An Adaptive Ensemble Classifier for Mining Concept-Drifting Data Streams

Dewan Md. Farid[a], Li Zhang[a], Alamgir Hossain[a], Chowdhury Mofizur Rahman[b],
Rebecca Strachan[a], Graham Sexton[a], and Keshav Dahal[c]

[a]*Computational Intelligence Group, Department of Computer Science and Digital Technology, Northumbria University, Newcastle upon Tyne, UK*
[b]*Department of Computer Science & Engineering, United International University, Bangladesh*
[c]*Artificial Intelligence Research Group, School of Computing, Informatics and Media, University of Bradford, UK*

Abstract

Traditional data mining techniques cannot be directly applied to the real-time data streaming environment. Existing mining classifiers therefore need to be updated frequently to adopt the changes in data streams. In this paper, we address this issue and propose an adaptive ensemble approach for classification and novel class detection in concept-drifting data streams. The proposed approach uses traditional mining classifiers and updates the ensemble model automatically so that it represents the most recent concepts in data streams. For novel class detection we consider the idea that data points belonging to the same class should be closer to each other and should be far apart from the data points belonging to other classes. If a data point is well separated from the existing data clusters, it is identified as a novel class instance. We tested the performance of this proposed stream classification model against existing mining algorithms using real benchmark datasets from UCI machine learning repository. The experimental results proved that our approach shows great flexibility and robustness in novel class detection in concept-drifting and outperforms traditional classification models in challenging real-life data stream applications.

Keywords: Concept-drift, Data streams, Decision trees, Adaptive ensembles, Clustering, Novel classes, Weighting instances

1. Introduction

Data stream analysis and mining is a challenging research area in data mining and machine learning. It has recently received much attention of computational intelligence researchers [1, 2, 3]. Data stream classification is a method of extracting knowledge and information from continuous data points [4]. Data in data streams is generated with time passing by and cannot be controlled by any pre-defined order. A data stream is real-time, infinite, and dynamic, which has very diverse characteristics compared to the traditional static data or database. It can be read and processed based on the order of data arrival. Therefore, a data stream has the following distinctive features: a) dynamic, b) infinite, c) high dimensional, d) orderly, e) non-repetitive, f) high-speed, and g) time-varying [5]. In real-life streaming environments such as weather predictions, astronomical and intrusion detection etc, new instances with new class labels may appear at any time. Most existing data mining techniques cannot detect and classify such novel class instances in data streaming environments, because they are trained on instances with a fixed number of class labels [1, 6]. Thus the existing mining models will misclassify these new instances with novel class labels [7, 8]. Such data mining classifiers therefore need to be updated constantly and retrained with the labeled instances of the newly arrived novel classes in data streams, otherwise the mining models will become less and less accurate as time passes by.

Novel class detection in concept-drifting data streams means the statistical properties of the target classes change over time in unforeseen ways. It is an integral part of any realistic data stream classification technique. There are two main approaches for data stream classifications: a) Single model incremental classification, and b) Ensemble model based classification. Single model classification techniques incrementally update a single classifier with new data to cope with the evolution of the data stream. To update the single model mining classifier, usually it requires complex operations to modify the internal structure of the classifier. In some cases these techniques perform poorly in the streaming environment. On the contrary, an ensemble approach uses a combination or a set of classifiers, i.e. it combines a series of classifiers with the aim of creating an improved composite model to handle concept-drift efficiently. These ensemble models can be more effciently built than updating a single model, and also have higher classification accuracy rates than single model classification techniques.

In this paper, we propose an adaptive ensemble approach, M, for classification and novel class detection in concept-drifting data streams, which significantly extends our previous work [8, 9] on novel class detection in concept-drifting data stream environments in several ways. First, in our previous work, we applied single model incremental learning for data stream classification. Second, we did not consider any training instance weighting approach. Third, more discussions and experiments are added in this work to prove the efficiency of the proposed adaptive ensemble model. This paper addresses several challenges in data stream classifications, i.e. infinite length, limited labeled data, concept-drifting, and concept-evolution. The infinite length problem in data streams can be handled by dividing a data stream into equal-sized sub-streams [7]. The proposed

adaptive ensemble model, M, uses traditional data mining classifiers (decision trees with clustering) and updates itself automatically so that it represents the most recent concepts in data streams. A new or unlabeled instance is classified by the majority of weighted voting among the classifiers in M. The instances of each sub-stream are labeled by M and then a new classifier is trained with the most recent dataset. As soon as the new classifier becomes competitive, one of the existing classifier in M is replaced by it, if necessary. The classifier with the smallest weight reflecting the minimum classification accuracy rate in M is chosen for replacement. We address the concept-evolution problem by updating each classifier in M with a novel class detector. Therefore, a novel class can be automatically detected, if all of the classifiers in M discover the novel class.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 provides an overview of concept-drifting and Section 4 discusses our proposed approach in detail. Section 5 then describes the test datasets and experimental analysis. Finally, Section 6 concludes with directions for future work.

2. Related Work

In order to extract the most useful information and knowledge from a large amount of complex data in real world applications such as medical diagnosis, radar signal classification, and economical changes etc, in the past decade, various data mining algorithms have been proposed and developed. Masud et al. [7] proposed an ensemble approach using traditional classifiers to detect novel classes before the true labels of the novel class instances arrived. In order to determine whether an instance belonged to a novel class, their classification model sometimes needed to gather more test instances to discover similarities and differences among the data. Elwell and Polikar [10] also introduced an ensemble of classifiers for incremental learning of concept-drift named $Learn^{++}.NSE$. It trained one new clssifier for each batch of data it received, and combined these classifiers using a dynamically weighted majority voting technique based on each classifier's time-adjusted accuracy rate on current and past environments. Su et al. [11] proposed an associative classification algorithm for data streams (AC-DC) based on association rules. Their work aimed to find relationships between itemsets and extract the complete set of frequent patterns from the input dataset. The perfomance of AC-DC was tested by applying 6 datasets from the UCI machine learning repository [12].

Hou et al. [13] proposed an algorithm for constructing a decision tree from multi-relational data streams, called RedTrees. Every node in RedTrees denoted one atom, but not one attribute. An atom joined the relational variables so that one path from the root node to any node was a relational classification pattern in RedTrees. The information entropy was used to select the split manners in RedTrees. Also, Surace and Worden [14] applied a negative selection algorithm of the human immune system to more general feature sets and time-series novelty detection, where the normal condition of the system or structure may change due to time-varying environments. Tsai et al. [15] proposed a concept-drift rule mining tree, called CDR-Tree, for mining concept-drifting rules in data streams. CDR-Tree initially integrated new and old instances from different time scales into pairs and mined the rules of concept-drift through the construction of a traditional decision tree. CDR-Tree also efficiently extracted a classification model of each data block for decision making. Haggett and Chu [16, 17] proposed a system which consisted of neural network-based novelty detectors with properties taken from dynamic predictive coding (DPC) for specific applications. The proposed system was demonstrated over two use-cases. It outperformed a specialist approach in each case. In addition, Zhou et al. [18] proposed a clustering approach called SWClustering algorithm. It used Exponential Histogram of Cluster Features (EHCF) to analyse evolving data streams over sliding windows, which effectively eliminated the outdated data records. In their work, EHCF was used to handle the in-cluster evolution with temporal cluster features representing the change of the cluster distribution. Kolter and Maloof [19] also presented an ensemble method for concept-drift that dynamically created and removed weighted experts in response to changes in performance using dynamic weighted majority (DWM). It trained online learners (i.e. classifiers) of the ensemble and added or removed experts based on the global performance of the ensemble.

