Detecting Concept Drift in Classification Over Streaming Graphs

Yibo Yao, Lawrence B. Holder
School of Electrical Engineering and Computer Science
Washington State University, Pullman, WA 99164
{yibo.yao, holder}@wsu.edu

ABSTRACT

Detecting concept drift in data streams has been widely studied in the data mining community. Conventional drift detection methods use classifiers' outputs (e.g., classification accuracy, error rate) as indicators to signal concept changes. As a result, their performance greatly depends on the chosen classifiers. However, there is little work on addressing concept drift in graph-structured data. In this paper, we present a Graph Entropy-based Method (GEM) to effectively detect concept drift in graph streams. Contrary to many related works, we investigate the intrinsic properties of data (i.e., subgraph distribution w.r.t. class membership), instead of monitoring classification outputs. This method can be combined with any graph stream classifier to facilitate classification on non-stationary graph streams. Our approach is combined with several graph stream classification algorithms and tested on synthetic and real-world graph data streams. The experimental results demonstrate the advantage of our method in detecting concept drift as well as improving classification performance.

Keywords

Concept drift; graph classification; graph entropy; streaming graph

1. INTRODUCTION

In recent years, with the emergence of networked data, graph mining has received considerable interest in the data mining community. Many approaches assume that data is generated from a stationary environment. However, in many application domains, graph data is gathered as a stream over time, e.g., network flows, financial transactions, social interaction, etc. This fact raises a problem to the traditional graph learning algorithms due to its non-stationary and dynamic nature. The concept to be learned changes as the underlying distribution that generates the data changes. These changes make the learning model built on the historic data inconsistent with the new data, and it becomes necessary to update the model regularly. Detecting concept drift (or concept change) is one of the core problems in data mining. The increasing volume and velocity of data calls for more efficient and accurate methods for identifying concept change in a data stream. In order to handle concept drift efficiently, a good detection method must detect the change when it occurs, and decide which data to keep and which data to discard [2].

Several effective methods for detecting concept drift over data streams have been proposed during the past years [8]. Most of them assume that the data is received in a stream of instance-label pairs $\{(x_1, y_1), \ldots, (x_t, y_t), \ldots\}$, where $x_t \in \mathbb{R}^n$ is a fixed-size vector and y_t belongs to a finite set $\mathcal{Y}$. However, these traditional approaches become infeasible when dealing with graph streams, because previously-unobserved subgraphs may appear at any time and as a result, the subgraph space may be expanding over time. We introduce a novel Graph Entropy-based Method (GEM) to detect concept drift in graph streams. We apply a sliding window technique on the data stream and compute graph entropy for the subgraphs in the window w.r.t. their class membership. Different from most popular drift detectors that use classification outputs (e.g., error rate) as indicators for detecting concept change, our approach investigates the intrinsic properties of the data (i.e., subgraph distribution w.r.t. class membership). Consequently, a series of entropy values is obtained while sliding the window on the stream by moving one instance forward at each step. We then employ the Cumulative Sum (CUSUM) [9] technique on the entropy sequence to find exactly where a significant shift occurs between successive entropy values and thus to signal concept change. Also, the proposed method is independent of any learning algorithm, so GEM can be incorporated into a graph classifier to make the learning process sensitive to concept change and improve the classification performance.

The aim of this work is to present an effective method to detect concept change for classification on streaming graphs. Identifying the concept changes allows us to discard outdated information and rebuild the learning model with the most relevant information. The specific contributions are:

- A new way to compute graph entropy based on a set of graphs in a sliding window.
- The use of CUSUM as a measure to detect significant deviation between successive entropy values.
- A novel method based on graph entropy to detect concept drift in streaming graphs.

KDD Workshop on Mining and Learning with Graphs (MLG). August 14, 2016. San Francisco, CA.

DOI: 10.475/123_4

The rest of this paper is organized as follows. In Section 2 we give a brief review of related work. Some important definitions are described in Section 3. In Section 4, we propose our graph entropy-based drift detection method. We introduce the experimental setup in Section 5 and empirically evaluate our approach in Section 6. The conclusion and future work are presented in Section 7.

