International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-7, Issue-6, March 2019

A Hybrid Error-Driven Approach to Data Stream Classification

G. Abinaya, Aditya Subramanian, Harsh Kumar, Sanjeev Rao, Sourav Patra

Abstract*: Challenges in the field of data stream mining include the vast volume of data being mined, the speed at which data arrives, and the presence of concept drifts. Traditionally, data classification has always involved the assumption of prior knowledge of the data sets, a method which is not particularly suitable when dealing with high-speed data streams. As such, various methods have been developed for the specific use-case of stream data mining, which are able to handle concept drifts during the data mining process with varying degrees of accuracy. Here, a probabilistic queuing model - based on an existing 'SyncStream' algorithm - is used in order to passively detect and account for the presence of abrupt concept drifts. In addition, other aspects of the system are tuned for better classification accuracy and throughput.

***Index Terms*: classification, data streams, data mining, queuing theory, learning.**

I. INTRODUCTION

A relatively recent area of interest in data mining is that of mining data streams; it differs from traditional data mining insofar as existing means of data mining suffer due to the underlying differences between static data sets and data streams: data streams can be thought of as an effectively infinite source of incoming data with occasional concept drifts — that is, differences in 'meaning' in the data — whereas static data sets are of finite size, with sources of error usually being fixed in nature — that is, multiple facets of the data set are known in advance; concept drifts are absent. As such, it does not lend itself well to classification due to the presence of both abrupt and slow (implicit) concept drifts; to handle this, newer approaches have been used to deal specifically with the problem of mining data streams, with high (but varying) levels of success, with a tradeoff between classification accuracy and speed being required in most cases [1] [2]

In this paper, a probabilistic algorithm inspired by past approaches is proposed, which is based on a queuing model for evaluating the presence of concept drifts. An error-driven learning approach is also made use of for classifying data into representative prototypes in a manner similar to [2], with a primary difference being the usage of probabilistic measures to reduce the effective rate of concept drift detection-avoidance. A new algorithm *HED-CDD* is proposed which makes use of a *C-HEAP* to aid the classification process. The layout of the paper is as follows: section II introduces a brief background required for the paper; section III examines some of the related works currently in the literature; section IV describes the proposed system, both as an overview and in detail; section V provides an analysis of certain features of interest in the system, and section VI concludes the paper.

Revised Manuscript Received on 30 March 2019.
* Correspondence Author
Mrs. G. Abinaya, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India.
Aditya Subramanian, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India.
Harsh Kumar, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India.
Sanjeev Rao, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India.
Sourav Patra, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India.

II. BACKGROUND

A brief description about the topics necessary as a background for this paper will be presented as follows. Specifically, two main topics are briefed - queuing theory, and data streams.

A. Queuing Theory

Queuing theory deals with the various types of queues and their corresponding characteristics; most queues can be described by their arrival, discipline and service pattern: the arrival pattern dictates how items are added to the queue and how they fit into the queuing system, the service pattern describes the way the items are serviced (e.g. with a single server, multiple servers or self-service, either in series or in parallel) and finally the queue discipline describes the type of the queue (e.g. FIFO, LIFO or random/SIRO disciplines) [3]. Regardless of the type or characteristics of the queue, it is nevertheless possible to represent all queues using Kendall notation [4]; for the purposes of this paper, only infinite-source queues with a single server and the FIFO discipline will be considered, which are represented in Kendall notation by $(M/M/1) : (\infty/FIFO)$, and will be briefly discussed below.

Queues of type $(M/M/1):(\infty/FIFO)$: Here, the queue is assumed to be of infinite size, with the service following a continuous-time Markov chain in time t ; assuming steady-state characteristics in this system, P_n can be derived as [3]:

$$Pn = P0 * \left(\frac{\lambda_0\lambda_1\lambda_2 \dots \lambda_{n-1}}{\mu_0\mu_1\mu_2 \dots \mu_n}\right), n = 1,2, \dots \quad (1)$$

Where

$$P_0 = 1 - \frac{\lambda}{\mu}$$

The average number of items present in the queue is given by:

$$L_q = \frac{\lambda^2}{\mu(\mu - \lambda)} \quad (2)$$

And finally, the waiting time in the system follows an exponential distribution with parameter $(\mu - \lambda)$ and so the average time spent waiting in the system is:

$$\frac{1}{\mu - \lambda} \quad (3)$$

Where λ is the mean number of arrivals in unit time and μ is the inter-service mean [3].

