Towards real-time feature tracking technique using adaptive micro-clusters

Article

Published Version

Towards Real-Time Feature Tracking Technique using Adaptive Micro-Clusters

Mahmood Shakir Hammoodi, Frederic Stahl, Mark Tennant, Atta Badii

Abstract

Data streams are unbounded, sequential data instances that are generated with high velocity. Classifying sequential data instances is a very challenging problem in machine learning with applications in network intrusion detection, financial markets and sensor networks. Data stream classification is concerned with the automatic labelling of unseen instances from the stream in real-time. For this the classifier needs to adapt to concept drifts and can only have a single pass through the data if the stream is fast. This research paper presents our work on a real-time pre-processing technique, in particular a feature tracking technique that takes concept drift into consideration. The feature tracking technique is designed to improve Data Stream Mining (DSM) classification algorithms by enabling real-time feature selection. The technique is based on adaptive summaries of the data and class distributions, known as Micro-Clusters. Currently the technique is able to detect concept drift and identifies which features have been involved.

Key words: Data Stream Classification, High Velocity data Streams, Concept Drift, Feature Tracking

---

Mahmood Shakir Hammoodi, Frederic Stahl, Mark Tennant, Atta Badii
Department of Computer Science, University of Reading, PO Box 225, Whiteknights, Reading, RG6 6AY, UK,e-mail: m.s.h.hammoodi@pgr.reading.ac.uk, pre.mahmood.shakir@uobabylon.edu.iq, F.T.Stahl@reading.ac.uk, m.tennant@pgr.reading.ac.uk, atta.badii@reading.ac.uk

1 Introduction

Velocity in Big Data Analytics [1] refers to data that is generated at ultra high speed and is live-streamed whereupon the processing and storing of it in real-time (Data Stream Mining, DSM) constitutes significant challenges to our current computational capabilities [2]. DSM requires techniques that are incremental, computationally efficient and can adapt to concept drift for applications such as real-time analytics of chemical plant data in the chemical process industry [3], intrusion detection in telecommunications [4], etc. A concept drift occurs if the pattern encoded in the data stream changes. DSM has developed various real-time versions of established predictive data mining algorithms that adapt to concept drift and keep the model accurate over time, such as CVFDT [5] and G-eRules [6]. The benefit of classifier independent concept drift detection methods is that it gives information about the dynamics of the data generation [7]. Common drift detection methods are for example ADaptive sliding WINdow (ADWIN) [8], Drift Detection Method (DDM) [9] and the Early Drift Detection Method (EDDM)[10]. To the best of our knowledge, no drift detection methods devised to-date, can provide the potentially highly valuable insights as to which features are involved in the concept drift. For example, if a feature is contributing to a concept drift it can be assumed that the feature may have become either more or less relevant to the current concept. This causal responsibility theoretic perspective of the evaluation of concept drifts has inspired the development of an efficient real-time feature tracking method based on feature contribution information. A classification algorithm that exploits this approach would not need to examine the entire feature space for feature selection as this method feeds forward the feature contribution information. This paper therefore describes our concept drift detection method for data stream classification with the feature tracking information feed forward capability linking features to concept drifts for feature selection purposes. The proposed method could be used with any learning algorithm either as a real-time wrapper or a batch classifier or realised inside a real-time adaptive classifier [11, 6].

This paper is organised as follows: Section 2 introduces related works, Section 3 highlights the Micro-Cluster approach on which the presented feature tracking is built, Section 4 introduces the proposed concept drift and feature selection method, Section 5 evaluates the methodology as a proof of concept and Section 6 provides concluding remarks.

2 Related Works

Clustering and classification approaches are the most common DSM tasks [12, 13], as well as drift detection methods with feature selection as discussed below.

2.1 Data Stream Classification

Several methods have been proposed for predictive analytics on data streams. The most notable data stream classifier is probably the Hoeffding Tree family of algorithms. The Hoeffding Tree algorithm by Domingos and Hulten [11] induces a decision tree incrementally in real-time. The Hoeffding Tree was improved in terms of speed and accuracy by proposing a Very Fast Decision Tree (VFDT) [5]. Although, high accuracy using a small sample is the main advantage of these algorithms, concept drifting cannot be handled, because a created sub-tree can only expand from the child nodes onwards. Therefore, further improvements have been made by the development of adaptive trees that can alter entire sub-trees using a sliding window model. The new version of VFDT was termed CVFTD where C stands for Concept Drift [5]. Broadly, in CVFDT alternative sub-trees are induced over time and if an alternative sub-tree outperforms (i.e. in terms of accuracy) the current active sub-tree, then the current sub-tree is replaced with the alternative one. Further data stream classification algorithms have been proposed, such as Accelerated Particle Swarm Optimisation (APSO) with Swarm Search [14], a Similarity Search Structure Called the Rank Cover Tree (RCT) [15], an Online Data Stream Classification with Limited Labels [16], Prototype-based Classification Model [17], On Demand Classification [18], Adaptive Nearest Neighbour Classification for Data Streams [19], Ensemble based Classification [20], VFDR [21], and G-eRules [6].