2. RELATED WORK

2.1 Concept Drift Detection

A variety of drift detection schemes have been developed in the past. Drift Detection Method (DDM) [7], Early Drift Detection Method (EDDM) [1] and Adaptive Windowing (ADWIN) [2] are among the most representative and popular detectors. DDM monitors the probability distribution of error rate generated by any online classification algorithm. In periods of no change, the error should decrease since more training examples will reduce the generalization error. It keeps track of the lowest error rate and its standard deviation so far. When the error rate over the current window exceeds the lowest threshold, a concept change is declared and the classification model needs to be relearned. EDDM behaves similar to DDM but uses the average distance between two successive errors and its standard deviation instead. For a stationary data stream, the distance between two consecutive classification errors will increase as the prediction accuracy is improving. If the current distance significantly exceeds the lowest distance, a concept drift is signaled. EDDM has been shown to outperform DDM on a gradual concept drifting data stream [1]. ADWIN uses a variable-size sliding window to deal with distribution change and concept drift in data streams. It maintains a window W with the most recent examples, and checks if two subwindows of W have significantly different averages. If so, a drift is signaled and the subwindow with older instances is discarded. Additionally, the authors provide a theoretical analysis about the bounds of false positive and false negative rates.

Vorburger and Bernstein [13] present an entropy-based measure to monitor concept drift in a reliable and noise-tolerant manner. They utilize the entropy to quantify the distribution inequality between two sliding windows containing older examples and recent examples respectively. If the distributions are equal, the entropy measure results in a value of 1, and if they are absolutely different it results in a value of 0. A similar approach in [5] adopts relative entropy (known as Kullback-Leibler distance) to measure the difference of distributions between two sliding windows on multi-dimensional data streams.

2.2 Graph Entropy

Noble and Cook [10] developed a method to compute substructure entropy based on the occurrences of n-vertex substructures in a given graph. Given an n-vertex substructure, they compute *conditional substructure entropy*, based on all possible extensions to that substructure. This entropy measure has been used to quantify the regularity of graphs with applications to anomaly detection. Graph entropy has been used in [12] to find prominent nodes in graphs and to characterize information flow in an organization. Dehmer and Emmert-Streib [6] present a new definition of graph entropy to measure the structural complexity based on local vertex functionals obtained by calculating j-spheres via Dijkstra's algorithm.

Our method quantifies the information content of subgraphs in a sliding window by calculating a graph entropy measure. It is quite different from DDM, EDDM and ADWIN. But it is similar to the work in [13] which also uses an entropy measure. The authors deal with traditional learning cases in which a fixed-length feature vector is considered to characterize an example in the data stream. However, it becomes infeasible to be applied on graph streams since the subgraph feature space keeps expanding when new graphs continuously stream in.

3. BACKGROUND

3.1 Notation and Definition

Definition 1. A **labeled graph** is denoted by a 4-tuple, i.e., $G = (V, E, \mathcal{L}, l)$, where

1) $V = \{v_1, \ldots, v_{|V|}\}$ is a set of nodes,
2) $E \subseteq V \times V$ is a set of edges,
3) $\mathcal{L}$ is a set of labels,
4) $l : V \cup E \rightarrow \mathcal{L}$ is a function assigning labels to nodes and the edges.

Definition 2. Let $G = (V, E, \mathcal{L}, l)$ and $G' = (V', E', \mathcal{L}', l')$ denote two labeled graphs. G is said to be a **subgraph** of G', i.e., $G \subseteq G'$, if and only if

1) $V \subseteq V'$,
2) $\forall v \in V, l(v) = l'(v)$,
3) $E \subseteq E'$,
4) $\forall (u, v) \in E, l(u, v) = l'(u, v)$.

Definition 3. A **graph stream**, denoted as $\mathcal{G}$, is a sequence containing an infinite number of graphs, i.e.,

$$\mathcal{G} = \{G_1, G_2, \cdots, G_i, \cdots\}$$

Each graph is received in a streaming fashion and has an associated class membership $y_i \in \{\pm 1\}$.

Graph Stream Classification: Given a graph stream $\mathcal{G}$, the goal of graph stream classification is to learn a discriminative model from the graphs seen so far, i.e., $\{G_1, \cdots, G_i\}$, and use the model to predict the class label of the next incoming graph G_{i+1}.