Furthermore, Aggarwal et al. [20] proposed an on-demand classification process to dynamically select the appropriate window of past training data to build a classifier so that the trained model can adapt quickly to the changes of the underlying data stream. Dia et al. [21] also introduced a clustering on-demand (COD) framework to dynamically cluster multiple data streams, which consisted of two phases, i.e., the online maintenance phase and the offline clustering phase. The online maintenance phase provided an efficient mechanism to maintain summary hierarchies of data streams with multiple resolutions in time linear in both the number of streams and the number of data points in each stream. Moreover, an adaptive clustering algorithm was devised for the offline phase to retrieve approximations of desired sub-streams from summary hierarchies according to clustering queries. Gaber and Yu [22] proposed a novel approach named as STREAM-DETECT to identify changes in data streams. Their approach was able to detect changes in data streams by measuring online clustering result deviation over time. Yang et al. [23] proposed a system called RePro for the processing of concept-drifting data streams by incorporating reactive and proactive predictions. RePro modified the prediction model for oncoming instances by detecting the concept change and also predicted the coming concept using the concept history.

The work presented in this research is inspired by the above related work to build an adaptive ensemble classifier and employ a weighted majority voting technique for classification. We also generate decision trees and perform data clustering for novel class detection in concept-drifting data streams.

3. Learning with Concept-Drift

In this section, we focus on the introduction of concept-drifting in data stream classifications. Existing data mining algorithms for incremental learning assumed data streams come under stationary distribution, where data concepts remain unchanged. But the concept of any instance might change any time in real world applications. Concept-drift refers to a change in the class definitions over time, or underlying class (concept) of the data changing over time [24, 25, 26]. For example, we have labeled historical (or training) data, $x_H = \{x_1, x_2, \cdots, x_t\}$, at every time step t. The task is to predict a class label of x_{t+1} as a target (or test) instance. In order to achieve this, we need to build a classification model using all or part of available labeled historical data for training. The classification model, M, predicts the class label for x_{t+1}, then x_{t+2}, and so on. Every instance, x_t, in time t is generated from a source, S_t, where S_t is a distribution over the data. A concept-drift is the dissimilarity between two sources, $S_p \neq S_q$, at two different time points p and q. The task of a data mining classifier is to classify $x \rightarrow C_i$ by determining the prior probabilities, $P(C_i)$, and the class conditional probabilities, $P(A_{ij}|C_i)$, from a given dataset. (The probability calculation is described in detail in sub-section 4.1.) The concept-drift may occur in three ways: a) class priors $P(C)$ might change over time; b) the distributions of one or several classes $P(x|C)$ might change; and c) the posterior distributions of the class memberships $P(C|x)$ might change.

Concept-drifting in data streams can be handled in three ways via: a) window-based approaches, b) weight-based approaches, and c) ensemble classifiers [15]. A window-based approach builds a classification model by selecting the instances within a fixed or dynamic stream sliding window, and adjusts window sizes based on the classification accuracy rate [27]. It combines all new and old instances together to generate a new training dataset, but only performs better for concept-drift in small datasets. In a weight-based approach, each training instance is assigned a weight. Based on the weights, some outdated training instances will be opportunistically discarded from the training dataset. The most popular evolving technique for handling concept-drift in data streams is to use an ensemble classifier (combination of classifiers), which is shown in Figure 1. The outputs of several classifiers are combined to determine a final classification, which is often called fusion rules. Also the weights are assigned to the individual classifier's outputs at each point in time. The weight is ususally a function of the historical performance in the past or estimated perfomance using 10-fold cross-validation. The best classifier among a number of classifiers (for either classification or prediction) can also be determined by performing 10-fold cross-validation. For example, in ensemble models, a mean error rate for each classifier is calculated and the best classifier with the lowest mean error rate will be selected.

Research showed the re-use of traditional data mining algorithms in nonstationary environments is a difficult and challenging task [28]. Data mining algorithms should be adaptive so that it can be continuously updated with the novel class instances as time passes. Most of the existing data mining algorithms are

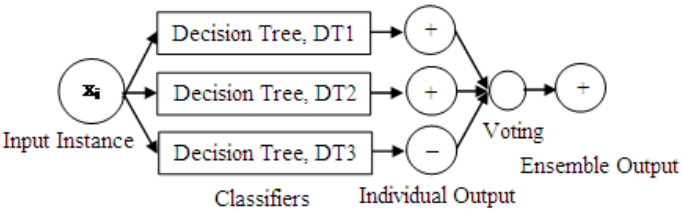

Figure 1: An example of an ensemble Classifier.

trained on datasets with a fixed number of class labels. Therefore, a new instance with a new class label will be misclassified by the traditional data mining classifiers. Figure 2 shows an example of novel class instances arriving in data streams. If we build a traditional decision tree with a fixed number of class labels (see the left-hand side of Figure 2), the decision rules are: a) if($x > x_1$ and $y < y_2$) or ($x < x_1$ and $y < y_1$) then class = plus, and b) if($x > x_1$ and $y > y_2$) or ($x < x_1$ and $y > y_1$) then class = minus. This decision tree classification model correctly classifies the instances with a fixed number of class labels, i.e. those with 'plus' and 'minus' labels, but it will misclassify any newly arrived novel class instances as shown on the right-hand side of Figure 2.

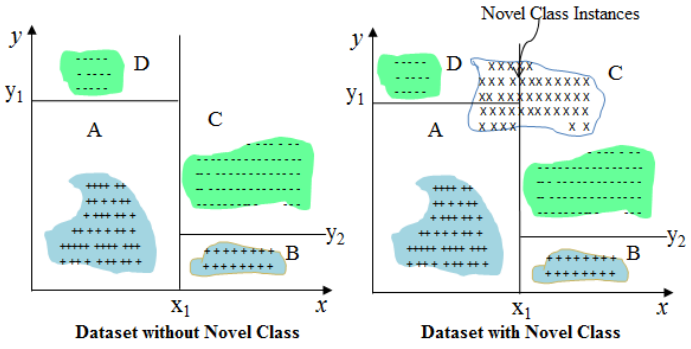

Figure 2: Instances with a fixed number of class labels (left) and instances of a novel class arriving in the data stream (right).

4. The Proposed Adaptive Ensemble Classifier

A data stream consists of a continuous sequence of instances: $\{x_1, x_2, \cdots, x_{now}\}$, where x_1 is the very first instance in the stream, and x_{now} is the latest instance, which has just arrived. One instance is observed at a time, not necessarily in equally spaced time intervals. Each instance x_i is a n-dimensional feature vector that consists of a number of attributes, $A_i = \{A_1, A_2, \cdots, A_n\}$, with a class label, $C_i \in \{C_1, C_2, \cdots, C_n\}$. Each attribute has an attribute value, $A_i = \{A_{i1}, A_{i2}, \cdots, A_{ip}\}$. A training instance x_i is labeled with a class value C_i, thus a pair (x_i, C_i) is called a labeled training instance. We refer to instances $\{x_1, x_2, \cdots, x_{now}\}$ as historical data and x_{now+1} as a test (or target) instance. Table 1 summarizes the most commonly used symbols and terms used throughout the paper.