Although, some of these works have focused on drift detection, none of these algorithms take online feature selection into consideration. If feature selection is applied at all, then it is mostly at the beginning of the data stream and it is assumed that the contribution of each feature to the concept remains invariant over time. However, this is an unrealistic assumption. The relevance of features for the concept may change over time and thus an online feature selection strategy may very well improve the predictive accuracy of the classifier. Also the data stream mining algorithms (i.e., classifiers) could potentially be speeded up as well, if features that have become irrelevant over time are taken out from the adaptation process.

2.2 Data Stream Clustering

Clustering algorithms need to use only a single pass over the training data in order to form their clusters, as a fast processing of the data is essential in order to keep the model as accurate as possible over time so as to rapidly adapt the model to emerging concept drifts. Several algorithms have been proposed for clustering data streams such as Birch [22], CluStream [23], HPStream [24], E-stream [25], POD-Clus [26], ClusTree [27], HUE-Stream [28], and MC-NN [29]. MC-NN is a development by some of the authors of this paper. MC-NN

is actually a classification algorithm, however, the underlying data structure on which the classification approach is based is that of Micro-Clusters. Micro-Clusters are statistical summaries which have been extended for predictive analytics from the aforementioned algorithms (i.e., Birch, CluStream, and MC-NN). The feature tracking technique presented in this paper is mostly based on an adaptation of MC-NN and hence MC-NN will be highlighted in more detail in Section 3.

2.3 Concept Drifts Detection Methods and Feature Selection

The aforementioned classifiers and clustering methods in Sections 2.1 and 2.2 are capable of detecting concept drift on their own. There also exist stand alone concept drift detection techniques that can be used in combination with batch learning algorithms and a sliding window approach. For example some of these techniques are CUmulative SUM (CUSUM) [30], Exponential Weighted Moving Average (EWMA) [31], ADaptive sliding WINdow (ADWIN) [8], Drift Detection Method (DDM) [9], Early Drift Detection Method (EDDM) [10], etc. However, we are not aware of any drift detection method that can also provide causal information linking an emerging drift to a set of features responsible for such drift. It is possible that over time, features may become more or less relevant for a data mining model, i.e. a decision tree. Thus in this paper we set out our approach to exploiting such responsibility-theoretic knowledge on the involvement of a feature in an emerging concept drift so as to develop a real-time feature tracking technique that can be used by a range of classification approaches as it feeds forward information about the specific features implicated in the emergence of a particular drift.

3 Micro-Cluster Nearest Neighbour (MC-NN)

This section summarises the MC-NN approach developed by some of the authors of this paper. MC-NN has originally been developed for predictive data stream analytics. However, the underlying Micro-Cluster structure has been adapted in order to develop a feature tracker for online feature selection purposes. Micro-Clusters in MC-NN provide a statistical summary of feature values retrieved from the stream over time stamps. In Tennant et al [29], the structure of Micro-Clusters is: $< CF2^x, CF1^x, CF1^t, n, CL, \epsilon, \Theta, \alpha, \Omega >$. Details about the Micro-Cluster structure components are listed in Table 1.

The feature centroid of a Micro-Cluster is calculated by $\frac{CF1^x}{n}$. Each Micro-Cluster is updated by adding a new instance to its nearest Micro-Cluster. The error ϵ is decremented by 1 if a new instance matches the CL. Otherwise, if

Table 1 The Structure of MC-NN Micro-Clusters

Structure Component	Description
$CF2^x$	a vector with the sum of squares of the features
$CF1^x$	a vector with the sum of feature values
$CF1^t$	a vector with the sum of time stamps
n	the number of data instances in the cluster
CL	the cluster's majority class label
ϵ	the error count
Θ	the error threshold for splitting the Micro-Cluster
α	the initial time stamp
Ω	a minimum (user defined) threshold for the Micro-Cluster participation minimum participation

the nearest Micro-Cluster does not match the CL of the new data instance, then the new instance is added to the closest or nearest Micro-Cluster (i.e., this is illustrated in Figure 1) matching the instance's CL, however, the error ϵ is incremented by 1 in both the Micro-Clusters concerned. This is illustrated in the flowchart in Figure 2.

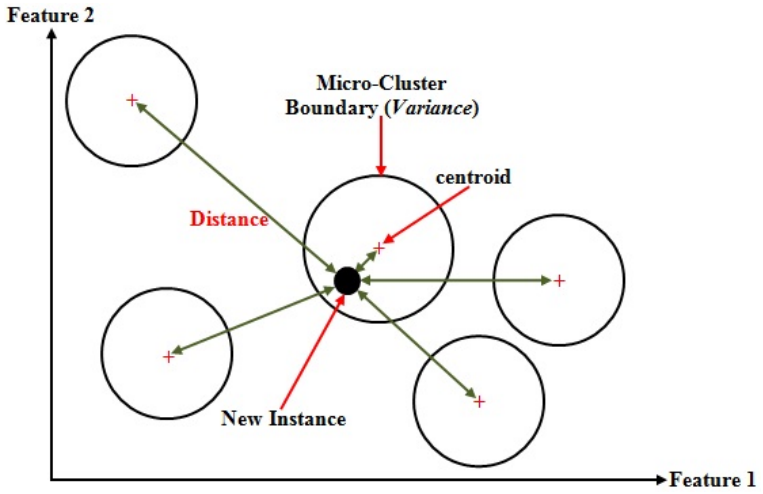


Fig. 1 An example of Adding A New Instance to the Nearest Micro-Cluster.