3.2 Concept Drift in Graph Classification

Subgraphs are important components of a network and they are frequently used as features to build learning models to classify graphs [4]. Let $\mathcal{F}(G)$ denote a set of discriminative subgraphs for a graph G. According to the Bayesian Decision Theory, a classification can be described by the prior distribution of the classes $P(y)$ and the likelihood function $P(\mathcal{F}(G)|y)$. The classification decision is then made by the posterior probability:

$$P(y|G) = P(y|\mathcal{F}(G)) = \frac{P(y)P(\mathcal{F}(G)|y)}{P(\mathcal{F}(G))} \quad (1)$$

Formally, **concept drift** between two time stamps t_1 and t_2 can be defined as [8]:

$$\exists t_1 \neq t_2, s.t., P(G_{t_1}, y) \neq P(G_{t_2}, y)$$

where $P(G_{t_1}, y)$ denotes the joint distribution between G and y at time t_1. There are two types of drifts which have been distinguished in the literature. *Real concept drift* is defined as a change in the target concept (i.e., $P(y|G)$), while *virtual concept drift* is defined as a change in data distribution (i.e., $P(G)$) without affecting $P(y|G)$ [8]. In this paper, we mainly focus on real concept drift and refer to it as concept drift. From formula (1), we may state that a concept drift in graph classification is regarded as a change in $P(y|G)$.

4. FRAMEWORK

In this section, we introduce our graph entropy-based drift detection method. The most relevant work is the entropy measure proposed in [13]. However, the authors only deal with traditional data points with fixed-length feature vectors. A graph stream is quite different since the space of subgraphs thought of as features keeps expanding when new instances are streaming in. It is not feasible to predefine a fixed-length feature vector consisting of these occurring subgraphs. On the other hand, discovering $\mathcal{F}(G)$, the set of discriminative subgraphs (e.g., frequent subgraphs, highly-compressing subgraphs), is a time-consuming task, although many efficient graph mining techniques have been developed to do that.

We choose to utilize entropy to characterize the subgraphs' distribution in a comprehensive and unified way. The entropy measure is computed based on all subgraphs in a sliding window over the data stream. We make similar assumptions as [13]:

- If the distribution of subgraphs w.r.t. class labels is not changing, the entropy values will remain stable and no concept drift occurs.
- If the distribution of subgraphs w.r.t. class labels is changing, a significant shift in the entropy values will be observed and concept drift occurs.

4.1 Graph Entropy

Entropy, as a measure of information content in a set, has been widely used in various domains. But there is no commonly used definition to compute graph entropy. We employ a sliding window on a graph stream to compute the conditional entropy of subgraphs inside the window w.r.t. their class membership. Our entropy definition is based on the distribution of subgraphs inside a fixed-size window W. Since the data is received in an infinite streaming fashion, we are not able to derive the exact probability distribution. Instead, we estimate each subgraph's probability by calculating the proportion of its instances inside a large enough window. Let W denote a window containing several graphs, $W = \{G_1, \cdots, G_{|W|}\}$, and $S = \{s_1, \cdots, s_n\}$ denote a set of subgraphs discovered from the graphs of W, the probability of each subgraph s_i can be estimated using:

$$P(s_i) = \frac{r_i}{\sum_{j=1}^{n} r_j}$$

where r_i is the number of occurrences of s_i in W. In a supervised learning scenario, each graph G_i has a class membership $y_i \in \{\pm 1\}$. Although we only consider 2-class learning problems, all definitions can be extended to multiple-class cases. In order to incorporate the class information into the computation of graph entropy, we need to obtain the conditional probability distribution of s_i. Since each s_i in the window has two associated values, namely r_{i+} and r_{i-}, denoting its number of occurrences respectively in positive and negative examples, the conditional probability of s_i can then be estimated using:

$$P(s_i|+) = \frac{r_{i+}}{\sum_{j=1}^{n} r_{j+}}, \qquad P(s_i|-) = \frac{r_{i-}}{\sum_{j=1}^{n} r_{j-}}$$

Thus, the entropy of the subgraphs w.r.t. their class labels in W can now be defined as:

$$e(W) = -\sum_{y \in \{\pm 1\}} P(y) \sum_{i=1}^{n} P(s_i|y) \log_2 P(s_i|y) \tag{2}$$

where $P(y = +1)$ is the fraction of positive graphs in W, and $P(y = -1)$ is the fraction of negative graphs in W. Given graph stream $\mathcal{G} = \{G_1, \cdots, G_i, \cdots\}$ a sequence of entropy values $\mathcal{E} = \{e_1, \cdots, e_i, \cdots\}$ is generated by moving the sliding window W one step forward over the stream. Here, e_i is the resulting entropy for the ith window which is based on the subset $\{G_i, \cdots, G_{i+|W|-1}\}$ of graphs from $\mathcal{G}$.

