CHAPTER 7

DYNAMIC ENSEMBLE GENERATION AND CLASSIFICATION
ALGORITHM

This chapter presents the detailed discussion about the various approaches towards an evaluation of a Dynamic Ensemble Generation and Classification (DEGC) of high-speed data streams using impact measures. It also presents a detailed discussion about the results being produced by different methods proposed. The method has been implemented and evaluated for their performance in various metrics.

In earlier days the data records have been maintained statically and the users have followed the same strategy in presenting the data. However, modern information technology has opened the gate for the users to represent their data in their way. For example, any government would collect their data in their format and force others to use the same one. While maintaining such format which is fluctuating would introduce more challenges to the users as well. The class ensemble approach is discussed, towards the problem of data maintenance. Given a set of data $D_s$, which contains $N$ number of samples with $k$ number of classes, classifying the $N$ number of samples from the data set $D_s$ towards $k$ number of class is a challenging issue where the $k-l$ number of samples does not belong to anyone of the class from $k$. This requires the attention of maintaining newly arrived samples in the class $K+1$. In case of classification, the ensembles have to follow a format and so that it can be classified accurately.

The dynamic ensembles are one which can be received in any format either it has been available in the set or a new one. As a tweet from the twitter, the topic and the
format of the data may be of any format. However, it must be classified into a class so that it can be maintained to propagate to the other users of the same group or interest. The ensembles may be seasonal or non-seasonal, but still, there will be the certain class of ensembles which has to be maintained in all the periods. The ensemble classes would change over a period but the, common ensemble classes have to be maintained in all the stages to perform the classification accurately.

The class-based ensembles are generated from the online data streams based on their type of data or the topic of conversation. For example, the seasonal tweet in a Twitter dataset would vary according to the seasons like festivals. However, the ensembles can be generated from the tweets and would be used to perform classification of the tweets. A terror attack oriented tweets would occur at any point in time or any time window of the year. So maintaining the tweets in a time window based is more essential and such tweets or the ensembles has to be maintained in all the time.

This research work is introduced in a min-max influence based ensemble generation and classification approach. The minimum ensemble influence measure is the value which denotes a minimum quantitative value an ensemble to have to be carried to the next time window. It has been computed based on the number of occurrence of the ensemble in the particular time window or the number of time it has been discussed in the particular time window. Similarly, the maximum influence measure is the value which is computed based on the number of occurrence of the ensemble class in any time window. The minimum value has been used for a special class ensemble where the maximum influence measure has been considered for the generic class ensembles.

To improve the performance of classification, this research work presents a dynamic ensemble generation algorithm for the online data streams. An Impact
measure based approach is discussed to improve the performance of classification. For each class identified, the method computes the impact measure towards each time window identified. The method classifies the ensemble topics into different classes and for each of them, different impact threshold is used. According to the impact measures and the thresholds, the ensemble will be carried to the future time window. The approach improves the performance of ensemble maintenance and classification performance.

With the rapid development of incremental learning and online learning, mining tasks in the context of data stream have been widely studied. Generally, data stream mining refers to the mining tasks that are conducted on a (possibly infinite) sequence of rapidly arriving data records. As the environment where the data is collected may change dynamically, the data distribution may also change accordingly. This phenomenon, referred to as concept drift, is one of the most important challenges in data stream mining.

A data stream mining technique should be capable of constructing and dynamically updating a model in order to learn dynamic changes of data distributions, i.e., to track the concept drift. For classification problems, concept drift is formally defined as the change of joint distribution of data, i.e., $P(x,y)$. Where $x$ is the feature vector and $y$ is the class label. Over the past few decades, concept drift has been widely studied. The majority of the previous works focus on the concept drift caused by the change in the class-conditional probability distribution, i.e., $P(x|y)$. In comparison, class evolution, which is another factor that induces concept drift, has attracted relatively less attention. Briefly speaking, class evolution is concerned with certain types of change in the prior probability distribution of classes, i.e., $P(y)$ and usually corresponds to the emergence of a novel class and the disappearance of an outdated class. Class evolution frequently occurs in practice. For example, new topics frequently appear on Twitter and outdated topics are forgotten with time. Besides, old
topics, e.g., topics on festivals, may also become popular again. Such phenomena can also be observed from other types of data streams, such as the click-through data of news or advertisements since the interests of clients may change over time. In some literature, class evolution is also called class-incremental learning or concept evolution. More formally, let $C_t$ denote the set of classes whose prior probability is positive at time stamp $t$. The class evolution involves the following forms

- Class emergence represents an example of an unknown class is received at the current time. That is, class $c$ emerges at time $t$ if $c \in C_1 \cup C_2 \cup \ldots \cup C_{t-1}$ and $c \in C_t$, such a class is called a novel class.

