaddressed in most of the cases. As a result, generally where a novel class detection system is present, any recurring class is falsely detected as novel class. This results in unnecessary waste of human and computational resources. In this paper, we have proposed a class-based ensemble of classification model addressing the issues of recurring and novel class in the presence of concept drift and noise. Our approach has shown impressive performance compared to the state-of-art methods in the literature.

Keywords: Novel Class, Recurring Class, Concept Evolution, Stream Classification.

1 Introduction

The problem of data stream classification has been studied among the research community over the recent years. One of the major characteristics of data stream mining is that, the classification is a continuous process thus the size of the training data can be considered infinite. So it is almost impossible to store all the examples to train the classifiers. Some methods regarding incremental learning are proposed in [4,9] to address this problem. Moreover, it is a common scenario that, the underlying concept may changes overtime; a characteristic known as *concept-drift*.

However, another significant phenomenon of the data stream is *concept-evolution*, which is considered as the emergence of novel classes in the stream. For example, a new topic may appear in social network or a new type of intrusion may be identified in the network. If the number of classes in the classifiers is fixed and no novel class detection system is present, then the novel class is falsely identified as existing class. Concept Evolution has become a new research

V.S. Tseng et al. (Eds.): PAKDD 2014, Part II, LNAI 8444, pp. 425–436, 2014.

direction for the researchers recently because of its practical importance. For example, if a new type of attack occurs in the network, it is imperative to identify it and take actions as soon as possible. Several approaches regarding this issue have been studied in the literature [5, 7].

A special case of concept-evolution is *recurring class* where a class reemerges after its long disappearance from the stream. For example, a popular topic may appear in a social network at a particular time of the year (i.e. festivals or elections). This result in a change of topics in the discussion on the social network over the time period and then when the event ends the topic disappears again. A recurring class creates several discrepancies if not properly handled. If it is not properly identified, then it is erroneously considered as a novel class or an existing one. As a result, a significant amount of human resources is wasted to detect its reappearance. Some studies regarding the problem of recurring class are present in [1, 6].

The classification model for data stream can be constructed by ensemble of classifiers. In an ensemble approach, multiple base classifiers learn the decision boundary on the learning patterns and their decisions on test example ares fused to reach the final verdict. The ensemble approach is more popular among the research community because of their higher accuracy, efficiency and flexibility [5].

The contributions of this paper are as follows. In this paper, we propose a new technique to generate ensemble of classifiers to detect novel and recurring class in the data stream which reduces overall classification errors. Moreover, we have observed the phenomenon that, if the class boundary between two classes is very close, then it is possible to get a false prediction if the instances falls closely to boundary region. In our approach, we have employed several strategies to mitigate this problem. Finally, we have also used the falsely predicted instances to update our model. Our proposed method has outperformed the state-of-the-art techniques in the literature.

The rest of the paper is organized as follows. In Section 2, we discuss the previous works regarding data stream classification in the literature. We present our approach in Section 3. We discuss the experimental results in Section 4. We briefly conclude in Section 5.

2 Previous Works

Several studies are present in the literature on data stream classification [1–4, 6, 8–10]. Due to page limitation we have highlighted studies only related to novel and recurring class detection. It has been observed that, existing approaches can be divided into two categories. First one is single model approach where one classification model is used and periodically updated for new data. On the other hand, batch-incremental method constructs each model using batch learning. When older model can no longer give satisfactory results, it is replaced by newer models [9]. The advantage of ensemble model is that, updating the classification model is much simpler in this case. However, these techniques generally do not include novel or recurring class detection.

An approach to identify recurring class is presented in [6]. Here in addition to primary ensemble model, an auxiliary ensemble of classifiers is present. The auxiliary ensemble model is responsible for storing all the classes even after they disappear from the data stream. When an instance is detected as outlier in the primary ensemble, but falls within the decision boundary of auxiliary ensemble, the instances is identified as recurrent class. Any test data outside the decision boundary of both ensembles are analyzed for novel class.