Table 1: Commonly used symbols and terms

Symbol	Term
x_i	A data point or instance
A_i	An attribute
A_{ip}	An attribute's value
C	Total number of classes in data streams
C_i	A class label
D_i	A dataset or data chunk
K	Number of clusters
T	A decision tree
W_i	Weight for a classifier
w_i	Weight for an instance

In this paper, we have proposed five algorithms for instance weighting, decision tree construction, clustering, the adaptive ensemble classifier, and classification with novel class detection in concept-drifting data streams. These proposed algorithms are described in the following sub-sections 4.1 to 4.4. Algorithms 4 and 5 (see sub-section 4.4) outline the top level overview of the proposed adaptive ensemble model, M, for classification and novel class detection. Especially, Algorithm 4 is used at the training stage to build the ensemble model. Algorithm 5 is used at the test stage to classify test instances and detect novel classes. Figure 3 shows the training and testing phases of the adaptive ensemble model.

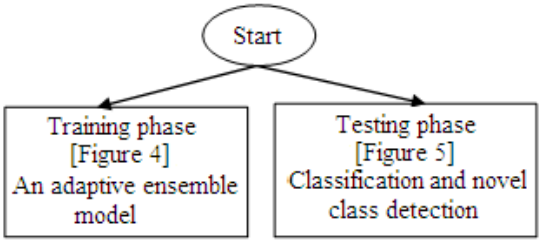

Figure 3: Training and testing phases of the adaptive ensemble model.

As discussed earlier, in this research, we build an ensemble model, M, to handle concept-drifting with novel class detection. The model M is updated continuously so that it represents the most recent concepts in data streams. Figure 4 shows the flow chart of the proposed adaptive ensemble classifier. Algorithm 4 shows the corresponding processing details. Figure 4 starts with the procedure of initializing a weight for each training instance in the training set, $x_i \in D$, using the highest posterior probability. The process of initializing weights for training instances is described in Algorithm 1 in sub-section 4.1. Then the processing generates a new dataset, D_{new}, from the original training set, D. For the initial run, a selection and replacement technique is used to generate the new dataset, D_{new}, from D. Otherwise, for subsequent runs, the training instances with higher weights are selected to construct the new dataset, D_{new}. Subsequently, it builds a decision tree, T, from D_{new}. The tree construction process is described in Algorithm 2 in sub-section 4.2. After that we further cluster the instances in D_{new} based on the similarities and differences of instances for each leaf node in T, and also calculate the threshold value based on the ratio of percentage of instances for each leaf node of T with instances in D_{new}. The similarities and differences of instances are calculated using the proposed clustering approach shown in Algorithm 3 in sub-section 4.3. We also assign a weight to the current tree $W_i \rightarrow T$ based on its classification accuracy rate for the categorization of the original training instances, $x_i \in D$. We then update the weights of the original training instances with the intention to increase the weights of those misclassified samples. Overall, the proposed ensemble model builds three decision trees and clusters the instances attached with each leaf node in each decision tree at the training stage.

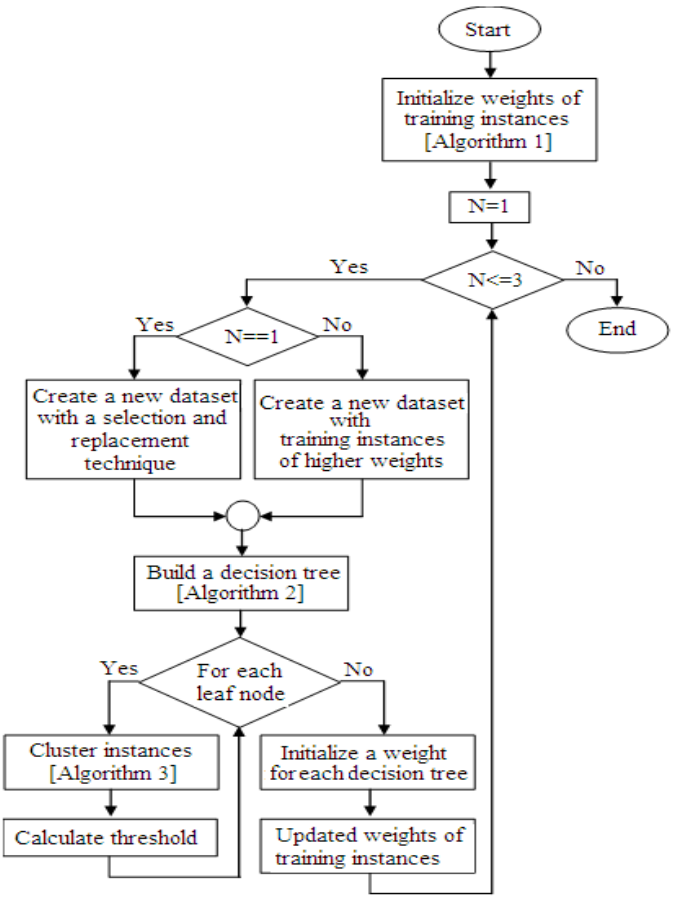

Figure 4: Flow chart of the proposed adaptive ensemble classifier.

Moreover, Figure 5 shows the flow chart of classification and novel class detection. Similarly, Algorithm 5 shows the corresponding processing details. First of all, Figure 5 divides a large data stream into equal-sized sub-data streams. This process helps to handle the infinite length problem in data streams. Then it shows the classification of each instance in each sub-data stream using the ensemble model, M, based on the weighted majority vote. If any instance does not belong to any existing data clusters, then this instance is stored in a separate database and we set $novel\_class\_flag$ to 1. Finally, if the threshold of $T_i \in M$ is abmornal and $novel\_class\_flag$ equals to 1, then a novel class has arrived. The following sub-sections introduce the proposed adaptive ensemble classifier and novel class detection step by step.

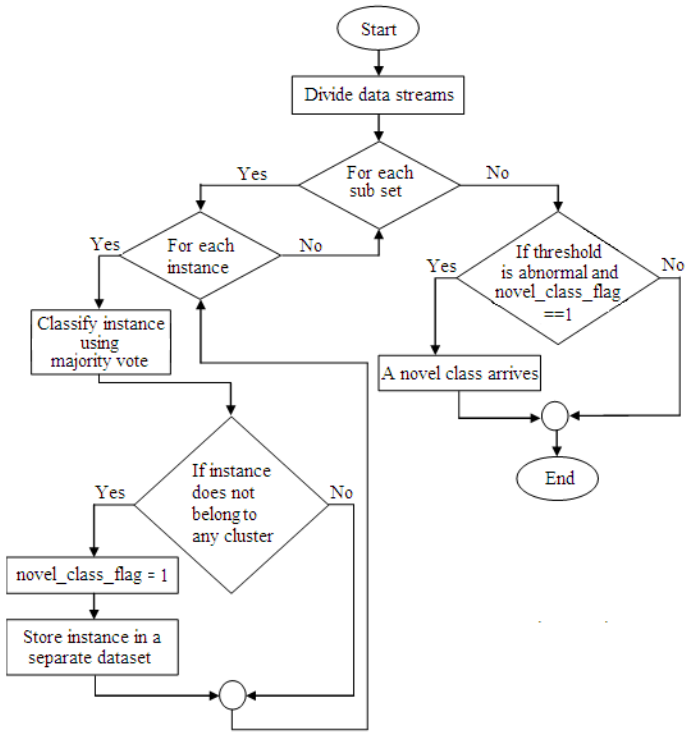

Figure 5: Flow chart of classification and novel class detection.