- Class disappearance describes the situation in which the example of an existing class would not be received in the next time stamp. That is, if class $c$ disappeared at time $t$, then $c \in C_{t-1}$ and $c \notin C_t$.

- Class reoccurrence defines the point where a disappeared class recurs later in the data stream. Class $c$ is a recurring class at time $t$, if $c \in C_1 \cup C_2 \cup \ldots \cup C_{d-1}$, $c \notin C_d \cup \ldots \cup C_{t-1}$ and $c \notin c \in C_t$.

Since the number of classes may change when class evolution happens, the model needs to be adapted not only to capture the distribution of existing classes but also to identify that of the novel classes. At the same time, the effects of disappeared classes need to be removed from the model. Hence, in comparison to the change of class conditional probability, class evolution brings additional challenges to data stream mining. In literature, a few approaches have been proposed to address class evolution problems. Although they have shown promising performance, they implicitly assume that classes emerge or disappear transiently. In other words, the example generation rate (EGR, i.e., the number of examples generated per-unit-time) of a class switches between two states, i.e., a constant positive value and zero. However, in a real-world scenario, it is more likely that classes evolve gradually. For example, in an early stage, an event may be discussed by a few participants on Twitter; the topic grows in
popularity over a period and then eventually fades away from attention. Motivated by this consideration, this work investigates the class evolution problem with gradually evolved classes. The gradual evolution of classes refers to the case that classes appear or disappear in a gradual rather than transient manner, i.e., the EGR changes more smoothly. A novel class-based (CB) ensemble approach, namely Dynamic Ensemble Generation and Classification (DEGC), is proposed. In contrast to the above-mentioned existing approaches, which process a data stream in a chunk-by-chunk manner and build a base learner for each chunk, DEGC maintains a base learner for every class that has ever appeared and updates the base learners whenever a new example arrives (i.e., in a one-pass manner). Furthermore, a novel under-sampling method is also designed to cope with the dynamic class-imbalance problem induced by gradual class evolution.

7.1 Existing Methodology

For classification problems, concept drift is formally defined as the change of joint distribution of data, i.e., $P(x, y)$ where $x$ is the feature vector and $y$ is the class label. Over the past few decades, concept drift has been widely studied. The majority of the previous works focus on the concept drift caused by the change in the class-conditional probability distribution, i.e., $P(x|y)$. In comparison, class evolution, which is another factor that induces concept drift, has attracted relatively less attention. Briefly speaking, class evolution is concerned with certain types of change in the prior probability distribution of classes, i.e., $p(y)$ and usually corresponds to the emergence of a novel class and the disappearance of an outdated class. Class evolution frequently occurs in practice. For example, new topics frequently appear on Twitter and outdated topics are forgotten with time. Besides, old topics, e.g., topics on festivals, may also become popular again. Such phenomena can also be observed from other types of data streams, such as the click-through data of news or advertisements since the interests of clients may change over time. Class evolution concerns a special case of concept drift and it will briefly review the typical strategies for dealing with concept drift.
Then, it will be preceded by the previous works dedicated to class evolution. A sliding window method stores in memory a number of the most recent examples; the window size can be fixed or variable. The model is updated based on new data, which are stored in the window. Old data, which tend to be affected by concept drift, are forgotten. In the presence of class evolution, although this method can adopt a model to class evolution by dropping previous data, it also forgets potentially useful information of the non-evolved classes, inevitably resulting in a negative impact on the mining performance.

The ensemble method mainly includes chunk-based ensembles, on-line ensembles and hybrid. A chunk based ensemble constructs each base learner by training it with a different chunk of data. A weighted combination of the base learners is applied to handle the concept drift. In the chunk-based ensemble strategy, class evolution would cause the base learners to have different sets of classes. Taking class emergence as an example, this would cause the collective votes of the earlier base learners to outweigh the correct votes for the novel class. On-line ensembles, e.g., on-line bagging and boosting, update each base learner in an on-line manner. This scheme would take a long time for class evolution adaptation. Hybrid ensemble methods aim to combine chunk-based ensembles and online ensembles, to have the advantages of both in a single framework. For example, the recently proposed AUE2 (Almost Universal 2) algorithm employs each chunk of data to initialize a new base learner and to update all existing ones. Then, base learners are weighted according to their accuracies to adapt to the concept drift.