The approaches described in [6] are considered as *chunk-based* method. A *class-based* ensemble approach is presented in [1]. Here an ensemble model is constructed for each class C of the data stream. Each ensemble has K micro-classifiers. Initially, micro-classifiers are trained from the data chunk. When a latest labeled chunk of data arrives, a separate micro-classifier is trained for each class. Then the newly trained micro-classifier replaces the one with highest prediction error of the respective class. An instance falls outside the decision boundary of all the micro-classifiers of all the classes is considered as an outlier and saved in a buffer. The buffer is checked periodically to detect novel class. Authors of [1] have shown theoretically and experimentally that, class-based approach is better than the chunk-based technique.

In this paper, we propose a more sophisticated approach to construct a class-based ensemble of classifiers. We have also present a better way to update and maintain the ensemble model. Moreover, we propose two types of outliers to update the classifiers and novel class detection and also take the wrongly predicted data into account to modify the classifiers. Experiments show the effectiveness of our methods compared to other techniques.

3 Our Approach

First, we discuss the fundamental concept of data stream classification. Then we describe our approach for stream classification subsequently.

3.1 Preliminaries

Each data in the stream arrives in the following format:

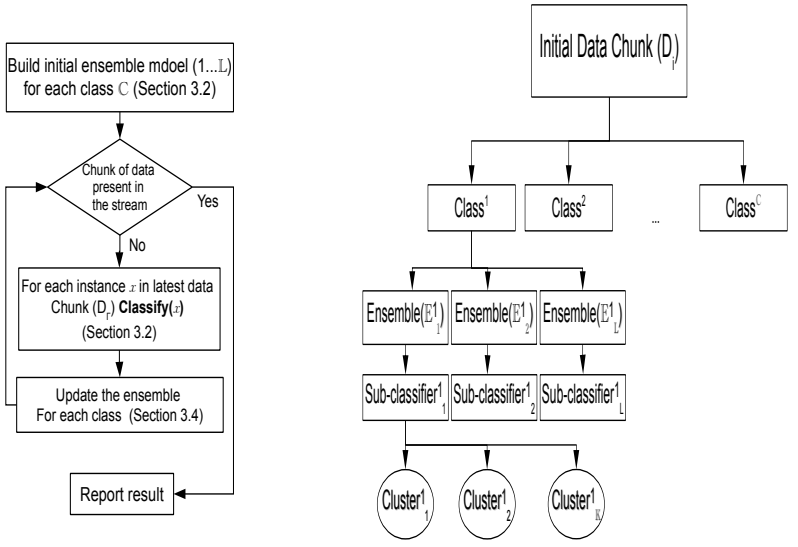
$$\begin{gathered} D_1 =< x_1, ...x_S >, \\ D_2 =< x_{S+1}, ...x_{2S} >, \\ ..... \\ D_\Gamma =< x_{(\Gamma-1)S+1}, ...x_{\Gamma S} > \end{gathered}$$

where x_i is the i^{th} instance in the stream and S is the size of the stream. D_i is the i^{th} data chunk and D_Γ is the latest data chunk. The problem is to predict the class of each data point. Let l_i and $\hat{l_i}$ be the actual and predicted label of instance x_i. If $l_i = \hat{l_i}$ then the prediction is correct otherwise it is incorrect. The goal is to minimize the prediction error.

Stream classification can be used in various applications such as labeling message in social network or identify intrusion in the network traffic. For example, in credit card fraud detection system, each transaction can be considered as an

instance or data point and can be predicted either as *authentic* or *fraud* by any classification technique. If the transaction is predicted as *fraud*, then immediate action can be taken to withhold the transaction. Sometimes, the predicted decision can be wrong (authentic transaction predicted as fraud or vice versa). This can be verified from the cardholder later. The feedback can be considered as "labeling" the instance and used to refine the classification model.

The major task in the data stream classification is to keep the classification model up-to-date by modifying it periodically with the most recent concept. The overview of our proposed approach is shown in Figure 1(a). The major parts of the algorithm will be described step-by-step.