4.1. Weighting Training Instances

First of all, we introduce how the weight of each training instance in D is initialized using naïve Bayes (NB) classifier. Usually, most existing data mining classification techniques set the weight, $w_i \rightarrow x_i \in D$, to one or an equal value for each training instance. However in this research, setting an equal value for $w_i \rightarrow x_i \in D$ contradicts the general research intention. Assigning appropriate weights, $w_i \rightarrow x_i \in D$, also proved to increase the classification accuracy rate of mining classifiers [29]. In this research, each training instance $x_i \in D$ is initially assigned a weight based on the posterior probability of a NB classifier. Inspection of the data shows the instances with less weights are either noisy or possess unique characteristics compared to other instances with higher weights in D.

The NB classifier first estimates the prior probability, $P(C_i)$, for each class, C_i, by counting how often C_i occurs in a given training dataset, D. For each attribute, A_i, the number of occurrences of each attribute value, A_{ij}, can be counted to determine $P(A_i)$. Similarly, the probability $P(A_{ij}|C_i)$ also can be estimated by counting how often each A_{ij} occurs in $C_i \in D$. This must be done for all $A_i \in D$ and all $A_{ij} \in D$. For classifying an instance, $x_i \in D$, $P(C_i)$ and $P(A_{ij}|C_i)$ from D are used to make the prediction. This is done by combining the effects of the different attribute values, $A_{ij} \in x_i$. We estimate $P(x_i|C_i)$ by Equation 1.

$$P(x_i|C_i) = \prod_{j=1}^{n} P(A_{ij}|C_i) \tag{1}$$

To calculate $P(C_i|x_i)$, we need $P(C_i)$ for each C_i, and $P(x_i|C_i)$, and estimate the likelihood that x_i is in each C_i. The posterior probability, $P(C_i|x_i)$, is then found for C_i. The class, C_i, with the highest probability is the one chosen for the instance, x_i. We initialize the weight, w_i, for each instance $x_i \in D$ using the highest posterior probability. Algorithm 1 outlines the weighting method for training instances.

Algorithm 1 Instance Weighting using NB Classifier

Input: $D = \{x_1, x_2, \cdots, x_n\}$ // Training data.
Output: $w_i \rightarrow x_i \in D$. // Weight for each instance.
Method:

1: **for** each class, $C_i \in D$, **do**
2: Find the prior probabilities, $P(C_i)$.
3: **end for**
4: **for** each attribute value, $A_{ij} \in D$, **do**
5: Find the class conditional probabilities, $P(A_{ij}|C_i)$.
6: **end for**
7: **for** each training instance, $x_i \in D$, **do**
8: Find the posterior probability, $P(C_i|x_i)$
9: Assign the weight, $w_i \rightarrow x_i \in D$, with Maximum Likelihood (ML) of posterior probability, $w_i = P_{ML}(C_i|x_i)$;
10: **end for**

4.2. Decision Tree Induction

A decision tree (DT), also known as a classification tree, is the most popular mining approach for classification and prediction in supervised learning. A DT has many advantages including a) easy to use and efficient, b) easy to implement, c) requiring little prior knowledge, and d) built from a large dataset with many attributes (because the tree size is independent of the dataset size). Each DT represents a rule set, and the training data are recursively partitioned into smaller subsets as the decision tree is being built. A DT has three main components: nodes, leaves, and branches. Each node is labeled with an attribute by which the data is to be partitioned. Each node has a number of branches, which are labeled according to possible values of the node attribute. A branch connects either two nodes or a node and a leaf. Leaf nodes represent the final categorizations of the data. To make a classification using a DT, a test instance starts at the root node and follows the tree down the branches until a leaf node representing a class value is reached. Most algorithms for DT induction adopt a greedy (i.e., non-backtracking) approach in which DTs are constructed in a top-down recursive divide-and-conquer manner.

Moreover, during the early 1980s, Quinlan [30] developed a DT algorithm known as ID3 (Iterative Dichotomiser) using information theory. ID3 uses information gain as its attribute selection measure. The attribute with the highest information gain is chosen as the root attribute. The expected information needed to correctly classify an instance in a training dataset, D, is given in Equation 2, where p_i is the probability that an instance, $x_i \in D$, belongs to a class, C_i, and is estimated by $|C_i, D|/|D|$.

$$Info(D) = -\sum_{i=1}^{m} p_i log_2(p_i) \tag{2}$$

$Info(D)$, also known as the entropy of D, is the average amount of information needed to identify C_i of an instance, $x_i \in D$. The objective of DT approach is to iteratively partition the given dataset, D, into subsets, $\{D_1, D_2, \cdots, D_n\}$, where all instances in each D_i belong to the same class, C_i. $Info_A(D)$ is the expected information required to correctly classify an instance, x_i, from D based on the partitioning by A. Equation 3 shows $Info_A(D)$ calculation, where $\frac{|D_j|}{|D|}$ acts as the weight of the jth partition.

$$Info_A(D) = \sum_{j=1}^{n} \frac{|D_j|}{|D|} \times Info(D_j) \tag{3}$$

Information gain is defined as the difference between the original information requirement and the new requirement that is shown in Equation 4.

$$Gain(A) = Info(D) - Info_A(D) \tag{4}$$

C4.5, a successor of ID3, uses an extension to information gain known as a gain ratio [31]. It applies a kind of normalization to information gain using a "split information" value defined analogously with $Info(D)$ as shown in Equation 5.

$$SplitInfo_A(D) = -\sum_{j=1}^{n} \frac{|D_j|}{|D|} \times log_2\left(\frac{|D_j|}{|D|}\right) \tag{5}$$

The attribute with the maximum gain ratio is selected as the splitting attribute. The gain ratio is defined in Equation 6.

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)} \tag{6}$$

Furthermore, C5.0, a successor of C4.5, improves the performance of DTs using boosting. Boosting is an approach of combining different classifiers, but does not always help when the training data contains a lot of noise. In C5.0, each classifier is assigned a vote, and the test instance is assigned a class value, C_i, using a voting technique. CART (classification and regression trees) is a process of generating a binary tree for decision making [32]. CART handles missing data and contains a pruning strategy. The SPRINT (Scalable Parallelizable Induction of Decision Trees) algorithm uses an impurity function called gini index to find the best split [33]. Equation 7 defines the gini for a dataset, D, where, p_j, is the frequency of class $C_j \in D$.

$$Gini(D) = 1 - \sum_{j=1}^{m} p_j^2 \tag{7}$$

The goodness of a split of D into subsets D_1 and D_2 is defined by Equation 8.

$$gini_{split}(D) = \frac{n_1}{n(gini(D_1))} + \frac{n_2}{n(gini(D_2))} \tag{8}$$

In this way, the split with the best gini value is chosen. In the last decade, much research has been done for optimal feature selection and classification, which adopted hybrid strategies involving evolutionary algorithms and inductive decision tree learning [34, 35, 36].

Algorithm 2 outlines the classic ID3 decision tree induction. In this research, we have employed both ID3 and C4.5 algorithms. We are especially motivated by C5.0, the boosting approach, and employ three decision trees and a weighted voting technique for the classification of each test instance.

Algorithm 2 Decision Tree Learning

Input: $D_i = \{x_1, x_2, \cdots, x_n\}$ // Training data.
Output: T, Decision tree.
Method:

```
1:  T = ∅;
2:  Determine the best splitting attribute;
3:  T = Create the root node and label it with the splitting attribute;
4:  T = Add an arc to the root node for each split predicate and label it with a corresponding attribute value;
5:  for each arc do
6:      D_ij = Create sub dataset by applying splitting predicate to D_i;
7:      if stopping point reached for this path, then
8:          T' = Create leaf node and label with an appropriate class;
9:      else
10:         T' = DTBuild(D_i);
11:     end if
12:     T = Add T' to arc;
13: end for
```

As mentioned earlier, three DT classifiers will be generated and employed to reflect the most recent concepts in data streams. A weighted majority vote of the three classifiers is used to classify each instance in the test data stream. The DT classifiers are also used to measure the abnormal distribution of the training and test data in order to identify the potential arrival of novel class instances in data streams.