4.2 Concept Drift Detector

The series of entropy values obtained over a graph stream indicates the subgraphs' distribution w.r.t. the class labels in each window. In a supervised learning setting, the amount of uncertainty indicates the amount of new information introduced when the learning process proceeds. As a result, a large magnitude of deviation between entropy values implies a concept drift. In other words, if the concept is stable over time, the amount of uncertainty in the window remains at a relatively low level and no significant shift occurs over the entropy series. If concept drift occurs, the magnitude of uncertainty starts to increase and results in a significant increase in the entropy. To this end, we adopt Cumulative Sum (CUSUM) on the entropy series to effectively detect shifts of these entropy values.

CUSUM was first introduced by Page [11] and has been used as a quality control technique for detecting change over a sequence of data. A basic assumption for using CUSUM is that the data complies with a normal distribution. Let $\mathcal{D} = \{D_1, \cdots, D_i, \cdots\}$ denote the consecutive difference of the entropy values, that is, $D_i = |e_i - e_{i-1}|$. We assume that $\mathcal{D}$ is drawn from a normal distribution with mean μ and standard deviation σ. Particularly, μ is the target value that needs to be controlled in the process. If there is no concept drift, the entropy values remain stable and thus μ ideally equals 0. We use a quantity C_i to record the cumulative sum of the distance between each D_i and μ:

$$C_i = max[0, D_i - (\mu + K_i) + C_{i-1}] \tag{3}$$

in which the starting value is $C_0 = 0$. K_i is usually called a *slack value* which allows a small deviation from the target μ. It is usually recommended to set $K_i = k\sigma_i$, where σ_i is the standard deviation which can be estimated from the subset $\{D_1, \cdots, D_i\}$. When concept drift occurs, the increased uncertainty will result in a jump in the value of C_i. If C_i

shifts upward to exceed a certain threshold $H = h\sigma_i$ (H is called the *maximum allowed deviation*), a change in the distribution needs to be signaled. Using $k = 0.5$ and $h = 5$ generally enables CUSUM to tolerate a shift of about $1\sigma_i$ from the target mean [9]. The point at which $C_i > H$ indicates that the process starts to be unstable at that point, but it does not necessarily mean that concept drift occurs at the same point. Therefore, another quantity N_i has been introduced to indicate the possible position where concept drift really happens.

$$N_i = \begin{cases} N_{i-1} + 1 & \text{if } C_i \neq 0 \\ 0 & \text{if } C_i = 0 \end{cases}$$

N_i is used to record the number of consecutive periods that C_i has been nonzero. If $C_i > H$ and thus a change is signaled, the concept drift likely occurs at position $i - N_i$ [9]. Then the classification model needs to be reconstructed based on the examples between $i - N_i$ (known as *warning point*) and i (known as *drift point*). Algorithm **1** lists the detailed procedures of the proposed drift detection method.

Algorithm 1. Graph Entropy-based Method (GEM)

Input:
$\mathcal{G} = \{G_1, \cdots, G_i, \cdots\}$: a graph stream
$|W|$: sliding window size

Output:
DRFT: a list containing detected drift points
WARN: a list containing detected warning points

initialize $\mathcal{E} = \{0\}, \mathcal{D} = \{0\}, \mathcal{C} = \{0\}, \mathcal{N} = \{0\}$
set $i = 1, start = 1, h = 5, \mu = 0$
while not at end of stream
  $W = \{G_i, \cdots, G_{i+|W|-1}\}$
  calculate $e_i = e(W)$ using equation (2)
  $\mathcal{E} = \mathcal{E} \cup \{e_i\}$
  $D_i = |e_i - e_{i-1}|$
  $\mathcal{D} = \mathcal{D} \cup \{D_i\}$
  σ_i = standard deviation of $\{\mathcal{D}_{start}, \cdots, \mathcal{D}_i\}$
  calculate C_i using equation (3)
  if $C_i > 0$
    $N_i = N_{i-1} + 1$
  else
    $N_i = 0$
  end if
  if $C_i \geq h\sigma_i$ %% a drift is signaled
    $drift = i + |W|$
    $warn = drift - N_i$
    append $drift$ to DRFT, $warn$ to WARN
    $C_i = 0$, $N_i = 0$, $start = i$
  end if
  $\mathcal{C} = \mathcal{C} \cup \{C_i\}$
  $\mathcal{N} = \mathcal{N} \cup \{N_i\}$
  $i = i + 1$
end while