(a) *Overall Approach*

(b) *Partial Structure of the Ensemble Model*

Fig. 1. Overall approach and structure of the classification model

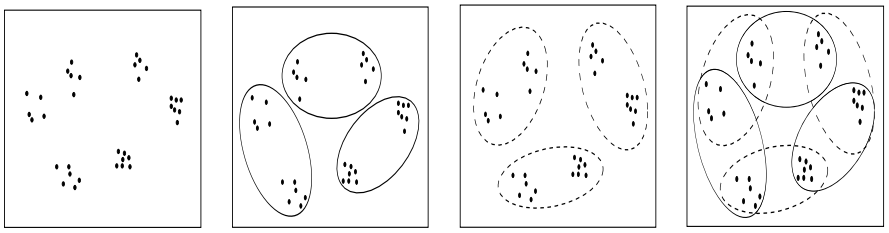
3.2 Ensemble Construction and Training

Now, we present the approach for generating the ensemble model. We will refer our model as **R**ecurring and **N**ovel **C**lass **D**etector **E**nsemble (RNCDE).

Initially, the data chunk is partitioned into $\mathbb{C}$ disjoint groups ($\mathcal{G}^1$,$\mathcal{G}^2$,...,$\mathcal{G}^{\mathbb{C}}$) according to the true class labels, where $\mathbb{C}$ is number of classes in the chunk. Therefore, each group contains the instances of one class only. Then an ensemble of size $\mathbb{L}$ is constructed for each class i using $\mathcal{G}^i$. Each ensemble $\mathbb{E}^i_l$, $i \in \mathbb{C}$, $l \in \mathbb{L}$ is composed of a sub-classifier $\mathbb{S}^i_l$. Each sub-classifier $\mathbb{S}^i_l$ is trained on the instance of class i ($\mathcal{G}^i$). We apply K-means clustering to generate $\mathbb{K}$ clusters on the instances of each class i. For each cluster $\mathbb{H}^i_{l_j}$ of ensemble l of class i, where $j \in \mathbb{K}$ we keep

a summary of the cluster i.e. μ, the centroid, r, the cluster radius (distance between centroid and the farthest data point of the cluster) and η, the number of points belonging to the cluster. This way we do not need every data point of the cluster. Therefore, each sub-classifier $\mathbb{S}_l^i$ is composed of all the clusters built from the instances of class i ($\mathbb{S}_l^i = \bigcup_{j=1}^{\mathbb{K}} \mathbb{H}_{l_j}^i$). This process for generating sub-classifiers $\mathbb{S}_l^i$ is repeated $\mathbb{L}$ times to construct the ensemble model $\mathbb{E}^i$ for class i ($\mathbb{E}^i = \bigcup_{l=1}^{\mathbb{L}} \mathbb{S}_l^i$). Finally, the overall model is the union of all the ensemble built for each class i ($\mathbb{E} = \bigcup_{i=1}^{\mathbb{C}} \mathbb{E}^i$). For visual purpose, the partial structure of the ensemble model is shown in hierarchical form in Figure 1(b). It should be noted that, each ensemble for class i has only one sub-classifier.

Note that, each sub-classifier $\mathbb{S}_l^i$ of an ensemble $\mathbb{E}^i$ is trained on the same data $\mathcal{G}^i$. We vary the seed parameters $(\gamma_1, \gamma_2, ...\gamma_{\mathbb{L}})$ of K-means clustering to diversify the sub-classifier. We have shown our method using a hypothetical example in Figure 2. In Figure 2(a), the instances of the same class are shown. The K-means clustering is applied to construct sub-classifier 1 using seed parameter γ_1 (Figure 2(b)), where $\mathbb{K} = 3$. Then again sub-classifier 2 is constructed by K-means clustering initialized by the seed parameter γ_2 shown in Figure 2(c). We can see that, identical instances belong to different clusters at each sub-classifier. This process is repeated $\mathbb{L}$ times to construct $\mathbb{L}$ alternating sub-classifiers $\mathbb{S}_1^i....\mathbb{S}_{\mathbb{L}}^i$ for class i which is shown in Figure 2(d).



(a) Instance in feature space (b) Sub-classifier 1 (c) Sub-classifier 2 (d) Sub-classifier 1 and 2 superimposed

Fig. 2. A hypothetical example of layer for 2-dimensional search space

The advantages of K-means clustering is that, its lower time complexity will allow to built classifiers in reduced time which is a critical requirements for data stream mining. Another benefit is that, after construction of the clusters, it is easy to modify them compared to other types of classifiers.