4.3. Data Clustering

Clustering is an example of unsupervised learning in machine learning and data mining research. It has been widely used in many real-world application domains, including biology, medicine, anthropology, marketing, pattern recognition, and image processing. Clustering is also called data segmentation, because clustering partitions large datasets into groups according to their similarities and differences. It is a form of learning by observation. Data clustering has recently become a highly active research topic, because assigning class labels to a large number of instances can be a very costly process. The goal of clustering is to determine the intrinsic grouping for a set of unlabeled data. It is the process of grouping the instances into clusters (or classes). Dissimilarities are assessed based on the attribute values describing the instances. Also, a cluster usually should consist of a group of instances that are similar to one another and are dissimilar to instances in other clusters.

Given a dataset, $D_i = \{x_1, x_2, \cdots, x_n\}$, of n number of instances, a similarity based clustering method constructs K clusters of the instances and $K \leq n$, which together satisfy the

following requirements: (1) each cluster must contain at least one instance, and (2) each instance must belong to exactly one cluster. A similarity measure, $sim(x_i, x_l)$, is also defined for the comparison of any two training instances, x_i, $x_l \in D_i$. A good clustering is that instances in the same cluster are "close" or related to each other, whereas instances of different clusters are "far apart" or very different from one another. Each instance $x_i \in D_i$ is assigned to one cluster K_j, $1 \leq j \leq k$, where $D_i \rightarrow \{1, \cdots, k\}$, and k is an integer value. Given a cluster, $K_j, \forall x_{jl}, x_{jm} \in K_j$, and $x_i \notin K_j$, $sim(x_{jl}, x_{jm}) > sim(x_{jl}, x_i)$. Algorithm 3 outlines the similarity based clustering method used in this research.

Algorithm 3 Similarity Based Clustering

Input: $D_i = \{x_1, x_2, \cdots, x_n\}$ // A set of instances.
Output: A set of K clusters.
Method:

```
1:  K_1 = {x_1};
2:  K = {K_1};
3:  k = 1;
4:  for i = 2 to n do
5:      find x_m in some cluster K_m in K such that sim(x_i, x_m) is
        the maximum;
6:      if sim(x_i, x_m) ≥ threshold_value then
7:          K_m = K_m ∪ x_i
8:      else
9:          k = k + 1;
10:         K_k = {x_i};
11:     end if
12: end for
```

The proposed clustering Algorithm 3 is used to especially measure the class distribution of the training and test data and monitor the arrival of an exceptional novel class. It confirms the arrival of a novel class after the three DT classifiers show abnormal class distributions in leaf nodes and indicate a novel class may arrive. The detailed approach for the ensemble classifier generation and novel class detection is introduced in the following.

4.4. An Adaptive Ensemble Classifier and Novel Class Detection

As mentioned earlier, Algorithm 4 presents the detailed procedure for the generation and training of the proposed adaptive ensemble classifier. It starts with initializing a weight for each instance in the training dataset, $x_i \in D$, using Algorithm 1. Then Algorithm 4 generates a new dataset, D_{new}, from the original training set, D, and builds a decision tree, T, from D_{new} using Algorithm 2. After that Algorithm 4 further clusters the training instances using Algorithm 3 and calculates the threshold for each leaf node in the decision tree. Finally, Algorithm 4 assigns a weight to the current decision tree based on its classification accuracy rate for the categorization of the original training instances.

Moreover, Algorithm 4 also updates the weight of each training instance, $x_i \in D$, in the following way so that if an training instance is correctly classified then its weight is decreased, or if misclassified then its weight will remain unchanged thus comparatively increased. First of all, we assign an error rate for each instance in D. If an instance, x_i, is misclassified, then $err(x_i)$ is 1. Otherwise, it is 0. We also calculate the overall misclassification error rate: $error(T_i) = \sum_i^d w_i * err(x_i)$, where $err(x_i)$ is the misclassification error and w_i is the initialized weight for each instance. Subsequently, if x_i is correctly classified, its weight will be decreased by multipling $\frac{error(T_i)}{(1-error(T_i))}$. As a result, the weights of misclassified instances are increased and the weights of correctly classified instances are decreased. Once the weights of all of the correctly classified instances are updated, the weights for all instances including the misclassified instances are normalized so that their sum remains the same as it was before. The normalizasion processing multiplies the sum of the old weights, divided by the sum of the new weights.

Algorithm 4 The Adaptive Ensemble Classifier

Input: $D = \{x_1, x_2, \cdots, x_n\}$ // Training dataset.
Output: M, An ensemble model.
Method:

```
1:  for each instance, x_i ∈ D, do
2:      Initialize the weight, w_i → x_i ∈ D, using Algorithm 1.
3:  end for
4:  for 1 to N, where N = 3 do
5:      if N = 1 then
6:          Generate a new dataset, D_new, from D using a selection
            and replacement technique.
7:      else
8:          Generate a new dataset, D_new, from D with x_i of higher
            weights.
9:      end if
10:     Build a decision tree, T, from D_new using Algorithm 2.
11:     for each leaf node in T do
12:         Cluster the instances of D_new using Algorithm 3.
13:         Calculate the threshold value based on the ratio of per-
            centage of instances in this leaf node and instances in
            D_new.
14:     end for
15:     Initialize the weight, W_i → T, based on its classification
        accuracy rate for the categorization of the instances, x_i ∈
        D.
16:     Update the weight, w_i, of each x_i ∈ D.
17: end for
```

The above processing iterates three times. Overall, the ensemble model, M, builds three trees. When classifying continuous data streams in real-time, instances in each sub data stream are labeled by M. The model assigns a class label to an instance using majority of weighted votes of the three decision trees. A new training dataset D_{new+1} is created by adding recently arrived instances in the data stream to the latest D_{new}, and a new T_{new} is built using D_{new+1}. Figure 6 shows the ensemble model, where decision trees are built from each data stream.

T_{new} built with the most recent dataset is also assigned a weight, $W \rightarrow T_{new}$, based on the classification accuracy rate

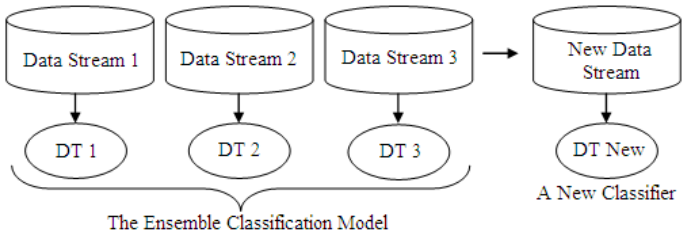

Figure 6: An example of DT_S construction from each data stream.

for D_{new+1}. If T_{new} has a higher weight (a better classification accuracy rate) than any of $T_i \in M$, then the minimum weighted $T \in M$ will be replaced by T_{new}. Thus T_{new} is added to M. Then instances $x_i \in D_{new+1}$ are clustered based on their similarities for each leaf node in T_{new}, and the threshold for each leaf node in T_{new} is also calculated. In this research, a novel class can be automatically detected. For example, if the number of instances in D classified by a leaf node of $T_i \in M$ increases or decreases in comparison with the threshold value calculated earlier, then a novel class may arrive. Also if the newly arrived instances do not belong to any existing clusters in all the three trees, which confirms a novel class arrives. The above processing details on the data stream classification and novel class detection are also summaried in Algorithm 5.