MC-NN splits a Micro-Cluster once the error count reaches Θ into two new Micro-Clusters while the original Micro-Cluster is removed to improve the fit to the evolving data stream. The new Micro-Clusters are placed about the original Micro-Cluster feature that has the greatest $Variance$ for a feature x which can be calculated using equation 1.

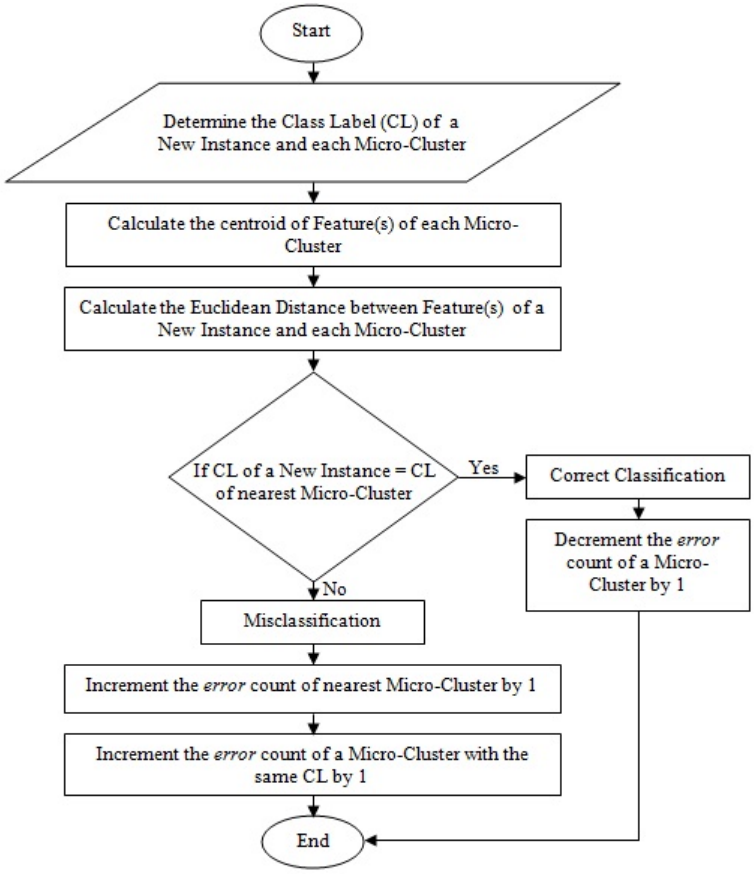


Fig. 2 The training task of MC-NN

$$Variance[x] = \sqrt{\left(\frac{CF2^x}{n}\right) - \left(\frac{CF1^x}{n}\right)^2} \tag{1}$$

The maximum variance of a feature x indicates that this feature may be the one most contributing to a mis-classification. The attribute with the maximum variance is either adding or subtracting the amount of variance (adding in one Micro-Cluster, subtracting in the other), as shown in Figure 3. New centroid values of the two newly generated Micro-Clusters are calculated.

MC-NN removes a Micro-Cluster if it has not participated recently in a fusion soak up, which can be calculated from $CF1^t$ by measuring the Triangle Number (Equation 2). We call this *Micro-Cluster Death*. Figure 4 shows an example of the Triangle Number.

$$Triangle\ Number\ \Delta(T) = ((T^2 + T)/2) \tag{2}$$

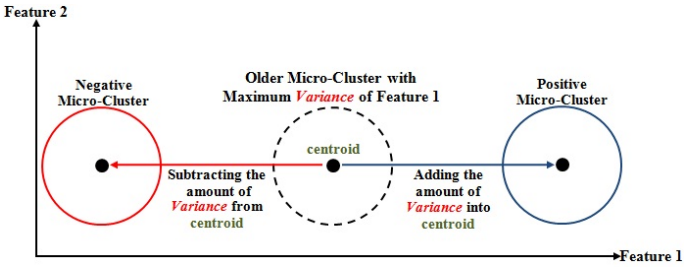


Fig. 3 Splitting of a Micro-Cluster.

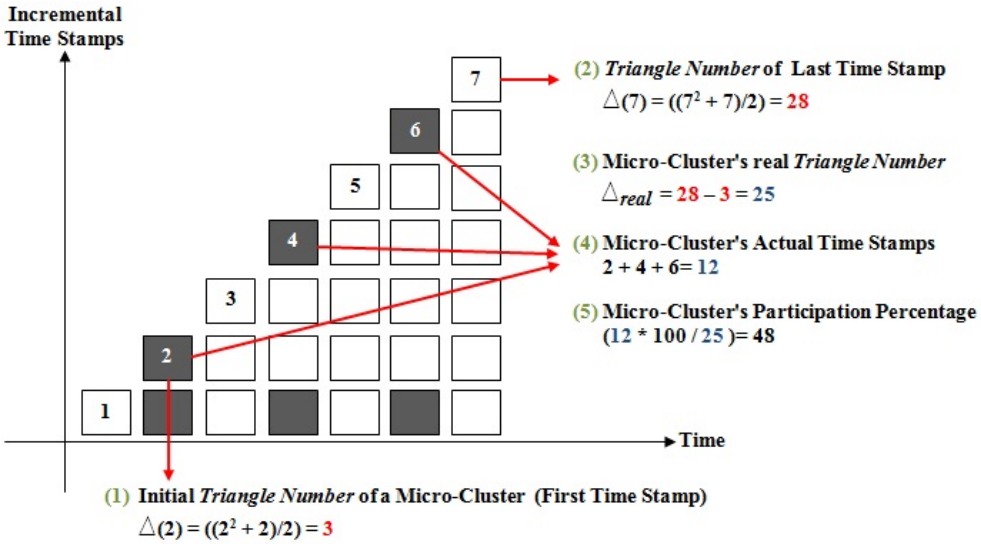


Fig. 4 An example of Triangle Number. The shaded shapes indicate that the Micro-Cluster has participated in absorbing new instances for a particular time stamp/window.