3.3 Classification

Here we describe our classification procedure and outlier detection. Each data point in the most recently arrived chunk is first checked for whether it is an outlier. We have maintained two types of outlier i.e. class-outlier ($\mathbb{C}$-outlier) and

universal-outlier ($\mathbb{U}$-outlier). If any instance is outside the decision-boundary of all the sub-classifiers of all the ensembles($\bigcup_{i=1}^{\mathbb{C}} \mathbb{E}^i$), then it considered as a $\mathbb{U}$-outlier. If a data point is a $\mathbb{U}$-outlier, then it is saved in *buffer* to analyze it further. If an instance x_i is not a $\mathbb{U}$-outlier then, it is inside the decision boundary of any class. It is possible that, x_i may be inside of more than one class due to noise and the curse of dimensionality. Let $\mathcal{E}_{x_i}$ be the set of such classes. We decide which class x_i belongs to by computing a coefficient (m-value). We called this coefficient *membership coefficient*. The m-value ($\tau^i_{l_j}$) for cluster $\mathbb{H}^i_{l_j}$, where $i \in \mathcal{E}_{x_i}$, $l \in \mathbb{L}$ and $j \in \mathbb{K}$ can be computed using the equation below,

$$\tau^i_{l_j} = \left(\frac{\eta^i_{l_j}}{\max\limits_{m \in \mathcal{E}_{x_i}, n \in \mathbb{L}, o \in \mathbb{K}} \eta^m_{n_o}} \right) / \left(\frac{d^i_{l_j}}{\max\limits_{m \in \mathcal{E}_{x_i}, n \in \mathbb{L}, o \in \mathbb{K}} d^m_{n_o}} \right)^{\beta}, \tag{1}$$

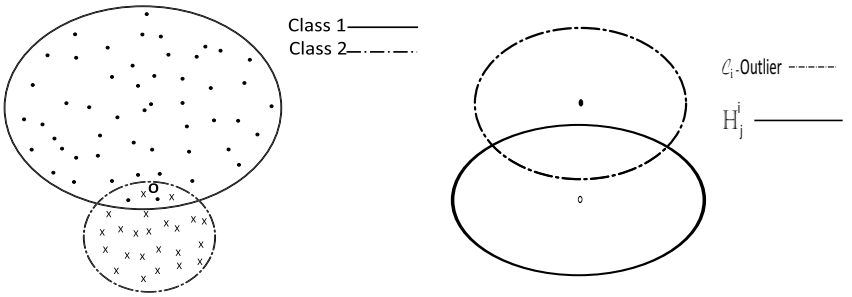
where $d^i_{l_j}$ is the Euclidean distance between the instance x_i and the centroid of cluster $\mathbb{H}^i_{l_j}$ where $\eta^i_{l_j}$ is the size of the cluster. Here β is the relative importance of the inverse of distance over the size of the classifier. We refer this constant as ξ-coefficient. The max size and max distance is used for normalization. After computing m-value for each cluster of all the sub-classifiers, the class label for instance x_i is computed using the equation below,

$$c = \arg \max_{i \in \mathcal{E}_{x_i}, l \in \mathbb{L}, j \in \mathbb{K}} \tau^i_{l_j} \tag{2}$$

The reason behind introducing the cluster size in the classification process is depicted in Figure 3(a). Here a hypothetical scenario is shown where two different clusters of different classes are present. Boundary of one of the clusters (cluster 1) is shown in continuous line (Class 1) and the other (cluster 2) is in dashed line (Class 2). We have also shown the data points of the clusters (i.e. dots and crosses). Now consider an instance shown by "O" in the figure. It is inside the boundary of both class. If we only consider only the Euclidean distance then it belongs to Class 2. However, from the figure it is evident that, it is more prone to the centroid of cluster 1 than cluster 2. Since size of cluster for Class 1 is larger, the decision boundary of cluster 1 is more expanded. Considering only the nearest neighbor to label the instance may result in erroneous prediction. However, if we make the assumption that, all the data points of a cluster are uniformly distributed, then the number of points in the overlapped region (common region between two clusters) will be greater for cluster 1 than cluster 2. In this case, the test instance will be labeled as Class 1. Therefore, a more sophisticated measurement can be possible if we take account the size of the cluster in the classification process.