Algorithm 5 Classification and Novel Class Detection

Input: Real data streams and the ensemble model, M.
Output: Labeling data streams and novel class detection
Method:

```
1: Divide the data stream into equal-sized sub-data streams.
2: for each sub-data stream do
3:    for each instance do
4:       Classify the instance using each T_i ∈ M.
5:       Return a weighted vote (which counts as one vote).
6:       Assign a class label with majority of weighted votes.
7:       if an instance does not belong to any existing clusters
         then
8:          novel_class_flag = 1
9:          Store this instance in a separate dataset.
10:      end if
11:   end for
12: end for
13: if threshold of T_i ∈ M is abnormal and
    novel_class_flag == 1 then
14:   A novel class arrives.
15: end if
```

5. Experiments

The ensemble classifier is evaluated using data streams representing concept-drifting in challenging real-life applications. In this section, we describe the test datasets, experimental environments, and present the evaluation results.

5.1. Datasets

Data stream mining is a process of analyzing online data to discover patterns. It uses sophisticated mathematical algorithms to segment the continuous data and evaluate the probability of future events. Table 2 describes the datasets from UCI machine learning repository [12] used in our experimental analysis to evaluate the developed ensemble classifier and novel class detection function. The employed test data include the NSL-KDD dataset, the large soybean and the image segmentation databases. Each test dataset employed in this research is roughly equivalent to a two-dimensional spreadsheet or a database table.

Table 2: Test dataset descriptions

Dataset	No of Att.	Att. Types	Instances	Classes
NSL-KDD data	41	Real & Nominal	25192	23
Soybean data	35	Nominal	683	19
Image seg. data	19	Real	2310	7

5.1.1. The NSL-KDD Dataset

The Knowledge Discovery and Data Mining 1999 (KDD99) competition data contain simulated intrusions in a military network environment. It is often used as a benchmark to evaluate models handling concept-drift. The NSL-KDD dataset is the new version of the KDD99 dataset, which solved some of the inherent problems of the previous dataset [37]. Although the NSL-KDD dataset still suffers from some of the problems raised by McHugh [38], the main advantage of the NSL-KDD dataset is that the training and test data do not include redundant and duplicate examples. Thus it becomes affordable for us to run the experiments on the complete set of training and test data without the need to randomly select a small portion of them. Each record in the NSL-KDD dataset consists of 41 attributes and 1 class label. Table 3 shows the number of training instances of normal and intrusion classes in the NSL-KDD dataset.

Table 3: Training instances in the NSL-KDD dataset

Types	Classes	Instances
Normal	normal	13449
Denial of service (DoS)	back, land, neptune, pod, smurf, tenddrop	9234
Remote to user (R2U)	ftp_write, guess_passwd, impa, multihop, phf, spy, warezclient, warezmaster	209
User to root (U2R)	buffe_overflow, perl, loadmodule, rootkit	11
Probing	ipsweep, nmap, satan, protsweep	2289
	Total = 23	Total = 25192

5.1.2. The Large Soybean Database

Another test dataset employed in this research is a database for large soybeans. There are 35 attributes in this dataset and all attributes are nominalized. There are altogether 683 data instances with 19 class values in this dataset.

5.1.3. The Image Segmentation Database

The goal of this dataset is to provide an empirical basis for research on image segmentation and boundary detection. This dataset is also used in this research to evaluate the ensemble classifier for novel class detection. There are overall 2310 data instances in this database. They are described by 19 attributes and 7 class values including brickface, sky, foliage, cement, window, path, and grass.

5.2. Experimental Setup

We implement the above proposed algorithms (Algorithms 1-5) in Java for experimental analysis and performance evaluation. We use NetBeans IDE 7.1 in redhat enterprise linux 5 for Java coding. NetBeans IDE is the first IDE providing support for JDK 7 and Java EE 6 (http://netbeans.org/index.html). The experiments were conducted using a machine with an Intel Core 2 Duo Processor 2.0 GHz processor (2 MB Cache, 800 MHz FSB) and 1 GB of RAM. The code for ID3 and C4.5 is adopted from Weka3, open source data mining software written in Java [39]. Weka3 is a collection of machine learning algorithms for data mining tasks, which contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. These learning algorithms can be either applied directly to a dataset or called from our own coding.

There are various approaches to determine the performance of data stream mining classifiers. The performance can simply be measured by counting the proportion of correctly classified instances in an unseen test dataset. Table 4 summarizes the symbols and terms used throughout in Equations 9 to 11. These equations are used to especially evaluate the performance of the proposed ensemble classifier. Equation 9 is used to calculate the percentage of how many novel class instances are misclassified as existing classes, with Equation 10 employed to produce the percentage of how many existing class instances are falsely identified as novel classes and Equation 11 used to calculate the total misclassification error.

Table 4: Used symbols and terms in Equations 9-11

Symbol	Term
N	Total instances in the data stream
N_c	Total novel class instances in the data stream
F_p	Total existing class instances misclassified as novel classes
F_n	Total novel class instances misclassified as existing classes
F_e	Total existing class instances misclassified
M_{new}	% of novel class instances misclassified as existing classes
F_{new}	% of existing class instances falsely identified as novel classes
ERR	Total misclassification error

$$M_{new} = \frac{F_n * 100}{N_c} \tag{9}$$

$$F_{new} = \frac{F_p * 100}{N - N_c} \tag{10}$$

$$ERR = \frac{(F_p + F_n + F_e) * 100}{N} \tag{11}$$

5.3. Experimental Analysis

We compare the proposed ensemble model for classification and novel class detection in concept-drifting data streams with the C4.5 decision tree induction classifier and the k-Nearest Neighbors (k-NN) algorithm. Both C4.5 and k-NN classifiers are widely used in real-life classification problems. They are employed as two baseline systems in our experiments. We will henceforth use the following acronyms: EM for the ensemble model, C4.5 for the C4.5 decision tree classifier, and k-NN for the k-Nearest Neighbors algorithm, for the discussion of the experimental results and performance comparison.

Table 5, 6, and 7 tabulate the performance details of classifiers in the presence of novel classes respectively for the test NSL-KDD dataset, the soybean and the image segmentation databases. Each table shows the number of training and test instances with existing and novel class labels used in each of the three test experimental runs for one test dataset. The results indicate the ensemble model trained with a fixed number of class labels shows great robustness to learn and classify new class concepts in each test data stream and outperforms the other two traditional supervised baseline algorithms.

Table 5: Performance comparison of classifiers in the presence of novel classes using the NSL-KDD dataset

Instances	Existing classes	Novel classes	Classifier	Correctly classified instances (%)
Training: 13449 Testing: 9234	1	6	EM C4.5 k-NN	90.74 86.49 80.75
Training: 22683 Testing: 220	7	10	EM C4.5 k-NN	85.00 44.09 41.36
Training: 22903 Testing: 2289	19	4	EM C4.5 k-NN	79.03 52.55 51.90

Table 6: Performance comparison of classifiers in the presence of novel classes using the soybean dataset

Instances	Existing classes	Novel classes	Classifier	Correctly classified instances (%)
Training: 192 Testing: 90	5	2	EM C4.5 k-NN	95.55 81.11 78.88
Training: 364 Testing: 172	10	3	EM C4.5 k-NN	95.93 71.51 68.02
Training: 630 Testing: 53	15	4	EM C4.5 k-NN	93.23 77.81 75.56

Figures 7, 8, and 9 further illustrate the comparison of the accuracy rates of classifiers in concept-drifting data streams on each test dataset. In order to compare the performances in intense concept-drifting classification tasks, among each test experimental dataset, test instances especially with a large number of novel classes are used to evaluate the ensemble and the two baseline classification models and produce the accuracy results shown in Figures 7, 8, and 9.