The Triangle Number gives more weight to recent instances than the older ones. If the participation percentage (shown in Figure 4) of a Micro-Cluster is lower than Ω, then the Micro-Cluster is removed.

Whereas MC-NN has been originally developed for classification purposes, we propose to adapt the MC-NN Micro-Cluster structure to detect concept drift and to enable the tracking of the involvement of each feature during concept drift.

4 Real-Time Feature Tracking using Adaptive Micro-Clusters

This section describes our technique for concept drift detection using feature tracking. The technique uses a sliding window approach and consists of four main components, which stand as the *normalisation* of each feature of a new training instance, a *Low Pass Filter* (LPF) to filter each feature of a Micro-Cluster in which the new instance has been fused in a *Micro-Cluster Feature Tracker with Drift Detection* (MCFT-DD) and Feature Analysis to identify irrelevant features, as shown in the flow chart in Figure 5. MCFT-DD is the central component of this technique and is based on the MC-NN approach highlighted in Section 3.

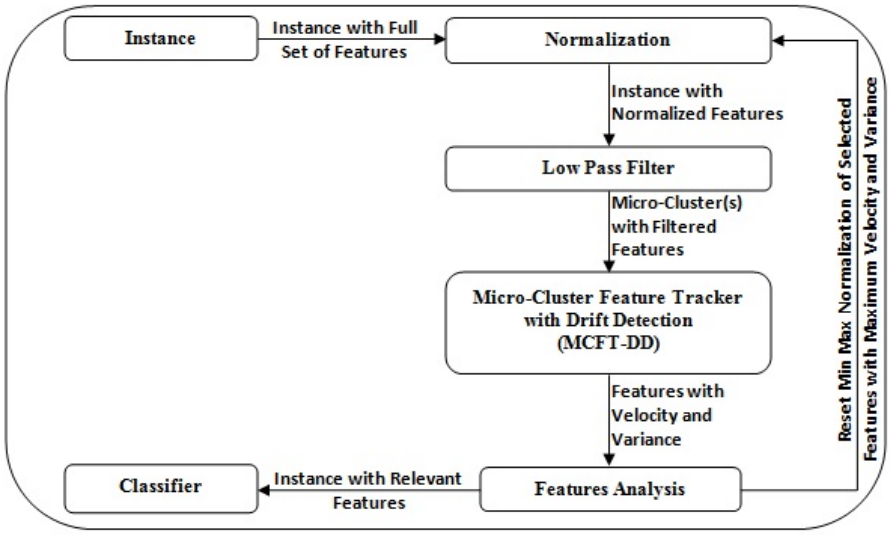


Fig. 5 Real-Time Feature Selection Technique using Adaptive Micro-Clusters

4.1 Normalisation

Normalisation is applied to fit the data (i.e., each feature of a new training instance) to be almost normally distributed in a pre-defined boundary such as [0,100]. We are using a Min-Max Normalisation technique which can be applied in real-time processing as minimum and maximum values of a feature x can be re-initialised as new instances arrive. Normalisation is used to avoid feature-bias which can potentially lead to mis-classifications as the relevant

relations between target class labels and features are considered by the classifier to be more or less important than they actually are. Equation 3 shows how to normalise a new data instance.

$$x = \left(\frac{current\ value - \min x}{\max x - \min x} \right) * (\max range - \min range) + \min range \quad (3)$$

Where x is a feature value of the new data instance, *max range* is the maximum range (default 100) of feature x and *min range* is the minimum range (default 0) of feature x. This equation is applied for every new data instance. Other normalisation techniques, such as the Z-Score have been considered. However, these alternative techniques rely on the standard deviation and the mean and thus require the buffering of data before normalisation can be applied, which is undesirable in real-time data analytics. However, the normalisation process described here can be updated incrementally instance by instance as shown in Figures 6 and 7 which show a flowchart and an example of our Min-Max Normalisation, respectively. Next the Micro-Cluster in which the normalised data instance needs to be soaked-up is determined using the same approach as described in Section 3 and accordingly a Low Pass Filter is applied to the Micro-Cluster.

4.2 Low Pass Filter (LPF)

LPF is a filter that passes signals with a frequency lower than a certain cut-off frequency α and attenuates signals with frequencies higher than the cut-off frequency. In this work LPF is used to avoid both outliers and noise in order to improve concept drift detection and feature tracking. Therefore, LPF is used to filter each normalised feature of a new training instance together with its nearest Micro-Cluster using Equation 4. This is illustrated in Figure 8 which shows a flowchart of an LPF.

$$new\ filter = \alpha * new\ value + (1 - \alpha) * old\ filter \quad (4)$$

The α threshold is set for 0.5 by default. Figure 9 shows an example of LPF. The centroid of each filtered feature of a Micro-Cluster is calculated. Micro-Clusters with filtered features are then passed to MCFT-DD.