3.4 Ensemble Update

When the labels for data points of a chunk are available (labeled by human expert), the incorrectly predicted data ($\mathcal{W}$) by the ensemble model is identified. Then the wrongly predicted data are separated according to their correct



(a) *Hypothetical decision boundary of 2 clusters of two different classes*

(b) *A scenario for cluster merging*

Fig. 3. $\mathbb{K}$ vs ERR

label. As a result, the all the inaccurately predicted data are partitioned into disjoint sets ($\mathcal{W}^1, \mathcal{W}^2, ...\mathcal{W}^{\mathbb{C}}$). Then the data in $\mathcal{W}^i$ are clustered using K-means clustering. The number of clusters $\mathcal{K}$ is computed using the following equation:

$$\mathcal{K} = \frac{|D_\Gamma|}{ChunkSize} \cdot \mathbb{K} \tag{3}$$

Here $ChunkSize$ is a constant which can be initialized manually. These newly formed clusters can be called $\mathcal{C}_i$-outlier clusters where $i \in \mathbb{C}$. The union of $\mathcal{C}_i$-outlier is the $\mathbb{C}$-outlier. After the formation of $\mathcal{C}_i$-outlier clusters, the Euclidean distance from each $\mathcal{C}_i$-outlier clusters to each $\mathbb{H}^i_{l_j}$ is computed. Now based on the distance among the clusters we make two types of modifications. One is cluster merge and the other is cluster replacement.

If the distance between a $\mathcal{C}_i$-outlier clusters and one of the clusters ($\mathbb{H}^i_{l_j}$) in the ensemble is less than the radius of $\mathbb{H}^i_{l_j}$ ($r^i_{l_j}$), then the two clusters are merged. Recall that, the data points of $\mathcal{C}_i$-outlier are actually the wrongly predicted instances clustered according to the actual class label i. So it is normal that, any cluster from $\mathcal{C}_i$-outlier will tend to very remain very close to the $\mathbb{H}^i_{l_j}$ in the ensemble model. A possible scenario depicting the condition for merging the clusters is shown in Figure 3(b). Here the distance between $\mathbb{C}_i$-oulier cluster and the centroid of $\mathbb{H}^i_{l_j}$ is less than the radius of $\mathbb{H}^i_{l_j}$ ($r^i_{l_j}$).

Now to merge the cluster, we have to calculate the new centroid, the cluster size and the radius. To calculate the position of new centroid we have used the the equation below:

$$\mu^i_{l_j} = \frac{\eta^i_{l_j} \cdot \mu^i_{l_j} + \eta_{\mathcal{C}_i-\text{outlier}} \cdot \mu_{\mathcal{C}_i-\text{outlier}}}{\eta^i_{l_j} + \mu_{\mathcal{C}_i-\text{outlier}}}, \tag{4}$$

where $\eta_{\mathcal{C}-\text{outlier}}$ and $\mu_{\mathcal{C}-\text{outlier}}$ are the size and centroid of the $\mathcal{C}_i$-outlier. Since two clusters are merged, size is addition of the size of two clusters. The radius is computed by combining the radii of two clusters with the distance between the centroids.

After the merging of clusters the remaining $\mathcal{C}_i$-outlier clusters are replaced with the clusters from the sub-classifier. The replacement policy is as follows. We keep a count of error $\varepsilon^i_{l_j}$ for each cluster $\mathbb{H}^i_{l_j}$ for each ensemble model. Recall that, classification is computed by the m-value of the cluster. If prediction is wrong then count of error is increased by 1 for the cluster with max $\tau^i_{l_j}$, because it falsely identified the class as i. Now we replace the remaining un-merged clusters with clusters with highest $\varepsilon^i_{l_j}$ values accordingly. This way, the sub-classier can get rid of the obsolete clusters and the issue of concept-drift is resolved. Since we replace the older clusters with the cluster constructed with the most recent data points, the ensemble model remains up-to-date with the latest concept.

3.5 Novel Class Detection

We have extended and generalized the idea of novel class detection in [1]. The primary assumption behind the novel class detection in [1] was, data points of the same class should be closer to each other (*cohesion*) and farther apart from the other classes (*separation*). However, first assumption (i.e. cohesion) may prove different in some complex cases. It may be possible that, data points of the same class may be clustered together in various groups where these groups may be scattered through the feature space.