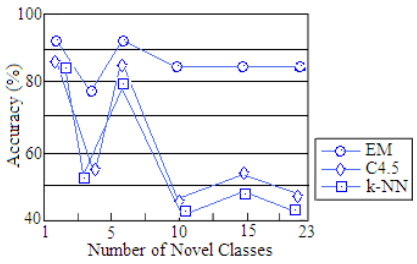

Figure 7: The comparison of accuracy rates of classifiers in concept-drifting using the NSL-KDD dataset.

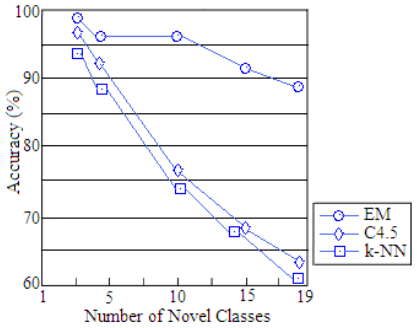

Figure 8: The comparison of accuracy rates of classifiers in concept-drifting using the soybean dataset.

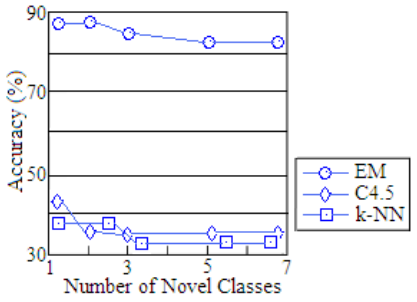

Figure 9: The comparison of accuracy rates of classifiers in concept-drifting using the image segmentation dataset.

Table 7: Performance comparison of classifiers in the presence of novel classes using the image segmentation dataset

Instances	Existing classes	Novel classes	Classifier	Correctly classified instances (%)
Training: 180 Testing: 50	6	1	EM C4.5 k-NN	90.00 40.00 38.00
Training: 987 Testing: 315	5	2	EM C4.5 k-NN	90.13 34.51 33.72
Training: 1343 Testing: 417	4	3	EM C4.5 k-NN	87.53 31.81 31.76

Table 8 also summarizes the error metrics of each classifier for each test dataset. The columns headed by M_{new}, F_{new}, and ERR report the values of the corresponding metrics on an entire dataset. The error rate values for each classifier shown in the above three columns are produced using Equations 9-11.

Table 8: Performance comparison of different classifiers

Dataset	Classifier	M_{new}	F_{new}	ERR
NSL-KDD	EM	0.9	0.4	0.7
	C4.5	1.9	0.9	1.5
	k-NN	5.8	0.9	3.3
Soybean	EM	3.7	0.0	2.0
	C4.5	5.0	1.0	4.0
	k-NN	5.6	1.9	5.1
Image Seg.	EM	0.0	0.0	3.3
	C4.5	6.2	2.3	8.0
	k-NN	3.7	3.0	10.4

Moreover, Table 9 summarizes the accuracy rate of each classifier on each dataset using 10-fold cross-validation. The 10-fold cross-validation is a process of breaking a dataset into 10 subsets of size $\frac{n}{10}$, where n is the total number of the instances in the dataset. Then each classification model is trained with nine subsets and tested on the remaining subset. This processing repeats 10 times and takes a mean accuracy rate for each classification model. The classification rates shown in Table 9 are calculated based on the number of instances that are correctly classified in each dataset. Similarly, the misclassification rates are calculated based on the number of instances that are incorrectly classified in each test dataset.

Table 9: Performance comparison of classifiers using 10-fold cross-validation

Dataset with instances	Classifier	Classification rate (%)	Misclassification rate (%)
NSL-KDD (25192 instances)	EM C4.5 k-NN	92.65 86.35 83.30	7.34 13.64 16.69
Soybean (683 instances)	EM C4.5 k-NN	98.24 91.50 89.75	1.75 8.49 10.24
Image Seg. (2310 instances)	EM C4.5 k-NN	92.77 90.34 84.71	7.22 9.65 15.28

Figure 10, 11, and 12 show the ROC (receiver operating characteristic) curves of classifiers on each dataset. A ROC curve

shows the trade-off between the true position rate and the false positive rate for a classifier. The true positive rate is the proportion of instances that are correctly classified, and the false positive rate is the proportion of instances that are misclassified. The vertical axis of an ROC curve represents the true positive rate. The horizontal axis represents the false positive rate. A classification model closest to the diagonal line of the ROC curve is the least accurate model. In all these three figures, the ensemble model shows the most promising and competitive accuracy rates followed by the performance of the C4.5 decision tree classifier and with the k-NN model achieving the last position in classification performance on the selected test sets.

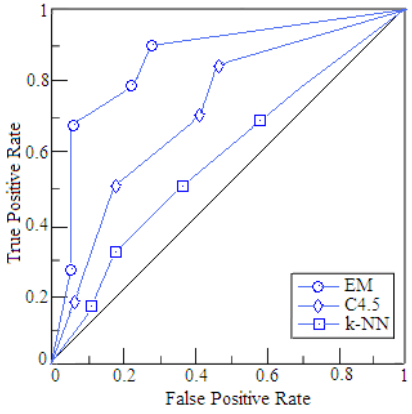

Figure 10: The ROC curves of classifiers using the entire NSL-KDD dataset.

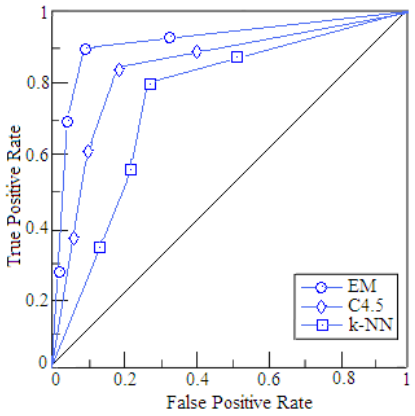

Figure 11: The ROC curves of classifiers using the entire soybean dataset.

Overall, the above experimental results indicate that the ensemble model has great flexibility and adaptability in novel class detection in data stream environments in comparison with the traditional C4.5 and k-NN classifiers. The generated decision trees and the similarity-based clustering method embedded in the ensemble classifier constantly monitor and identify the arrival of exceptional classes. The ensemble model representing new concepts in data streams also employs a majority weighted voting technique for classification. The above evaluation results prove that it outperforms the traditional classifiers and improves classification accuracy rates greatly in challenging real-life data stream applications.

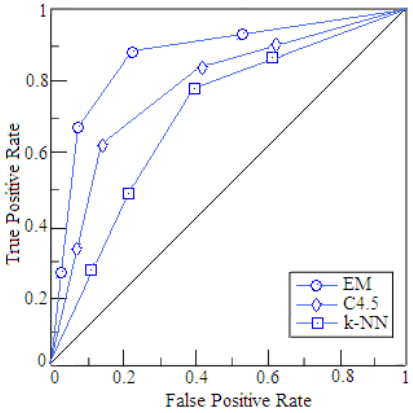

Figure 12: The ROC curves of classifiers using the entire image dataset.