B. Data Streams

In the context of data mining, data streams can be regarded as a practically infinite and high-speed source of data; mining these streams involves "extracting knowledge structures from continuous, rapid data records" [5]. It stands that storing such data sets wholesale is not possible; learning and prediction thus has to occur on a continuous basis instead. Given some underlying knowledge about the data stream in the beginning (e.g. *spam/text*) in the context of representative examples, the aim then, is to classify future, "real" data from the stream under similar or dissimilar attributes. Changes in the data stream concerning rules or instances over time can impede efforts, in a process referred to as *concept drift* [5]. An alternative means of viewing a data stream — and one that is held in this paper — is to consider it as an infinitely long queue of data items waiting to be classified, where a group of classifiers is responsible for classifying and extracting labels from them (which approximates a server in the context of queuing theory.) This property is exploited to make a probabilistic model for determining when abrupt concept drifts may occur, borrowing concepts from networks of waiting lines [6] in addition to the general method.

Multiple types of concept drift can occur in a data stream, with up to 5 different types of drift, and their combinations [7], which can pose different types of challenges. Noise in the data stream can reduce accuracy, if excessively aggressive methods treat them as a drift instead of errors in the source; incremental drifts may reduce accuracy up to the point where the classification methods recognize the presence of a drift (e.g. by reducing accuracy below a threshold); sudden or abrupt drifts require checks to confirm their presence (as opposed to e.g. being just noise in the source); gradual drifts can be confused with noise in lenient methods, and a combination of these exist in real data streams, with the additional constraint of not being slow to classify the stream.

III. RELATED WORK

Means of detecting concept drift can fall into either of two broad classes of methods, that is, active and passive approaches [8]. Originally, single-model classifiers were used on the data stream being classified [2]; more recent methods currently being employed involve making use of ensemble and weighted ensemble models [9], which use multiple models so as to achieve better accuracy than single-model methods; some other ways involve analyzing the data stream classifiers to determine the presence of concept drift. This involves determining the accuracy of classifiers, and making a decision as to whether concept drift has occurred on the basis of the recent accuracy history; a consistent decrease in accuracy is typically indicative of concept drift [10] [11]; this is similar to, and sometimes used with, the sliding window approach for classifiers, with drift detection being tackled by the windows. This has its problems, namely, deciding the size of the window either in advance, or during runtime [2]; an adaptive version [12], ADaptive WINdowing changes the size of the window dynamically [13], thereby solving the problem of having very small window sizes (which may lead to classifiers not learning concepts properly), or very large window sizes (which may lead to invalid concepts being learnt) [1], before mining begins; however, it still achieves a tradeoff at best, viz. window sizes still incur problems like delays and optimal sizes. Hoeffding Trees are another widely used method [14], which are decision trees using Hoeffding bounds; strong bounds on performance can be achieved as a result.

Another common method of handling concept drifts as mentioned earlier is that of using ensemble classification models, which involves essentially using a set of classifiers on the data stream; by replacing the classifiers which perform the worst on the current data set, with ones that are better suited to the task (based on earlier trained data), classification accuracy can be enhanced [9]. Two methods for ensemble learning on online (i.e. streaming) data were proposed in [12] in the form of ADWIN bagging and Adaptive-Size Hoeffding Tree bagging, and have been widely made available in practice [15]; such classification methods perform well even in outlier detection, which has been adapted by other systems; however, problems with sliding windows still exist in some, like in the case of [16].

A variant on the supervised approach — *catastrophic forgetting* — uses deep neural networks for classifying data streams, wherein the deep neural network used is increasingly trained with data that is not independent and identically distributed (i.i.d.); by using *memory efficient rehearsal*, examples generated then fine-tune the neural network, serving as an approximate upper bound [17]. Some approaches, such as windowing methods, opt to retrain the classifiers on the new data after the detection of a concept drift having occurred; others, however, choose to update only those parts of the model which have been directly affected by the concept drift [2]; typically, clustering algorithms are used in tandem with these approaches, both for minimizing the space required (compression) for storing concepts, as well as in creating new concepts when an abrupt concept drift occurs [1]; recently, synchronization-based clustering has gathered much attention in the literature, especially those based on the Kuramoto model [18] [19] [20]; by varying the clustering model used in the system, it is possible to achieve better performance owing to reduced space demands, as well as better inter- and intra-classification of concepts [2] — an example being one using a double-layer oscillatory network for analysing clusters [20]; another one uses a linearized version of the Vicsek physical-based clustering model [19] for handling general data sets as an improvement over the