Considering class emergence, since the base learner is mainly trained by the non-evolved class, the novel class is highly imbalanced in the existing base learners. Moreover, the examples from the novel classes are not enough in the early stage of gradual class evolution. Hence, it is still difficult to recognize novel class efficiently when class evolution occurs. Apart from the previous strategies, drift detection
methods explicitly determine the drift of concept and update the model accordingly. To adapt to the new concept, most of these approaches forget any information learned before the detected drift. Similarly to the sliding window strategy, for class evolution, this means that useful information will be forgotten. DDD (Diversity for Dealing with Drift) is a special type of drift detection method that keeps old ensembles while they are useful. However, DDD can only keep old ensembles corresponding to one of the previous concepts.

Therefore, in the case of class evolution, DDD will also forget information when more than one class evolution, behaviour happens over time. Class reoccurrence in class evolution is relevant to recurrent concept drift, which represents the case where a past concept reoccurs again in the data stream. However, the two cases are substantially different. Recurrent concept means a reoccurred joint distribution for all data and thus the whole class set involved in the concept also reoccurs. On the other hand, when class reoccurrence happens, the current concept may not be identical to any previous concept since some other classes might have disappeared. Hence, class reoccurrence may not lead to a recurrent concept and thus might not be handled effectively with existing algorithms for recurrent concept drift.

7.2 Proposed System

In this research, the proposed approach is promising the performance which implicitly assumes that the classes are emerging or disappear transiently. In other words, the Example Generation Rate (EGR, i.e., the number of examples generated per-unit-time) of a class switches between two states, i.e., a constant positive value and zero. However, in a real-world scenario, it is more likely that classes evolve gradually. For example, in an early stage, an event may be discussed by a few participants on Twitter; the topic grows in popularity over a period and then eventually fades away from attention. Motivated by this consideration, this work
investigates the class evolution problem with gradually evolved classes. The gradual evolution of classes refers to the case that classes appear or disappear in a gradual rather than transient manner, i.e., the EGR changes more smoothly. A novel class-based (CB) ensemble approach, namely Dynamic Ensemble Generation and Classification (DEGC), is proposed. In contrast to the above-mentioned existing approaches, which process a data stream in a chunk-by-chunk manner and build a base learner for each chunk, DEGC maintains a base learner for every class that has ever appeared and updates the base learners whenever a new example arrives (i.e., in a one-pass manner).

Classification and adaptive novel class detection of feature-evolving data streams address four such major challenges, namely, infinite length, concept-drift, concept-evolution and feature-evolution. Since a data stream is theoretically infinite in length, it is impractical to store and use all the historical data for training. Concept-drift is a common phenomenon in data streams, which occurs as a result of changes in the underlying concepts. Concept-evolution occurs as a result of new classes evolving in the stream. Feature-evolution is a frequently occurring process in many streams, such as text streams, in which new features (i.e., words or phrases) appear as the stream progresses. Most existing data stream classification techniques address only the first two challenges and ignore the latter two. In this, it proposes an ensemble classification, where each classifier is equipped with a novel class detector, to address concept-drift and concept-evolution. To address feature-evolution, it proposes a feature set homogenization technique.

7.3 Methodology

The impact measure based ensemble classification algorithm has various stages. They are discussed in detail in this section. From the detailed study of the related works, analyzed that the ensemble generation has the number of factors namely concept drift, concept emergence, ensemble length. This research work addresses all
the problems of ensemble generation and for the problem of concept drift, the time window based approach is discussed.

The approach maintains different concept window, where the number of class ensembles is stored. For the problem of emergence, the window adapts the space with modern ensemble class; the length of the window has been a dynamic one to challenge the novel ensembles. Figure 7.1 shows the block diagram of impact measure based on a classification of ensembles.