6. Conclusions

In this paper, we introduce an adaptive ensemble model for classification and novel class detection in concept-drifting data streams. Especially, novel class instances in data streams can be automatically detected in our approach. Our work addresses challenging issues in data stream classifications such as infinite length, limited labeled data, concept-drift, and concept-evolution. The proposed adaptive ensemble model generally continuously updates itself with newly arrived data instances so that it represents the most recent concepts in data streams. Another main objective of this research is to minimize the total misclassification error (ERR) in concept-drifting classification tasks. We tested the performance of the ensemble model on three benchmark datasets borrowed from UCI machine learning repository. The experimental results proved that this ensemble classifier efficiently detects the arrival of novel class instances and also greatly improves the classification accuracy rates under different circumstances. In future work, we will focus on concept-drifting under dynamic feature/attribute sets to further extend the capabilities of the current adaptive ensemble classifier.

Acknowledgment

We appreciate the support for this research received from the European Union (EU) sponsored (Erasmus Mundus) cLINK (Centre of excellence for Learning, Innovation, Networking and Knowledge) project (EU EACEA 42/11).

References

[1] S.-H. Liao, P.-H. Chu, P.-Y. Hsiao, Data mining techniques and applications - A decade review from 2000 to 2011, Expert Systems with Applications 39 (2012) 11303–11311.

[2] M. M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, N. C. O. Kevin W. Hamlen, Facing the reality of data stream classification: coping with scarcity of labeled data, Knowledge and Information Systems 33 (1) (2012) 213–244.

[3] J. Read, A. Bifet, G. Holmes, B. Pfahringer, Scalable and efficient multi-label classsification for evolving data streams, Machine Learning 88 (1-2) (2012) 243–272.
[4] J. Read, A. Bifet, G. Holmes, B. Pfahringer, Efficient Multi-label classification for evolving data Streams, 2010.
[5] I. Zliobaite, Learning under concept drift: an overview, 2009.
[6] W. hua XU, Z. QIN, Y. CHANG, Clustering feature decision trees for semi-supervised classification from high-speed data streams, Journal of Zhejiang University-SCIENCE C (Computers & Electronics) 12 (8) (2011) 615–628.
[7] M. M. Masud, J. Gao, L. Khan, J. Han, B. Thuraisingham, Classification and novel class detection in concept-drifting data streams under time constraints, IEEE Transactions on Knowledge and Data Engineering 23 (6) (2011) 859–874.
[8] A. Biswas, D. M. Farid, C. M. Rahman, A new decision tree learning approach for novel class detection in concept drifting data stream classification, Journal of Computer Science and Engineering (JCSE-UK) 14 (1) (2012) 1–8.
[9] D. M. Farid, C. M. Rahman, Novel class detection in concept-drifting data stream mining employing decision tree, In Proc. of the 7th International Conference on Electrical and Computer Engineering (ICECE 2012), Dhaka, Bangladesh (2012) 630–633, and IEEE Xplore Digital Archive.
[10] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Transactions on Neural Networks 22 (10) (2011) 1517–1531.
[11] L. Su, H. yan Liu, Z.-H. Song, A new classification algorithm for data stream, I.J.Modern Education and Computer Science 4 (2011) 32–39.
[12] A. Frank, A. Asuncion, UCI machine learning repository, URL `http://archive.ics.uci.edu/ml`, 2010.
[13] W. Hou, B. Yang, C. Wu, Z. Zhou, RedTrees: A relational decision tree algorithm in streams, Expert Systems with Applications 37 (2010) 6265–6269.
[14] C. Surace, K. Worden, Novelty detection in a changing environment: a negative selection approach, Machanical Systems and Signal Processing 24 (4) (2010) 1114–1128.
[15] C.-J. Tsai, C.-I. Lee, W.-P. Yang, Mining decision rules on data streams in the presence of concept drifts, Expert Systems with Applications 36 (2009) 1164–1178.
[16] S. J. Haggett, D. F. Chu, Evolving novelty detectors for specific applications, Neurocomputing 72 (10-12) (2009) 2392–2405.
[17] S. J. Haggett, D. F. Chu, I. W. Marshall, Evolving a dynamic predictive coding mechanism for novelty detection, Knowledge-Based Systems 21 (3) (2008) 217–224.
[18] A. Zhou, F. Cao, W. Qian, C. Jin, Tracking clusters in evolving data streams over sliding windows, Knowledge and Information Systems 15 (2) (2008) 181–214.
[19] J. Z. Kolter, M. A. Maloof, Dynamic weighted majority: an ensemble method for drifting concepts, Journal of Machine Learning Research 8 (2007) 2755–2790.
[20] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu, A framework for on-demand classification of evolving data streams, IEEE Transactions on Knowledge and Data Engineering 18 (5) (2006) 577–589.
[21] B.-R. Dai, J.-W. Huang, M.-Y. Yeh, M.-S. Chen, Adaptive clustering for multiple evolving streams, IEEE Transactions on Knowledge and Data Engineering 18 (9) (2006) 1166–1180.
[22] M. M. Gaber, P. S. Yu, Detection and classification of changes in evolving data streams, International Journal of Information Technology & Decision Making 5 (4) (2006) 659–670.
[23] Y. Yang, X. Wu, X. Zhu, Combining Proactive and Reactive Predictions for Data Streams, Proc. of ACM SIGKDD (2005) 710–715.
[24] P. P. Angelov, X. Zhou, Evolving fuzzy-rule-based classifiers from data streams, IEEE Transactions on Fuzzy Systems 16 (6) (2008) 1462–1475.
[25] M. Markou, S. Singh, Novelty detection: a reviewpart 1: statistical Approaches, Signal Processing 83 (12) (2003) 2481–2497.
[26] M. Markou, S. Singh, Novelty detection: a reviewpart 2:: neural network based approaches, Signal Processing 83 (12) (2003) 2499–2521.
[27] H.-F. Li, S.-Y. Lee, Mining frequent itemsets over data streams using efficient window sliding techniques, Expert Systems with Applications 36 (2009) 1466–1477.
[28] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine Learning 23 (1) (1996) 69–101.
[29] D. M. Farid, C. M. Rahman, Assigning weights to training instances increases classification accuracy, International Journal of Data Mining & Knowledge Management Process 3 (1) (2013) 13–25.
[30] J. R. Quinlan, Induction of Decision Tree, Machine Learning 1 (1986) 81–106.
[31] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA .
[32] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and Regression Trees, Statistics probability series, Wadsworth, Belmont .
[33] J. Shafer, R. Agrawal, M. Mehta, SPRINT: A scalable parallel classifier for data mining, Morgan Kaufmann (1996) 544–555.
[34] D. Turney, Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm, Journal of Artificial Intelligence Research (1995) 369–409.
[35] S. R. Safavian, D. Landgrebe, A Survey of Decision Tree Classifier Methodology, IEEE Transactions on Systems, Man and Cybernetics 21 (3) (1991) 660–674.
[36] W. Y. Loh, X. Shih, Split selection methods for classification tree, Statistica Sininca 7 (1997) 815–840.
[37] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, Proc. of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA) .
[38] J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by Lincoln Laboratory, ACM Transaction on Information and System Security (TISSEC) 3 (4) (2000) 262–294.
[39] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The WEKA Data Mining Software: An Update, SIGKDD Explorations 11 (1), URL `http://www.cs.waikato.ac.nz/ml/weka/`.