Sync clustering algorithm used for concepts in [1].
As far as probabilistic means of processing data streams are concerned, multiple means are used to handle data streams probabilistically; in one such paper, a concept-drift detection system is described which processes data in chunks to arrive at a drift indication, by computing differences in successive chunks [21]; another involves using probabilistic classifiers such as the Naïve Bayes classifier, using Bayes' Theorem on assumed independence between features [22]; others use Bayesian models to determine concept drift using the notion of drift and change-point detection, using predictions of trained models, and removing drift points probabilistically in a system known as *Bayesian Conditional Model* [23]. The Hoeffding bounds calculated for the construction of the Hoeffding Tree uses a probabilistic measure for calculation, too [14].

On the other hand, very few such procedures were found which directly corresponded to queuing measures of handling data streams; this may be due to the relatively strict assumptions and conditions imposed by the classical (i.e. Jacksonian [6]) queues, which assume a Poisson arrival rate and exponential service time distribution [3]. This means that it is not directly applicable to many areas outside direct queues; however, research in other fields has revealed that even with a strict implied model and assumptions, an accurate result can be achieved insofar as the queue model is concerned – one such paper used performance measures to evaluate the queuing model, and it was found that, even with relaxed assumptions (since the system being measured was not a traditional queue, nor did it follow any of the queuing models' assumptions strictly) a high correlation was achieved, with outliers being explained due to a few factors not considered in the simplified model [24]. Queuing theory was used to handle the case of mining multimedia data streams in [25]; a $G/G/1$ queue was used — an arbitrary arrival and service process - for a boosted classifier architecture; however, reference to concept drift in the resulting data stream was sparse. It was noted that the arrival process could be modeled as a Poisson process with an exponential service used — similar to the $M/M/1$ model.

IV. PROPOSED METHOD

This section will describe the method by which *HED-CDD* works in tandem with *C-HEAP* to probabilistically determine concept drift occurrences; before this, however, a broad overview will be provided which aims to explain the intent and reasoning for the algorithm.

A. Overview

As mentioned earlier in subsection II-B, a data stream can be visualized as a queue of infinite length, carrying data of the specified overarching type (subject to the data stream source), with a set of classifiers in a akin to the original Bagging algorithm [13] and the newer adaptive-window method used in OzaBagADWIN [12]; here, an initial drift measure is estimated based on the source of the data (viz. the type of data stream), which roughly corresponds to the measure at which quick, abrupt concept drifts are estimated to occur.As an aside, it does not make a significant difference *what* the initial value of the drift measure is, as the value will be auto-adjusted over time, although a significant performance hit can occur if extreme values of the measure are kept in the beginning, which indicates a low confidence in the data stream; even if it asymptotically adjusts to long-term, reflected values, the classifiers used will take a longer time than usual in the interim period. Since major sources of concept drift (i.e. abrupt drifts) are relatively insignificant with respect to the long-term implicit concept drifts, it can be assumed that low values of drift measure hold.

This concept drift measure is then used by *HED-CDD*, which uses a system-wide "drift measure" timer, kept for every T_s seconds which performs a check every time the timer runs out; if an abrupt concept drift has occurred during this time period, the new duration is readjusted to a new T_s, which is given by the mean time dictated by the queuing model; since it is assumed that abrupt drifts are rare vis-à-vis implicit concept drifts and anomalies, the model holds; nevertheless, the drift measure is adjusted to reflect the new confidence in the data stream, as it cautiously assumes further (future) abrupt drifts in the data stream. It is important to note, however, that the queuing model is based on the assumption that abrupt concept drifts are rare, and so can be estimated by a continuous/memoryless [3] Markov chain; if the data stream has a high percentage of concept drifts (reflecting sources of data that are not consistent, or those with low confidence in drift measures), a different queuing model will have to be used, such as a non-Markovian $G/G/1$ model.