If the data points of a novel class emerge in the stream, we can assume that, the instances belonging to novel class will be far from the decision boundary of existing classes. Since data points of $\mathbb{U}$-outlier are outside the decision boundary of all the existing classes, these data are analyzed for novel classes. Recall that, the $\mathbb{U}$-outliers are stored in a buffer, if the size of the buffer reaches a threshold then they are analyzed for novel class. We have modified the metric called q-NSC authors of [1] used and called it ***q*-mNSC**. In this method, another metric called *q,c-neighborhood* is used. We modify the definition of q,c-neighborhood also, which we called q,h-neighborhood. We define it as follows:

q,h-neighborhood: The q,h-neighborhood (q,h(x) in short) of an $\mathbb{U}$-outlier x is the set of q clusters that are nearest to x. (q-nearest cluster h neighbor of instance x).

Here q is a user defined parameter which can be initialized at the beginning. In summary, we compute the nearest q number of clusters from instance x regardless of the class the clusters belong to.

Now suppose, $\bar{D}_{h_{out},q}(x)$ be the mean distance of a $\mathbb{U}$-outlier instance x to its q nearest $\mathbb{U}$-outlier neighbors. Moreover, let $\bar{D}_{h,q}(x)$ be the mean distance from x to its $q, h(x)$ and $\bar{D}_{h_{min},q}(x)$ be the minimum value among all $\bar{D}_{h,q}(x)$. Here, h is the set of clusters from the existing classes. Then the q-mNSC of x can be computed according our definition:

$$q\text{−mNSC(x)} = \frac{\bar{D}_{h_{min},q}(x) - \bar{D}_{h_{out},q}(x)}{\max(\bar{D}_{h_{min},q}(x), \bar{D}_{h_{out},q}(x))} \tag{5}$$

The value of q-mNSC(x) ranges between -1 to +1. When the value is positive x is closer to $\mathbb{U}$-outlier instances and away from the existing classes resulting more cohesion and vice versa.

Now we explain how we can utilize the metric to detect novel class. First, we apply K-means clustering on $\mathbb{U}$-outliers to partition the data to $\mathcal{K}_0$ number of clusters, where $\mathcal{K}_0 = \mathbb{K} \cdot \frac{|\text{buffer}|}{ChunkSize}$. The reason for applying clustering is to reduce time complexity(reduces from $O(n^2)$ to $O(\mathcal{K}_0^2)$, where n is the total number of data points in $\mathbb{U}$-outlier). For each $\mathbb{U}$-cluster we compute q nearest cluster h_n for all the sub-classifiers of all the class. After that, for each $\mathbb{U}$-cluster we compute q-nearest neighbor cluster of that $\mathbb{U}$-cluster. Then we apply the Equation 5 to compute the q-mNSC for each $\mathbb{U}$-cluster. This way we get a q-mNSC value for each $\mathbb{U}$-cluster in the ensemble. If the positive value of q-mNSC is greater than a fixed number (q_α) than we can conclude a novel class has emerged at the stream.

4 Experimental Findings

First we discuss about the data set and then the parameter settings. Later, we describe the results and our remarks.

We apply the procedure described in [5] to generate synthetic datasets with concept evolution and drift. We generate three types of datasets as described in [5]. Each dataset contain 2.5×10^5 instances with 40 real value attribute. We refer each set as SynNCX having X classes (i.e. SynNC10 where total 10 classes are present).

We have also taken the real-life dataset *Forest* from UCI database and the *10 percent* version of KDD CUP 1999 intrusion detection challenge. First dataset contains 581000 instances with 7 classes and 54 numeric attributes while the second datasets have 490000 instances having 23 classes and 34 numeric attributes. We randomly permutate the instances and construct 10 sequences and report the average results. We have made adjustments to have novel instances in the sequences.

We have compared our approach (RNCDE) with class-based approach (CL) [1], ECSMiner (EM) [5], the clustered-based method presented in [8] (OW) and chunk-based approach (SC) described in [6].

4.1 Parameter Settings

We have set the size of the ensemble $\mathbb{L} = 3$, number of clusters per sub-classifier $\mathbb{K} = 20$. The minimum number of instances to detect novel class $q_\alpha = 20$. Moreover, ξ is varied between 3 to 8 and size of the *buffer* is set to the 20% of the size of the chunk. These parameters are set either according to the parameters of the previous works or by running preliminary experiments.