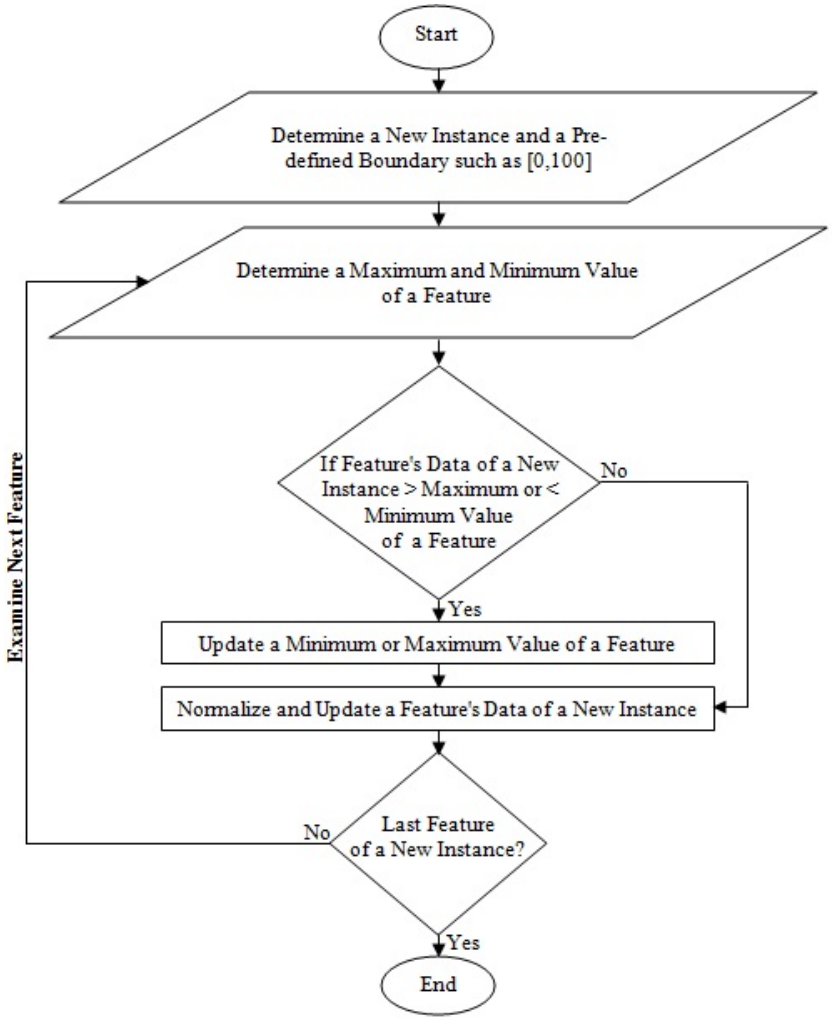
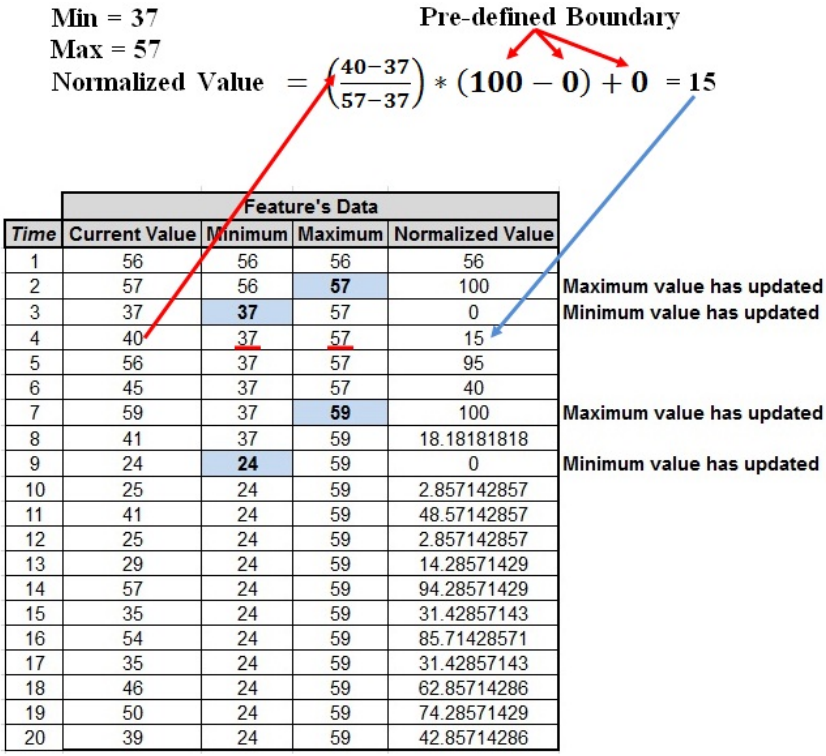


Fig. 6 Flowchart of Min-Max Normalization.