7.4 Feature Extraction

Feature extraction is an attribute decline process. Unlike feature selection, which ranks the existing attributes according to their predictive importance feature extraction
actually transforms the attributes. The transformed attributes, or features, are linear grouping of the original attributes. The feature extraction process results in a much slighter and richer set of attributes. Models built on extracted features may be of higher quality, because the data is described by fewer, more important attributes. Feature extraction developments a data set with higher dimensionality onto a slighter number of extent As such it is useful for data visualization, since a complex data set can be effectively envisions when it is reduced to two or three dimensions. Some applications of feature extraction are latent semantic analysis, data compression, data decomposition and projection and pattern recognition. Feature extraction can also be used to enhance the speed of data streams and effectiveness of supervised learning. The Extracted features have been used to perform ensemble classification in the next stage. The equation (7.1) and equation (7.2) for the arrival of complete stream data as follows, where $ds$ is data stream and $md$ is metadata and $sf$ is stream feature.

\[ Sd_j = \sum_{i=1}^{size(md)} Md(i) \in Ds \]  
\[ Sf = \sum(Sd \in Sf) \cup Sd_i \]  

These concepts are implemented in section 7.4.1 as an algorithm. The feature extraction algorithm extracts various features from the stream data and such extracted features are converted into stream feature. The generated stream feature has been used to perform ensemble classification. The concept is implemented as an algorithm in section 7.4.1

### 7.4.1 Algorithm for feature extraction

<table>
<thead>
<tr>
<th>Input: Data Stream $Ds$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Stream Feature $Sf$.</td>
</tr>
</tbody>
</table>

Start

Read data stream $Ds$;

Read Meta Data $Md = Ds$.Meta-Data;

For each frame $F_i$ from $Md$;

Extract data from $Ds$;
7.5 Dynamic Ensemble Generation and Classification (DEGC)

The dynamic ensemble classification has been performed based on the time window ensemble generated using the previous trace. First, the method identifies the number of features or chunks present in the feature vector. Then for each feature identified, its data length has been computed. Based on the data length computed, the method computes the Multi-Attribute Ensemble Similarity measure (MAES). The MAES value has been estimated for each ensemble class available in the time window ensemble class. Based on the computed MAES measure, a single class has been identified. For adding stream Data to stream feature The Equation (7.3) used for calculating the multi-attribute ensemble similarity where sf is the size of feature and sf is stream feature, EC<sub>i</sub> represents ensemble class as an output. The Equation (7.4) is used to calculate the cumulative MAE and choose the ensemble class with less MAE value using Equation (7.5).

\[
MAES = \sum_{i=1}^{\text{size}(EC_i)} \sum_{j=1}^{\text{size}(S_f)} \text{Dist}(S_f(j, EC_i(j)), F_j)
\]  

(7.3)

\[
\text{MAES/size}(E_{ci})
\]  

(7.4)

\[
Ec = \text{Min}(\text{MAES.Class})
\]  

(7.5)

The ensemble classification algorithm computes the multi-attribute ensemble similarity with different ensemble classes available and based on the measure estimated a single class had been selected. The data length is computed, the method computes the Multi-Attribute Ensemble Similarity measure (MAES) in equation (7.3) to (7.5). The MAES value has been estimated for each ensemble class available in the time window ensemble class. Based on the computed MAES measure, a single class has been identified. The concepts are implemented in section 7.5.1 as an algorithm.
7.5.1 DEGC Algorithm

Input: Stream Feature $S_f$, Time window Ensemble $Two$.  
Output: Ensemble Class $E_c$.  

Start  
Read stream feature $S_f$;  
Read Time window Ensemble $Two$;  
Compute the size of feature $F_s$ by belongs $sf$;  
For each ensemble class $E_{ci}$  
    For each feature $F_i$ of $S_f$  
        Compute data length $D_l = \text{size}(S_f(F_i))$.  
    End For;  
End For;  
For each feature $E_f$ of class $E_{ci}$  
    Compute Multi Attribute Ensemble Similarity MAES as given in eq(7.3)  
    Compute cumulative MAES as given in eq(7.4)  
End For;  
Choose the ensemble class with less MAES value as given in eq(7.5)  
Stop.