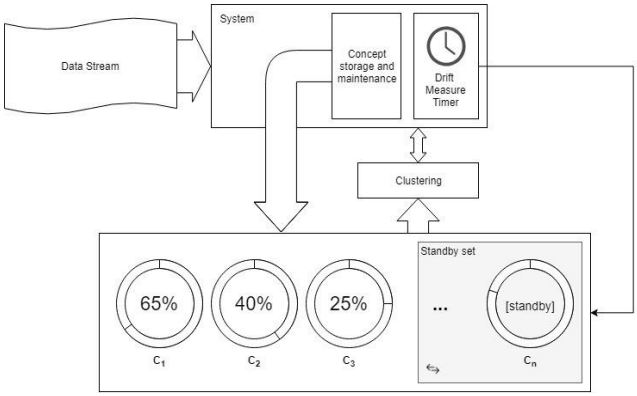

Fig. 1 Architecture of the system

B. Clustered Data Representation

As has been discussed before in prior literature, historical data representation cannot be stored in the case of a data stream, as doing so requires vast amounts of memory, and with it, power to process it (as a function of the effective history size) [2]; as such, reducing the number of stored data points becomes a necessity for efficient operation, which has been achieved by clustering incoming data from the stream over time, with only the resultant data points being stored over time, as representative concepts [1]. We use a similar method, based on improvements made to the synchronization-based clustering algorithm (SynC), as in the Effective SynC (ESynC) algorithm using a linear version [19] of the original Vicsek model by [26]; unlike the original SynC algorithm which modeled data points according to a phase oscillator in d-dimensional space (as per the Kuramoto model) [18],

the Vicsek model assumes each example in d-dimensional Euclidean space as agents flocking towards nearby agents over δ time steps; a few definitions are given below, which are used by the linear Vicsek model by [19]:

1) Definition 1

The δ near neighbor point set $\delta(P)$ of point P is defined as:

$$\delta(P) = \{X | 0 < dis(X,P) \leq \delta, X \in S, X \neq P\} \quad (4)$$

Where $dis(X,P)$ is the measure of dissimilarity between point X and P in set S, with a predefined threshold parameter δ [27].

2) Definition 2

The linear Vicsek model as given by [19] is an adjustment to the original Vicsek model; where X is a point $(x_1, x_2, \ldots, x_d)$ in d-dimensional Euclidean space, the renewal characteristics (which describe the way point X behaves over time) are given by [19]:

$$X(t+1) = X(t) + \frac{1}{1 + |\delta(X(t))|} * \sum_{Y \in \delta(X(t))} (Y - X(t)) \quad (5)$$

3) Constraints

Like in [1], adapting the model to data classification requires the use of an additional constraint so as to prevent different class labels from intermixing; this was achieved by the use of 'cannot-link' constraint, used to prevent data with different labels from being classified under the same group [1]; a similar method is used for the linear Vicsek model as in [19]:

$$X(t+1) = X(t) + \frac{1}{1 + |\delta(X(t))|} \sum_{Y \in \delta(X(t)), X_t = Y_t} (Y - X(t)) \quad (6)$$

Where X_t and Y_t are the class labels of X and Y respectively, thereby ignoring dissimilar pairs.

C. Concept Drift Detection

Detecting concept drift has been tackled in multiple ways - some ways include statistical approaches, with tests like the Kolmogorov-Smirnov test, Wilcoxon Rank Sum and 2-sample t-test being used [24]; in effect, many of the drift detection approaches can be summarized as supervised and unsupervised methods, with the former relying on known class labels being used to check for the presence of concept drift as compared with the current data, and unsupervised methods eschewing class labels altogether, and measuring relative movements between data; some approaches include using principal component analysis (PCA) for unsupervised concept drift detection, and the CUmulative SUM (CUSUM) test as an example of supervised (explicit) concept drift detection; however, as noted in [28], supervised methods suffer from the sheer amount of data required for labels (a rough estimate is provided therein, for the costs regarding a fraction of the actual data size of a Twitter data stream), while unsupervised methods have high false positive rates of drift detection, which prove to be a minor nuisance at best and slow down the entire classification at worst. Some ways around these are to combine multiple methods together; for example, in [1] where the PCA is used as a first line of detection and the statistical analysis method to reduce the false positive rate.

In addition to the rest of the system, detection of concept drift is aided by the use of two additional methods: principal component analysis (PCA) and a windowed method, as with the ensembles. As has been mentioned earlier, the accuracy of classifiers is known to decrease whenever a concept drift (gradual or abrupt) occurs; the overall idea of this is to use PCA as a first-line-of-defense where the incoming data is compared with that already present to determine the drift, if any, and use the ensemble window-based accuracy in the event that a false positive occurs; as a result, any incoming data which differs slightly from the rest — enough to trigger a detection by PCA — but has no significant bearing on the classification performance need not necessarily present an actual drift. The potential downside to this method, however, is that window-based methods are inherently delayed, with *reaction time* being dependent on the size of the window being used. ADWIN and similar methods [11] [12] [14] use variable window sizes to deal with this, an approach that is adapted here.