4.3 Micro-Clusters Feature Tracker with Drifts Detection (MCFT-DD)

The MCFT-DD component of our technique detects concept drift and tracks the relevance of features. Essentially, MCFT-DD enables this by monitoring the *split* and *death rate* of the Micro-Clusters in order to detect concept drifts plus monitoring $Velocity$ and $Variance$ of the individual Micro-Clusters in order to track which features are involved in a concept drift. The rates of Micro-Cluster splits and removals are calculated concurrently for each time



	Feature's Data			
Time	Current Value	Minimum	Maximum	Normalized Value
1	56	56	56	56
2	57	56	**57**	100
3	37	**37**	57	0
4	40	37	57	15
5	56	37	57	95
6	45	37	57	40
7	59	37	**59**	100
8	41	37	59	18.18181818
9	24	**24**	59	0
10	25	24	59	2.857142857
11	41	24	59	48.57142857
12	25	24	59	2.857142857
13	29	24	59	14.28571429
14	57	24	59	94.28571429
15	35	24	59	31.42857143
16	54	24	59	85.71428571
17	35	24	59	31.42857143
18	46	24	59	62.85714286
19	50	24	59	74.28571429
20	39	24	59	42.85714286

Maximum value has updated
Minimum value has updated
Maximum value has updated
Minimum value has updated

Fig. 7 An Example of Min-Max Normalization.

window of the data stream in order to detect whether a concept drift has occurred. This is illustrated in Figure 10.

If the percentage of the *split* and *death rates* differ from the mean (of the statistical windows) by 50% (default value), then this is considered to be a concept drift. In our experiments we used the default value of 50% as it yielded good results in most cases. However, the user may change this to a different threshold. Then a closer look can be taken into the individual features by examining their change in velocity. This is tracked through an extension of MC-NN's Micro-Cluster structure by: $< CF1^{hx}, n_h >$. Where these components are equivalent to $CF1^x$ and n. However, the h denotes that these components are *historical* summaries (taken from the statistical windows); the value of h is a fixed user-defined parameter and denotes how many time stamps the historical summaries are behind the recent ones. The *Velocity* of a feature x can then be calculated using equation 5. Figure 11 shows an example of feature $Velocity$ using a Micro-Cluster consisting of two features.

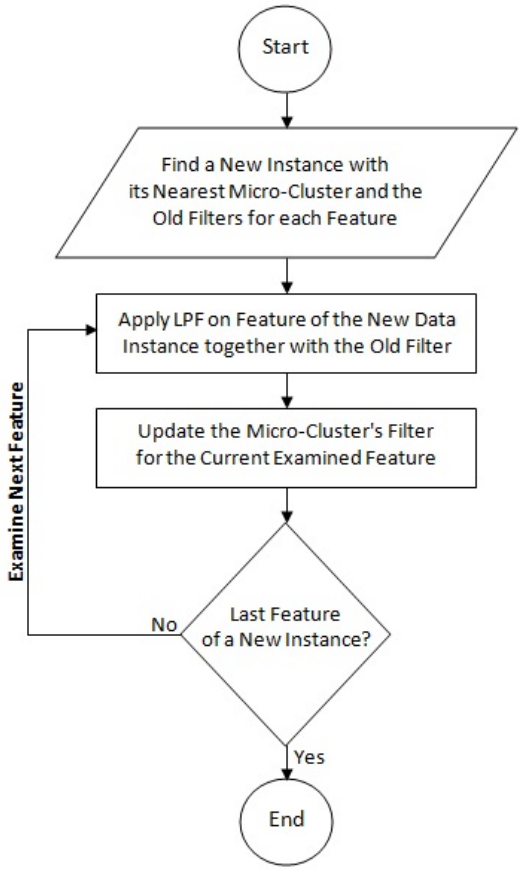
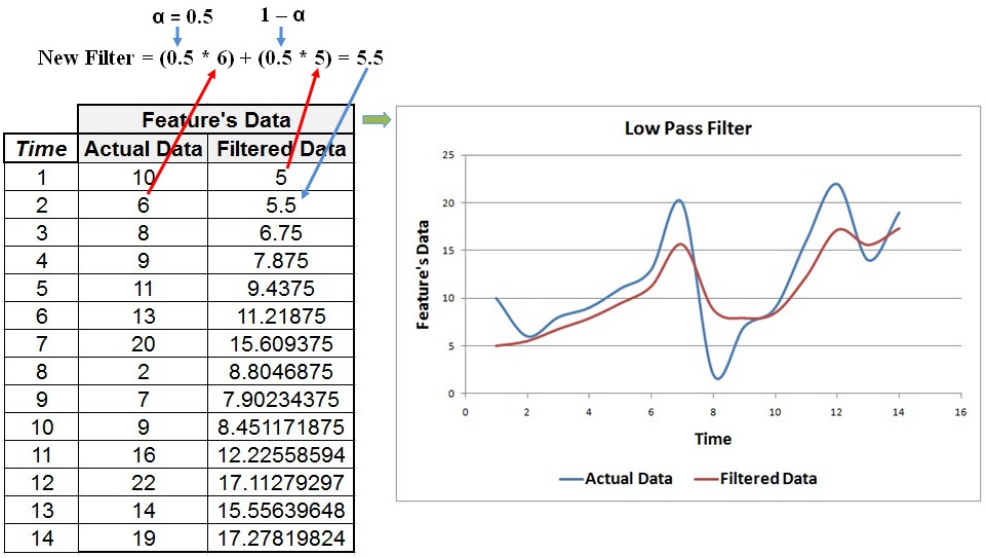