4.3 Analysis

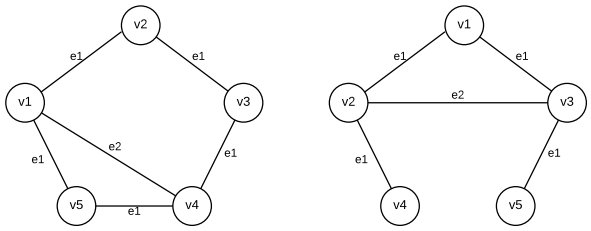
The entropy computation of our method is based on the discovered subgraphs from the graph instances in a window. However, discovering all subgraphs from a graph is impractical. We can specify a certain type of subgraph for computing the entropy instead of finding all subgraphs from a graph. Some recommended subgraph types are: node, edge, clique, and n-length shortest path. In our experiments, we choose to investigate the probability distribution of edges in a graph instead of discovering the whole set $\mathcal{F}(G)$, since edges are basic components of a graph and play significant roles in connecting entities. Each edge is characterized by the two endpoints' labels and its edge label. If the edge is directed, the direction needs to be considered in the characterization. Compared with using $\mathcal{F}(G)$, using edges significantly reduces the computational cost of entropy calculations. Let W denote a window containing $|W|$ graphs, $W = \{G_i, \cdots, G_{i+|W|-1}\}$, then the computational complexity of each window is $O(\sum_{t=i}^{t=i+|W|-1} |E_t|)$, where $|E_t|$ is the number of edges in G_t.

5. EXPERIMENTAL SETTING

There is little study on designing strategies to find concept drift in graph data, especially for graph streams. Therefore, no benchmark datasets in the literature are available to evaluate different drift detection methods on graph streams. We generate several synthetic graph datasets and impose different types of concept drift on them. We also choose one real-world graph dataset which has been used extensively in the literature for testing graph stream classification techniques. Concept drifts may manifest in different forms [8], but we have only considered two different patterns in our experiments: abrupt and gradual.

5.1 Synthetic Data

SUBGEN[1] is an artificial graph data generator. The input for SUBGEN is a user-defined subgraph pattern, and the output is a connected graph generated by starting with the input patterns and then randomly adding nodes and edges. We design four simple substructures as instances for generating synthetic data consisting of undirected, labeled graphs. The numbers of nodes and edges in each graph have been set to 20 and 100 respectively. In Figure 1, the two



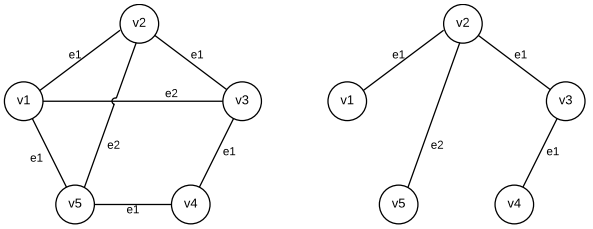
(a) positive instance (b) negative instance

Figure 1: Instances used to generate $S1$.

instances have been used to generate the first dataset $S1$. It contains 2000 graphs, in which half of them are generated by embedding instances of (a) and another half generated by embedding instances of (b). The graphs produced using instances (a) are labeled as positive while the graphs produced using instance (b) are labeled as negative. Another synthetic dataset $S2$ is generated by embedding the two instances from Figure 2 in a similar way.

We then artificially generate two datasets with different types of concept drift incorporated:

[1]http://ailab.wsu.edu/subdue/



(a) positive instance (b) negative instance

Figure 2: Instances used to generate $S2$.

- SYN1. We introduce an abrupt concept drift into $S1$ by reversing the class membership of the second half examples. Ideally, the concept drift will occur at the middle position (i.e., the $1001st$ example).
- SYN2. $S1$ and $S2$ are concatenated together. The ideal drift position will be the concatenation point (i.e., the $2001st$ example).
- SYNMIX. We merge the ending part of $S1$ and the beginning part of $S2$ with mixing percentage 10%, 30% and 50%. For instance, 10% from the $1701-1800$ graphs in $S1$ are switched with 10% from the $201-300$ graphs in $S2$; 30% from the $1801-1900$ graphs in $S1$ are switched with 30% from the $101-200$ graphs in $S2$; 50% from the $1901-2000$ graphs in $S1$ are switched with 50% from the $1-100$ graphs in $S2$. Then $S1$ and $S2$ are concatenated together to create a data stream with gradual concept drifts.