1) Principal Component Analysis

Principal concept analysis has been used in the previous literature [1] to perform unsupervised concept drift detection, an approach which undoubtedly has benefits regarding autonomy; as compared to supervised methods, it requires very little configuration (if any), and does not require a past 'history' of labeled data. On the other hand, it has problems regarding false positive rates, leading it to be coupled with other approaches such as statistical methods and tests [28]. Here, the principal component analysis approach as used by [1] is adapted; in effect, PCA is used to reduce the number of features in the incoming data so as to make it easier to analyse for deviations in the incoming data. Deviations which are then flagged by PCA, hence, signal a potential concept drift; however, since PCA does not always discriminate between concept drift amongst those in the same class or otherwise, it is likely to trigger false positives; as a result, it is merely the first measure for detecting concept drift, with the windowed approach being used afterwards.

D. Prototypical Representativeness

During the process of mining continuous data streams, known examples which are both fresh and stale can be made use of in classifying the incoming data. This is handled differently based on the method; for example, window-based techniques use a sliding window (of either fixed or varying size) to represent prototypes, while representativeness-based methods such as [1] and [2] use a metric known as *representativeness* $Rep(y)$ which is a time- and accuracy-dependent metric of how 'representative' a given prototype is; the basic idea to change how important a given

prototype is by increasing its importance (representativeness) if it is (a) newer, or (b) representing an incoming example correctly, and decreasing it otherwise (viz. time decay and inaccurate responses.)

We adapt this approach of representativeness based prototypical models, by making use of a max-heap to store prototypes, with the sorting criteria being the representativeness of the prototype. Instead of using a time-dependent metric, which may not always lend itself well to a suitable implementation regarding the relativity involved in measuring 'freshness' of data, the heap is used to store details of the prototypes' representativeness which is sorted (i.e. prototypes' positions rearranged whenever a prototype is 'selected' as the best match); this closely matches the approach by [1], with the exception of being time-independent: effective prototypes stored in the heap are not retired until new ones replace it, thereby avoiding the problem of short-lived representativeness. In effect, it can be assumed that the representativeness relationship follows a loose bathtub curve, where both new and old prototypes are given more importance: new ones because they more effectively represent the stream and old prototypes because they are on the verge of being retired (when a new prototype takes its place.)

E. C-HEAP Structure

In addition to that which has been mentioned earlier, C-HEAP uses a max-heap to store information about the prototypes; this is then linked with a graph that stores information about the concepts that are characteristic of the current data stream. In effect, C-HEAP consisted of a linked heap-graph, where each concept in the data stream which has been gathered by the classifiers is roughly linked to one prototype in the heap. Whenever a concept drift occurs, the graph is condensed via clustering, to reduce the number of concepts - so as to decrease space requirements and for reducing the effective number of examples required for the classifier training. While it is not always guaranteed for an equal relation to exist between the two (viz. one concept per prototype), clustering aims to minimize the number of concepts, and hence bring the ratio of prototype:concept as close as possible. The classifiers used are then trained on the concepts present in C-HEAP, so as to adapt to the stream at the current point in time, thereby learning the new properties of the stream.