7.5.2 DEGC Min Max Influence Measure estimation

The DEGC minimum and maximum influence measure represents the influence of the ensemble class or feature in any time window ensemble. As the method maintains the ensemble logs of a different time window, the influence of any class can be estimated. To compute the ensemble influence measure, the algorithm first estimates the number of occurrence in receiving the particular stream class. Based on the number of occurrences and the total number of data streams received, the method estimates the influence measure. First, the algorithm classifies the stream as generic and special. Based on the classification results, the method computes the minimum and maximum influence measures. As the algorithm maintains time window ensemble
classes, computing the min-max measures is possible. The equations are given in (7.6), (7.7) and (7.8) used to find the min-max influence measure.

\[
\text{MinIF} = \frac{\sum_{i=1}^{\text{size}(\text{Twe})} \text{Twe}(i) \in \text{Enc}}{\text{size}(\text{Twe})}
\]  
\[ (7.6) \]

\[
\text{MaxIF} = \frac{\sum_{i=1}^{\text{size}(\text{Twe})} \sum_{\text{Twe}(i) \in \text{Enc} \cap \text{Twe}}}{\text{size}(\text{Twe})}
\]  
\[ (7.7) \]

\[
\text{MinIF} = \frac{\sum_{i=1}^{\text{size}(\text{Twe})} \text{Twe}(i) \in \text{Enc} \cup \text{Twe}}{\text{size}(\text{Twe})}
\]  
\[ (7.8) \]

The algorithm described in section 7.5.1 computes the maximum and minimum influence measure for the ensemble feature given or ensemble class given. The computed measures have been used to generate time window ensemble class to support classification. The concept is implemented in algorithm 7.5.3.

### 7.5.3 Algorithm for DEGC Min Max Influence Measure

**Input:** Stream Feature $S_f$ or Ensemble Class $Ec$;  
Time Window Ensemble $Twe$;  
**Output:** Min and Max Influence Measures $\text{MinIF}$ and $\text{MaxIF}$;  
Start  
Read Ensemble Class $Ec$;  
Ensemble Class $Enc$= Perform Ensemble Classification $Ec$;  
If $Enc$.Type==Special then  
Minimum Influence Measure $\text{MinIF}$ by the equation (7.6) &  
Maximum Influence Measure $\text{MaxIF}$ by the equation (7.7);  
Else  
$\text{MinIF} = 0$;  
End if;  
If $Enc$.Type==Generic then  
Minimum Influence Measure $\text{MinIF}$ by the equation (7.6) &  
Maximum Influence Measure $\text{MaxIF}$ by the equation (7.7);  
End if;  
Stop
7.6 Results and Discussion

The proposed dynamic time window based ensemble generation algorithm has been implemented using Apache and its performance in ensemble generation has been validated using the twitter datasets. This method has produced efficient results in all the parameters considered than other methods compared.

![Figure 7.2 Comparison of classification performances](image1.png)

Figure 7.2 Comparison of classification performances

Figure 7.2 presents the result of comparison being performed with varying number of classes. The proposed DEGC algorithm has improved the classification performance up to 97% than other methods in all the number of classes considered.

![Figure 7.3 Comparison on data stream mining performance](image2.png)

Figure 7.3 Comparison on data stream mining performance
Figure 7.3 shows the comparative result on data stream mining performance produced by different approaches. The proposed Min-Max algorithm has produced higher mining performance than others.

![False Classification Ratio](image)

**Figure 7.4 Comparison on false classification ratio**

Figure 7.4 shows the comparative result of false classification ratio produced by different methods. The results show that the proposed approach has produced less false ratio than others.

![Space Utilization Performance](image)

**Figure 7.5 Comparison on space utilization performance**

The figure 7.5 shows the efficiency of the ensemble generation and classification algorithm is depending on how the space available has been utilized. The method has
been evaluated for its efficiency in space utilization. The proposed approach has produced higher utilization performance than other methods compared.

![Time Complexity](image)

**Figure 7.6 Comparison of time complexity**

Figure 7.6 shows the time complexity being introduced by different methods on ensemble generation and classification. The proposed DEGC approach has reduced the time complexity than other methods.

### 7.7 Summary

The above research work was analyzed with the high-speed streaming of data that can be categorized in the form of classification and clustering can be mainly achieved in the performance as well as in improving the data security. So, the algorithm has achieved the high-speed streaming of data preprocessing in order to improve the data mining accuracy. It is also verified under different kinds of data sets experimented. This research work analyses a novel Min-Max influence measure based time window ensemble generation algorithm. The method maintains various time window ensemble classes and handles the ensemble generation problem efficiently. For each ensemble, class identified the min-max measure had been estimated to carry or drop the ensemble class to the future time window. This improves the performance of data stream mining performance and classification. The method reduces the space complexity and time complexity compared to other methods.