4.2 Evaluation

We have used the following evaluation criteria for performance measurements. M_{new} = % of novel class instances misclassified as existing class, F_{new} = % of

existing class instances misclassified as novel class, OTH = % of existing class instances misclassified as another existing class and ERR = average misclassfication error (average of three types of error).

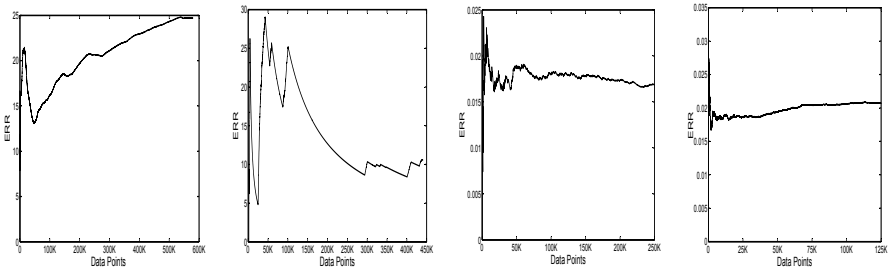
Initially, we construct the ensemble model from first three data chunks. Then we begin our performance evaluation from the chunk four. Table 1 summarizes the results from all the methods. We have taken the summary results on other methods from [1] and compared with our approach. OTH can be calculated from the other errors, so we do not show it. From the table, we can see that, OW has the highest error rate, because it can not detect majority of the novel class instances. Therefore, the F_{new} rate is also high in case of OW.

EM can identify novel class but it can not detect recurring class. As a result, recurring classes are detected as novel class and it has a high F_{new} rate also. SC maintains an auxiliary ensemble model which contains classifiers for all the class including recurring class. Therefore, it has comparatively lower F_{new} rate than EM. CL uses class- based ensemble to detect novel and recurring class and it has a lower error rate than the approaches above. Our proposed method RNCDE also have shown comparatively lower error rate than other methods. In *Forest* dataset, the ERR is slightly higher than CL, but in other case RNCDE shows better performance than other approaches.

Table 1. Summary results on all the datasets

Performance Criteria	Methods	SynNC10	SynNC20	SynNC40	Forest	KDD
F_{new}	OW	0.9	1.0	1.4	0.0	0.0
	EM	24.0	23.0	20.9	5.8	16.4
	SC	14.6	13.6	11.8	3.1	12.6
	CL	0.01	0.05	0.13	2.3	5.0
	RNCDE	0.01	0.03	0.03	5.8	4.8
M_{new}	OW	3.3	5.0	7.1	89.5	100
	EM	0.0	0.0	0.0	34.5	63.0
	SC	0.0	0.0	0.0	30.1	61.4
	CL	0.0	0.0	0.0	17.5	59.5
	RNCDE	0.0	0.0	0.0	14.4	60.1
ERR	OW	7.5	7.7	8.0	30.3	37.6
	EM	8.2	7.9	7.2	13.7	28
	SC	5.1	4.8	4.3	11.5	26.7
	CL	0.01	0.02	0.05	7.3	26.0
	RNCDE	**0.019**	**0.02**	**0.02**	**10.57**	**24.76**

In Figure 4, ERR rates for both Synthetic and Real Data are shown. In each case X axis represents number of data points and Y axis represents the ERR. For example from the Figure 4(a) and 4(b), we can see that, ERR rates after 300000 data points are 20% for *forest*, 10% in *KDD*. For synthetic data ERR remains almost constant. In case of KDD we can see at the beginning ERR fluctuates, but the ERR decreases afterwards. This occurs because the at first the class boundary among classes are not accurately drawn so misclassification among existing classes (OTH) raises ERR. When the concept is learned comprehensively then ERR decreases. On the other hand, in forest ERR rises gradually. This is because M_{new} increases continuously when more data points arrive.