Fig. 8 Flowchart of Low Pass Filter.

$$Velocity[x] = \mid \frac{CF1^x}{n} - \frac{CF1^{hx}}{n_h} \mid \tag{5}$$

For the purpose of feature analysis, which will be discussed later in the Section, a counter is kept for each feature of a Micro-Cluster. The counter is incremented by 1 if the particular feature was the feature with the highest $Variance$ in the current time window. A high $Velocity$ combined with maximum $Variance$ during a concept drift indicates that the feature has changed. The assumption here is that this particular feature may have changed its contribution from a classification standpoint. Thus feature selection can be limited to examining only the features that have changed their velocity when there is a concept drift detected. Section 5 evaluates this approach as a principal proof of concept.



	Feature's Data	
Time	Actual Data	Filtered Data
1	10	5
2	6	5.5
3	8	6.75
4	9	7.875
5	11	9.4375
6	13	11.21875
7	20	15.609375
8	2	8.8046875
9	7	7.90234375
10	9	8.451171875
11	16	12.22558594
12	22	17.11279297
13	14	15.55639648
14	19	17.27819824

Fig. 9 An Example of a Low Pass Filter.

4.4 Feature Analysis

Feature analysis is facilitated using historical data (i.e. a statistical window between $time - 1$ and and *point-of-drift time*). For each time window, $Velocity$ rate and history of maximum $Variance$ are calculated separately. Once a drift is detected, features can be analysed. If a feature x with maximum $Velocity$ (re)appears in the history of maximum $Variance$, then the Max and Min value of normalisation of the feature needs to be reset. In addition, this feature can be designated as an irrelevant feature for a classifier, as shown in Algorithm 1.

5 Experimental Evaluation

This section evaluates the proposed technique in terms of adaptivity to concept drift with feature analysis. The implementation of our experiments was realised in the Java based Massive Online Analysis (MOA) framework [32].

5.1 Experimental Setup

For the experiments we used the following datasets.

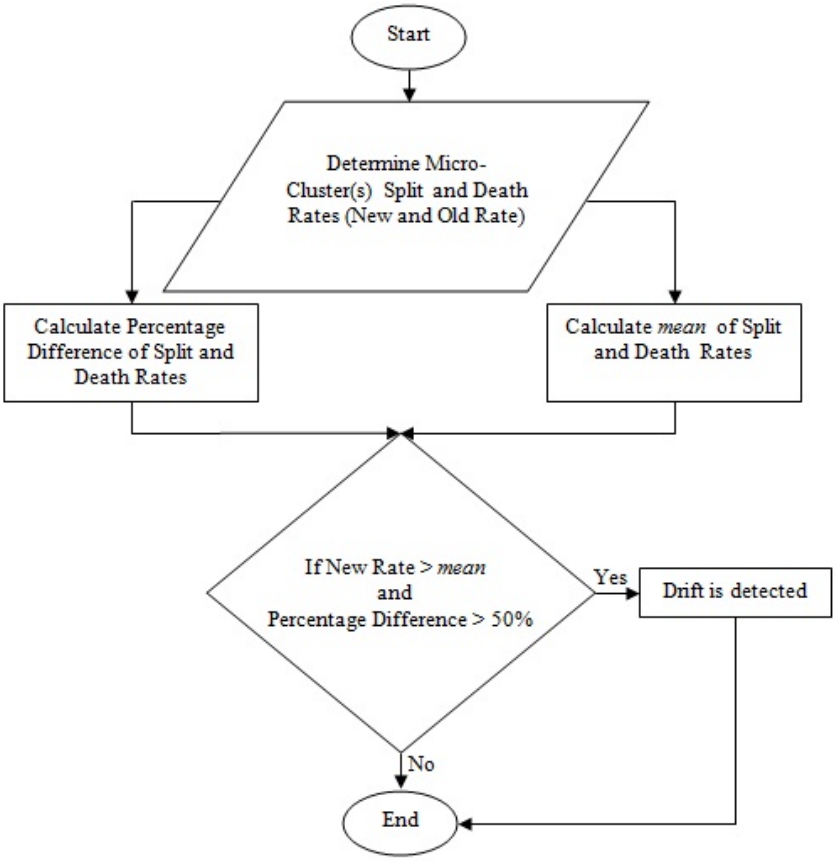


Fig. 10 Flowchart of MCFT-DD.