5.2 Real-World Data

In addition to the artificial data mentioned above, we also conduct experiments on one real-world data source. The Internet Movie Database (IMDb, www.imdb.com) is an online source containing movies and the associated information, such as actors, directors, etc. We collect a sample of IMDb data consisting of 7,855 movies released in the United States from 1990 to 2012, and construct a graph for each movie. In a graph, each node denotes a movie's ID, an actor's ID or a director's ID, and every edge denotes the *acted-by* relationship between the movie and an actor or the *directed-by* relationship between the movie and a director. Additionally, we include the older movies acted by the same actors or directed by the same directors into the graph to form more connections. An example of IMDb graph data is shown in Figure 3.

A graph stream is then created by feeding the movie graph objects chronologically according to their release dates in the United States. Three classification tasks are involved in our experiments with artificially imposed concept changes:

- Box-Office. We define a movie as successful if its opening weekend revenue is no less than $2 million. Our task is to predict whether a newly released movie will be successful or not. In the first half of the movie stream, a successful movie is considered as a positive example while an unsuccessful movie as a negative example. But in the second half of the movie stream, we reverse the class membership of each movie.

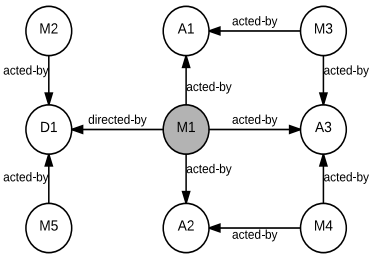


Figure 3: Graph representation for a movie (M1) in IMDb. M1 is acted by actors A1, A2 and A3, and directed by director D1. M2 and M5 are movies directed by the same director but released earlier than M1. Similarly, M3 and M4 are older movies acted by the same actors.

- Genre. *Comedy* and *Thriller*$\backslash$*Crime* are two top genres in the IMDb dataset. There are 2,934 *Comedy* movies and 2,224 *Thriller*$\backslash$*Crime* movies (movies with both labels are removed). In the first half of the stream, our task is to classify a newly released movie into two categories: *Thriller*$\backslash$*Crime* or non-*Thriller*$\backslash$*Crime*; in the second half of the stream, our task is to classify a newly released movie into another two categories: *Comedy* or non-*Comedy*.
- REALMIX. We extract the first half of Box-Office data and the second half of Genre data together, and then mix them in a similar way as SYNMIX to form gradual concept drifts.

5.3 Classification Algorithms

In order to evaluate the proposed drift detector, we compare it with two drift detection methods, namely DDM [7] and EDDM [1]. It is not the aim of the present work to compare the performance of different classification algorithms. Our goal is to demonstrate the advantage of our entropy-based approach in error reduction when it is incorporated into any online graph stream classifier. We choose two classifiers: INCremental SVM learner (INCSVM) [14] which uses an incremental SVM method to classify nodes or subgraphs in a dynamic network, and DIscriminative Clique Hashing (DICH) [3] technique which decomposes a compressed graph into a number of cliques and extracts clique-patterns over the stream as features. We incorporate the drift detectors into these two algorithms' learning progress. We vary the window size in our experiments, i.e., $|W| = \{300, 400, 500\}$.

5.4 Evaluation Metrics

The overall classification error rate has been used to evaluate the classifier's prediction performance with the different concept drift detection methods. It is defined as the total number of incorrectly classified examples divided by the total number of examples in a stream. For abrupt concept drift, we can identify clearly where the drift occurs. Therefore, the detected drift positions will be another important evaluation criterion. We use *delay* to denote the distance between detected positions and correct positions. It represents how quick a system can react to a concept change. The lower a delay is, the quicker the detector reacts. On the other hand, for gradual concept change, it is hard to iden-

tify the exact drift position. In this case, the best evaluation metric will be the classification error rate.

6. EXPERIMENTAL RESULTS

6.1 Abrupt Drift

6.1.1 *Evaluation on Synthetic Data*

Table 1 presents the values of delay for the two synthetic datasets using different window sizes. We can find that all the drift detectors succeed in detecting the concept change with various delays. But the delays generated by GEM are much lower than those generated by DDM and EDDM.