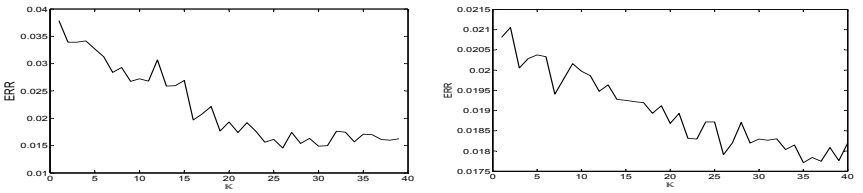


(a) *Forest* (b) *KDD* (c) *SynNC20* (d) *SynNC40*

Fig. 4. ERR for Datasets

4.3 Parameter Sensitiveness

We have observed the effect of a number of parameters on our algorithm. Due to page limitation we describe only one parameter number of clusters per sub-classifier $\mathbb{K}$. The $\mathbb{K}$ is varied between 10 to 50. The impact of varying $\mathbb{K}$ for synthetic dataset is shown in Figure 5. We can see from the figure that, ERR decreases, if the number of cluster $\mathbb{K}$ increases. The reason behind this is when the number of clusters increases more accurate decision boundary can be drawn among the classes. When the value of $\mathbb{K}$ is increased, more clusters will be formed on the same instances. Therefore, the size of the clusters will be comparatively lower and each cluster will learn the small portion of the total concept. If the boundary between two classes is noisy then more and smaller clusters will perform better than fewer and larger clusters. In other words, the boundary of the class will be more accurate constructed if an increased number clusters is formed. That is why ERR deceases if $\mathbb{K}$ is increased. However, it should be noted that, if the value of $\mathbb{K}$ is high, then it would result in high space requirements and increased time complexity, which has a detrimental effect on the performance of the model. So the value of $\mathbb{K}$ should be adjusted to balance between the performance and accuracy.



(a) *SynNC20* (b) *SynNC40*

Fig. 5. $\mathbb{K}$ vs ERR

5 Conclusion

In this paper, we have proposed a new ensemble model for detecting novel and recurring class in continuous data stream (RNCDE) which can be considered as a class-based approach as opposed to the chunk-based approach. Our algorithm have shown good performance against state-of-the-art methods in the literature. We have built our initial ensemble model for each class and updated and modified it periodically to learn the most recent concept. Each ensemble model has a sub-classifier which is composed of a number of clusters. The union of the cluster constitutes the concept of class. Our method has been proven very effective in data stream mining. Inspired by the promising results, we will concentrate on more efficient techniques for data stream classification. We are also planning to experiment our method on other real life data.

References

1. Al-Khateeb, T., Masud, M.M., Khan, L., Aggarwal, C., Han, J., Thuraisingham, B.: Stream classification with recurring and novel class detection using class-based ensemble. In: Proceedings of International Conference on Data Mining, pp. 31–40 (2012)
2. Gao, J., Fan, W., Han, J.: On appropriate assumptions to mine data streams: Analysis and practice. In: Proceedings of International Conference on Data Mining, pp. 143–152 (2007)
3. Hashemi, S., Yang, Y., Mirzamomen, Z., Kangavari, M.: Adapted one-versus-all decision trees for data stream classification. IEEE Transactions on Knowledge and Data Engineering 21(5), 624–637 (2009)
4. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 97–106 (2001)
5. Masud, M., Gao, J., Khan, L., Han, J., Thuraisingham, B.: Classification and novel class detection in concept-drifting data streams under time constraints. IEEE Transactions on Knowledge and Data Engineering 23(6), 859–874 (2011)
6. Masud, M.M., Al-Khateeb, T.M., Khan, L., Aggarwal, C., Gao, J., Han, J., Thuraisingham, B.: Detecting recurring and novel classes in concept-drifting data streams. In: Proceedings of International Conference on Data Mining (2011)
7. Masud, M.M., Chen, Q., Khan, L., Aggarwal, C., Gao, J., Han, J., Thuraisingham, B.: Addressing concept-evolution in concept-drifting data streams. In: Proceedings of the International Conference on Data Mining, pp. 929–934 (2010)
8. Spinosa, E.J., de Leon F. de Carvalho, A.P., Gama, J.: Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks. In: Proceedings of the ACM Symposium on Applied Computing, pp. 976–980 (2008)
9. Yang, Y., Wu, X., Zhu, X.: Combining proactive and reactive predictions for data streams. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 710–715 (2005)
10. Zhang, P., Zhu, X., Guo, L.: Mining data streams with labeled and unlabeled training examples. In: Ninth IEEE International Conference on Data Mining, pp. 627–636 (2009)