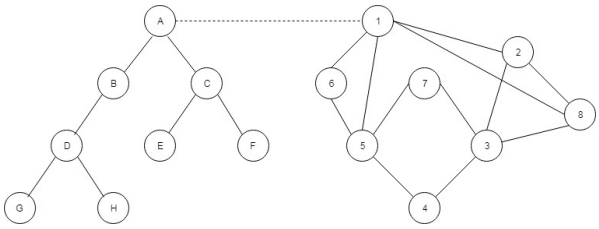

Fig. 2 Diagram of structure C-HEAP

F. HED-CDD

As mentioned before, HED-CDD makes use of C-HEAP to store prototypes on which to train classifiers on. A special global timer is maintained, called the drift measure timer which periodically checks for the presence of concept drift, as outlined in the earlier sections. Every time the timer expires, it is refreshed and the delay is adjusted, according to the compensation mechanism for adjusting λ - since λ is the rate of arrival of concept drifts, it can be assumed that the timer is set to the mean, i.e. $1/\lambda$. Concurrently, a set of classifiers are trained on the initial representative samples for the system; these will be used to classify the incoming data from the stream, in accordance with the bagging procedure [13]. In addition to the timer being refreshed when it runs out, the presence of concept drift is checked for as mentioned earlier (in particular, the accuracy of classifiers in a window decreases whenever a concept drift occurs, which is big enough, i.e. abrupt or gradual over a long period of time); if present, the list of current concepts in the system are clustered to reduce their number, and the classifiers trained on them (thereby capturing the concepts), so as to enable the representativeness based learning like in [1].

A listing of the algorithm has been given. A bagging procedure is used in **cons_bag** (which can be replaced with existing bagging procedures); **dss** is the data stream, **csrs** is the list of classifiers, and a lookahead L_q is employed to detect whether an upcoming change is imminent, by using the mean-items-in-system formula as detailed earlier. Change detection is handled by a combination of PCA and sliding window method as per the bagging procedure.

Algorithm 1 *HED-CDD*

```
procedure HED-CDD(λ, μ, d, dss, cons, csrs[])
    C-HEAP→ initialize(dss, cons)
    init_proto ← (C-HEAP→ get_proto())
    T_s ← 1/λ
    for i ∈ csrs do
        train(csrs(i), init_proto)
    end for
    for data ∈ dss do
        for i ∈ csrs do
            consume(csrs(i), data)
        end for
        cons_bag(csrs)
    end for
    for every T_s cycles do
        change ← check_concept_drift()
        if change or next_change then
            C-HEAP← cluster(new concepts)
            C-HEAP→ trim_all()
            new_proto ← (C-HEAP→ get_proto())
            for i ∈ csrs do
                train(csrs(i), new_proto)
            end for
            c ← (μ − (d × λ))/(μ − λ)
            T_s ← T_s × c
            λ ← 1/T_s
        end if
        next_change ← False
        L_q ← (λ²)/(μ × (μ − λ))
        if L_q > C-HEAP→ L_qmax then
            next_change ← True
        end if
    end for
    return C-HEAP
end procedure
```

V. ANALYSIS

The parameters of the system, along with a brief analysis of its theoretical operation are discussed below. These include formulae for theoretical (predicted) throughput of the system, along with the effective drift measure variation rate, as per the queuing model described in subsection II-B.

A. Throughput

As given earlier in equation (3), the mean time spent by a data item in a queue can be found using the mean-time-in-system formula:

$$\frac{1}{\mu - \lambda}$$

Finding the throughput, however, is a little bit more involved, because the queuing model used assumes a Poisson arrival rate (i.e. for rare occurrences); as a result, a simple throughput calculation will only account for the rate at which abrupt concept drifts pass through the system. For a more accurate result, the throughput of the abrupt concept drifts has to be included with the normal data rate, which is out of the scope of the text; as a result, the above calculation represents only a lower bound on throughput, or alternatively, the abrupt concept drift throughput.

B. Calculating Drift Measure

As stated earlier, whenever the timer T_s runs out, the system checks for any concept drifts which may have occurred between the last run of the timer and the present; if any abrupt drifts have occurred, the drift measure - representing the error in the data stream - is adjusted according to the convergence c (as detailed below), causing T_s to decrease. In effect, it starts checking more frequently, akin to a sliding window method [12]. On the other hand, if no concept drift has occurred in the meantime, the drift measure is decreased by a percentage of its total.

A proof to determine the minimum % convergence c that is required for T_s to converge to a stable value in the long run is as follows:

1) Abrupt concept drifts are assumed to be rare, and can be represented as a Poisson process with mean λ.
2) The drift measure directly corresponds to the arrival rate as per the queuing model, λ.
3) In the queuing model, the probability that a concept drift occurs is the same as the probability that there is no waiting time, i.e. the concept drift (the "customer" in the traditional queuing model) does not have to wait to be classified (or "served"). This probability is:

$$1 - P(\text{abrupt drift in stream} > 0)$$

$$= 1 - \frac{\lambda}{\mu}$$

4) After assigning the 'weights', i.e. $\times d$ when a concept drift has occurred and $\times c$ when it has not, it can be represented as:

$$d\left(\frac{\lambda}{\mu}\right) + c\left(1 - \frac{\lambda}{\mu}\right)$$

5) This results in T_s converging when the sum equals 1:

$$= (d - c)\left(\frac{\lambda}{\mu}\right) + c = 1$$

$$= \frac{d\lambda - c\lambda}{\mu} + c = 1$$

$$= \frac{c(\mu - \lambda) + d\lambda}{\mu} = 1$$

$$= c(\mu - \lambda) = \mu - d\lambda$$

$$\therefore c = \frac{\mu - d\lambda}{\mu - \lambda}$$

In effect, this can be viewed as the expected value of λ over the running time of the algorithm, which automatically converges if it remains stable, or fluctuates until a stable value is reached. That is, λ is iterated in order for it to stabilize, which will henceforth be referred to as λ_{iter} when used in this way.

A graph of c versus λ/μ (λ, not λ_{iter}) is shown as follows; since μ can be approximated to be the same regardless of the type of data used, the only factor is the rate of errors encountered during operation (i.e. drift-measure); it shows that for data streams which are expected to be very noisy, an appropriate value of both c and d has to be chosen so that the effective λ does not diverge. Effectively, λ is equivalent to λ_{iter} after each round of the algorithm.

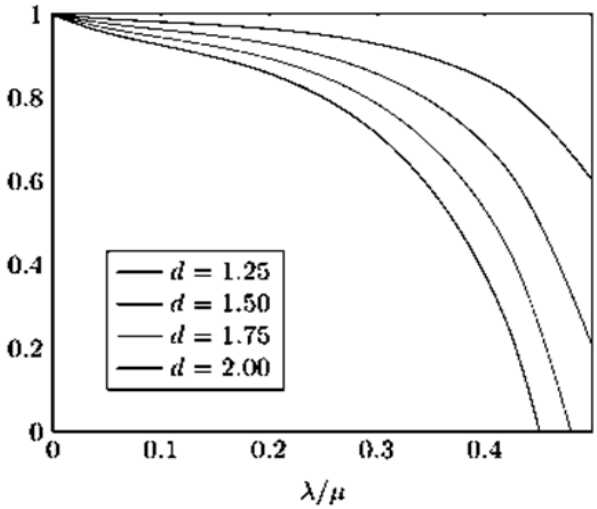

Fig. 3 Calculating parameter c for various values of d

VI. CONCLUSION

In this paper, a system for the detection of concept drift with adaptive measures, based on a hybrid queuing- and error-based model has been proposed, with *HED-CDD* using a *C-HEAP* for determining the presence of concept drift,

by basing it on previous methods regarding error-driven and representativeness-based learning. *HED-CDD* attempts to strike a balance between the most important prototypes, while at the same time combining it with an ensemble approach. In addition, by including a probabilistic means of handling concept drifts, it also tries to reduce the rate of 'false alarms' as common with other methods of unsupervised drift-detection techniques, and by including a prototypical approach, it becomes possible for the best of both techniques to be realized. In general, the involvement of probabilistic methods in data stream mining seems to be promising, with further investigation in related methods needed. While previous means involved modeling as a network of waiting lines, approaching it in a different manner (viz. queues of concept drifts in a data stream) may offer promising results as well. A final area of improvement is in handling other types of drift - that is, apart from noise, gradual and abrupt drift.

As it stands, *HED-CDD* retrains the classifiers when it detects an abrupt concept drift occur; however, like other adaptive-based learners, this may face the problem of class imbalance [29]; as a result, more work may be required in this area in order to tackle the class-imbalance problem effectively. Another area requiring focus is to investigate the effect of distributed data streams; namely, how *HED-CDD* can be adapted to work with distributed data streams, where data need not necessarily be independent and identically distributed (i.i.d). While it is initially suspected that multiple independent queues would be a possible solution, in reality it may involve changing the model to either a multiple-server model (i.e. $(M/M/s):(\infty/FIFO)$ queues) instead of a single-server model, as well as cross-referencing timers – if multiple ones are used - instead of holding the assumption of independence, with respect to concept drifts.