Table 1: Values of *Delay* on SYN1 and SYN2

	SYN1		SYN2	
Methods	INCSVM	DICH	INCSVM	DICH
DDM	28	395	122	423
EDDM	50	113	395	483
GEM(300)	5	5	**5**	**5**
GEM(400)	4	4	9	9
GEM(500)	**3**	**3**	16	16

Figure 4 shows the overall classification error rates for the two classification algorithms with the three concept drift detectors incorporated. We observe that both algorithms are able to achieve better classification results (i.e., lower error rate) when combined with the proposed GEM approach. Note that the lowest delay in detecting a concept drift does not guarantee the lowest error rate. Because when a concept drift is signaled, the learning model is rebuilt upon the training graphs between a warn point and a drift point.

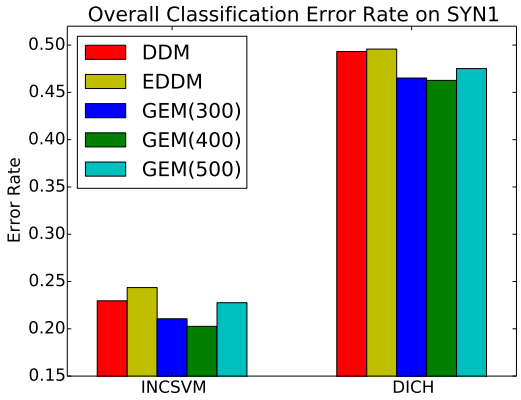
6.1.2 *Evaluation on Real-World Data*

Similar to what was done for evaluating the drift detectors on the synthetic datasets, we conduct experiments on the real-world data. The experimental results are shown in Table 2 and Figure 5, respectively.

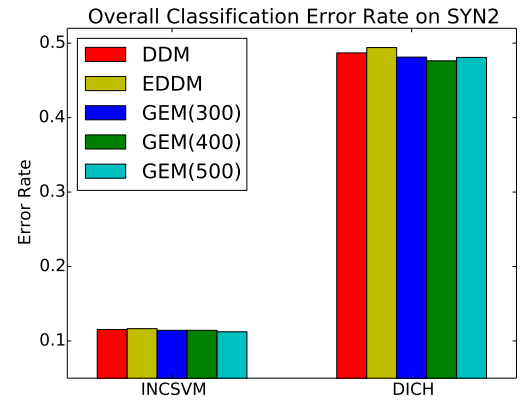
Table 2: Values of *Delay* on IMDb: Box-Office and Genre

	Box-Office		Genre	
Methods	INCSVM	DICH	INCSVM	DICH
DDM	164	445	279	784
EDDM	670	812	864	1001
GEM(300)	N/A	N/A	N/A	N/A
GEM(400)	N/A	N/A	N/A	N/A
GEM(500)	**100**	**100**	**100**	**100**

In Table 2, we find that GEM failed to signal any drift positions when the window sizes are 300 and 400, but succeeded in generating the minimum delays for the two prediction tasks if the window size is set to 500. Therefore, a proper window size is needed in order to effectively detect the drifts and obtain the minimum delays. Since GEM was not able to detect any drift when $|W| = 300$ or $|W| = 400$, the classification error rates of GEM(300) and GEM(400) are actually the error rates using the corresponding classifier without incorporating any drift detector (see Figure 5). However, when a proper window size is chosen (i.e., $|W| = 500$), the two



(a) SYN1



(b) SYN2

Figure 4: Overall classification error rate on the synthetic datasets for different concept drift detectors combined with different classification algorithms.

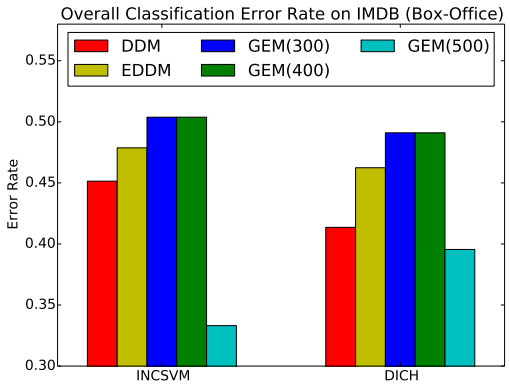
classification algorithms with GEM can achieve their lowest error rates, respectively.