VII. ACKNOWLEDGEMENTS

We would like to thank an anonymous reviewer for preliminary review of the text; any errors are solely on us and not them. In addition, extra software not mentioned elsewhere in this paper include figures made using draw.io (available at www.draw.io)

REFERENCES

1. J. Shao, F. Huang, Q. Yang and G. Luo, "Robust Prototype-Based Learning on Data Streams," IEEE Transactions on Knowledge and Data Engineering, vol. 30, pp. 978-991, 5 2018.
2. J. Shao, Z. Ahmadi and S. Kramer, "Prototype-based learning on concept-drifting data streams," in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014.
3. T. Veerarajan, Probability, Statistics and Random Processes, Tata McGraw-Hill, 2008.
4. R. P. Sen, Operations research: algorithms and applications, PHI Learning Pvt. Ltd., 2009.
5. M. M. Gaber, A. Zaslavsky and S. Krishnaswamy, "Mining data streams: a review," ACM Sigmod Record, vol. 34, pp. 18-26, 2005.
6. J. R. Jackson, "Networks of waiting lines," Operations research, vol. 5, pp. 518-521, 1957.
7. B. Krawczyk and A. Cano, "Online Ensemble Learning with Abstaining Classifiers for Drifting and Noisy Data Streams," Applied Soft Computing, vol. 68, pp. 677-692, 7 2018.
8. G. Ditzler, M. Roveri, C. Alippi and R. Polikar, "Learning in nonstationary environments: A survey," IEEE Computational Intelligence Magazine, vol. 10, pp. 12-25, 2015.
9. H. Wang, W. Fan, P. S. Yu and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003.
10. V. Losing, B. Hammer and H. Wersing, "Tackling heterogeneous concept drift with the Self-Adjusting Memory (SAM)," Knowledge and Information Systems, vol. 54, pp. 171-201, 2018.
11. A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in Proceedings of the 2007 SIAM international conference on data mining, 2007.
12. A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby and R. Gavaldà, "New ensemble methods for evolving data streams," in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009.
13. N. C. Oza, "Online bagging and boosting," 2005.
14. P. Domingos and G. Hulten, "Mining high-speed data streams," in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 2000.
15. A. Bifet, G. Holmes, R. Kirkby and B. Pfahringer, "MOA: Massive Online Analysis," Journal of Machine Learning Research, vol. 11, pp. 1601-1604, 2010.
16. L. Tran, L. Fan and C. Shahabi, "Distance-based outlier detection in data streams," Proceedings of the VLDB Endowment, vol. 9, pp. 1089-1100, 2016.
17. T. L. Hayes, N. D. Cahill and C. Kanan, "Memory Efficient Experience Replay for Streaming Learning," arXiv preprint arXiv:1809.05922, 2018.
18. C. Böhm, C. Plant, J. Shao and Q. Yang, "Clustering by synchronization," in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010.
19. X. Chen, "An effective synchronization clustering algorithm," Applied Intelligence, vol. 46, pp. 135-157, 2017.
20. A. V. Novikov and E. N. Benderskaya, "SYNC-SOM," in Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods, 2014.
21. P. Sobhani and H. Beigy, "New drift detection method for data streams," in Adaptive and intelligent systems, Springer, 2011, pp. 88-97.
22. G. Brewka, "Artificial intelligence—a modern approach by Russell Stuart and Norvig Peter, Prentice Hall. Series in Artificial Intelligence, Englewood Cliffs, NJ.," The Knowledge Engineering Review, vol. 11, pp. 78-79, 1996.
23. S. Bach and M. Maloof, "A bayesian approach to concept drift," in Advances in neural information processing systems, 2010.
24. R. Dor, J. M. Lancaster, M. A. Franklin, J. Buhler and R. D. Chamberlain, "Using queuing theory to model streaming applications," 2010.
25. B. Foo and M. Schaar, "A distributed approach for optimizing cascaded classifier topologies in real-time stream mining systems," IEEE transactions on Image processing, vol. 19, pp. 3035-3048, 2010.
26. T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen and O. Shochet, "Novel Type of Phase Transition in a System of Self-Driven Particles," Phys. Rev. Lett., vol. 75, no. 6, pp. 1226-1229, 8 1995.
27. X. Chen, "Clustering based on a near neighbor graph and a grid cell graph," Journal of Intelligent Information Systems, vol. 40, pp. 529-554, 2013.
28. T. S. Sethi and M. Kantardzic, "On the reliable detection of concept drift from streaming unlabeled data," Expert Systems with Applications, vol. 82, pp. 77-99, 2017.
29. T. R. Hoens, R. Polikar and N. V. Chawla, "Learning from streaming data with concept drift and imbalance: an overview," Progress in Artificial Intelligence, vol. 1, pp. 89-101, 2012.