6.2 Gradual Drift

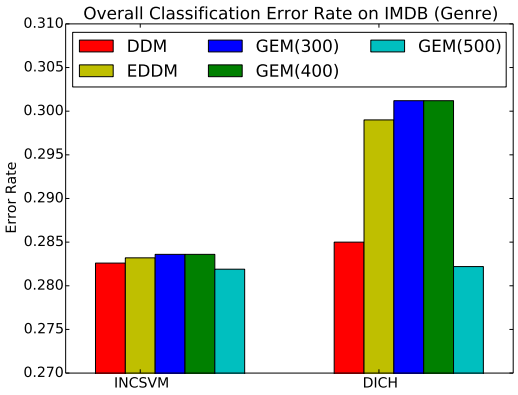
We investigate the performance in detecting concept drifts from a gradual changing environment. Because of the nature of the two mixed datasets, we know that concept drifts will occur somewhere around the concatenation position, but we have no idea where exactly those drifts are located. Therefore, we do not calculate the values of delay for each algorithm. The only metric for this situation is the classification error rate. We record the error rates obtained from each algorithm in Table 3. The results for GEM were obtained by setting $|W| = 500$.

Table 3: Overall classification error rate

	SYNMIX		REALMIX	
Methods	INCSVM	DICH	INCSVM	DICH
DDM	0.190	0.480	0.372	0.360
EDDM	0.202	0.488	0.344	**0.322**
GEM(500)	**0.182**	**0.472**	**0.328**	0.330



(a) Box-Office



(b) Genre

Figure 5: Overall classification error rate on IMDb (Box-Office and Genre) for different concept drift detectors combined with different classification algorithms.

Overall, these experimental results validate that the proposed drift detection method can achieve improved classification performance with proper choice of its parameters. Unlike DDM and EDDM, GEM characterizes the intrinsic properties of the graph data and detects concept change by investigating the statistical information of the subgraphs instead of relying on the classification algorithms' outputs. The results also demonstrate that GEM enables the stream classifiers to react more quickly compared to other drift detectors when a concept change occurs, and thus reduces classification errors.

7. CONCLUSION

We present a novel framework for studying the problem of detecting concept drift on graph streams. Our detection method is based on graph entropy measured inside a sliding window. We compute a series of entropy values by moving the window on a data stream. CUSUM is utilized in our method to signal concept change by detecting significant shift between successive entropy values. The experimental results obtained from several datasets show that the proposed approach outperforms the competing methods, which fully demonstrate the effectiveness of our method in detecting concept drift and reducing classification errors.

Our future work will include exploring the pros and cons of the proposed method by conducting comparisons with more state-of-the-art drift detectors. We will also explore an intelligent way to set the size of the sliding window adaptively in order to reduce memory usage. Furthermore, we will develop an elegant way to estimate the probabilities of subgraphs used in the entropy computation.

8. ACKNOWLEDGMENTS

This material is based up on work supported by the National Science Foundation under Grant No. 1318913.

9. REFERENCES

[1] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno. Early drift detection method. In *ECML PKDD Workshop on Knowledge Discovery from Data Streams*, 2006.

[2] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining*, 2007.

[3] L. Chi, B. Li, and X. Zhu. Fast graph stream classification using discriminative clique hashing. In *Advances in Knowledge Discovery and Data Mining*, pages 225–236. Springer, 2013.

[4] D. J. Cook and L. B. Holder. *Mining graph data*. John Wiley & Sons, 2006.

[5] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Symposium on the Interface of Statistics, Computing Science, and Applications*, 2006.

[6] M. Dehmer and F. Emmert-Streib. Structural information content of networks: Graph entropy based on local vertex functionals. *Computational Biology and Chemistry*, 32(2):131–138, 2008.

[7] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence–SBIA*, pages 286–295. Springer, 2004.

[8] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4), 2014.

[9] D. C. Montgomery. *Introduction to statistical quality control*. John Wiley & Sons, 2007.

[10] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 631–636, 2003.

[11] E. S. Page. Continuous inspection schemes. *Biometrika*, pages 100–115, 1954.

[12] J. Shetty and J. Adibi. Discovering important nodes through graph entropy: The case of Enron email database. In *Proceedings of the 3rd International Workshop on Link discovery*, pages 74–81. ACM, 2005.

[13] P. Vorburger and A. Bernstein. Entropy-based concept shift detection. In *IEEE International Conference on Data Mining*, pages 1113–1118, 2006.

[14] Y. Yao and L. Holder. Scalable SVM-based classification in dynamic graphs. In *IEEE International Conference on Data Mining*, pages 650–659, 2014.