Algorithm 1 Feature Analysis

Input: $Velocity$[features] and history of maximum $Variance$[features] of a statistical window between $time - 1$ and *point-of-drift time*, and new instance

Output: Relevant features and reset Min and Max value of normalisation of selected features

for each drift detection **do**
  Apply *median* filter to $Velocity$[Features]
  for each feature with $Velocity$ greater than *median* **do**
    if feature has $Variance$ which is greater than 0 **then**
      $Normalization \Leftarrow reset\ Min\ and\ Max\ value$
      $new\ instance \Leftarrow delete\ irrelevant\ feature$
    end if
  end for
end for
$Classifier \Leftarrow new\ instance\ with\ relevant\ feature(s)$

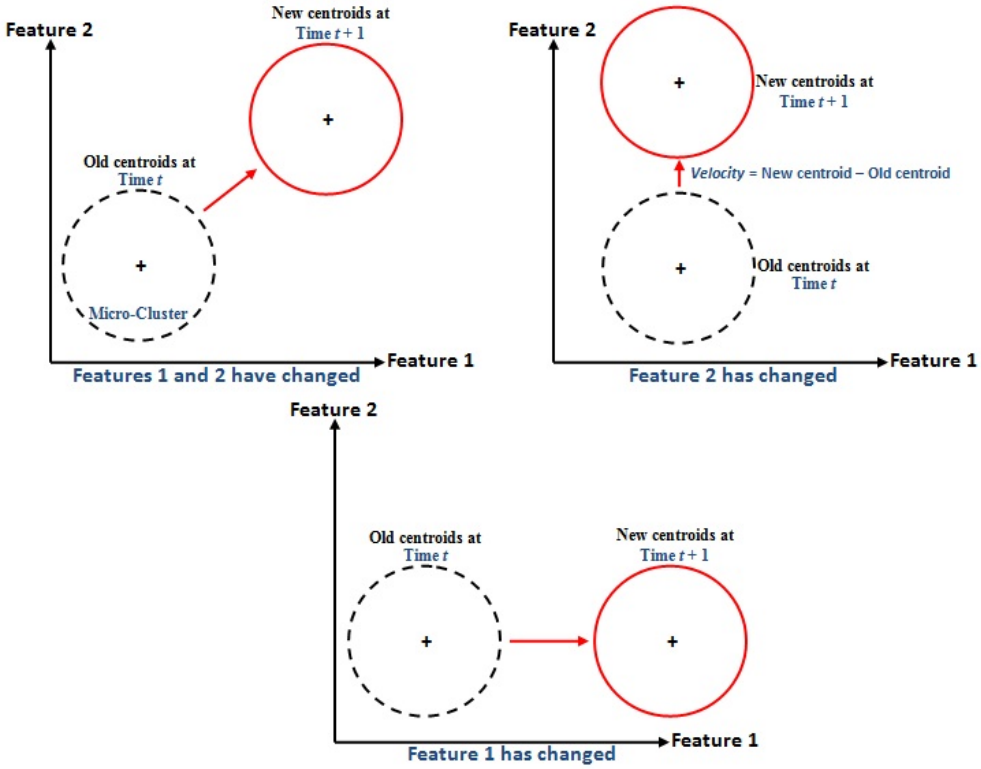


Fig. 11 An example of Feature Velocity.

Artificial Datasets,***SEA Generator***, this artificial dataset was introduced by Street and Kim [33], it generates an artificial data stream with continuous attributes. The third attribute is irrelevant for distinguishing between the class labels.

HyperPlane Generator, this artificial dataset creates a linearly separable model. It slowly rotates in *'D'* dimensions continuously changing the linear decision boundary of the stream. This constant concept change makes it very difficult for data stream classifiers to keep a good classification accuracy and computational efficiency. We generated a stream with continuous attributes. The 1^{st} attribute is irrelevant for distinguishing between the class labels (i.e., it is generated randomly).

The ***Random Tree Generator*** was introduced by Domingos and Hulten [11] and generates a stream based on a randomly generated tree. New examples are generated by assigning uniformly distributed random values to features, which then determine the class label using the randomly generated tree.

In each generated stream, 2 features are swapped making one of them irrelevant for the classification task as shown in Table 2, which presents an overview of generated streams including the setting of our method and which features have been swapped. In the experiments the expression **Time t** refers to a particular time window. i.e. according to Table 2 time *T=1* refers to instances 1-1000, *T=2* to instances 1001-2000, etc.

Table 2 Setup of the data streams. Drifts are generated through the individual data stream generators and by swapping features.

Generator	Instances	Features	CL	Time of Generated Drift by Generator	Swapped Features	Time of Swapped Features	Θ	Ω	Window Size
SEA	10,000	3	2	5	2 with 3	6	3	50	1000
HyperPlane	10,000	3	2	5	1 with 2	6	300	50	1000
Random Tree	100,000	3	2	5	2 with 3	5	5000	50	10,000

5.2 Results

The evaluation is focused on drift detection with feature analysis to identify relevant features for a classifier. Micro-Cluster Split-and-Death rates are used for detecting concept drifts, whereas, $Velocity$ and $Variance$ are used for analysing features.

For the experiments the default parameters stated in Section 5.1 of the method were used unless stated otherwise. In Figure 12 it can be seen that the Split-and-Death rates at the time of concept drift increase, indicating that the current set of Micro-Clusters no longer fit the concept encoded in the stream. Figures 13 and 14 show the velocities of the features in the Micro-Clusters and the history of the maximum $Variance$. In this case we know that the concept appeared due to our swapping of the features, hence we would expect a higher $Velocity$ and $Variance$ for those swapped features as can be seen in Figures 13 and 14. Regarding the Random Tree data stream it can be seen that the $Velocity$ increases rapidly for the swapped attributes as we would expect, but the $Variance$ is only increased for one of the drifting attributes. However, both results examined together give a good indication that attributes 2 and 3 should be re-examined for use for the adaptation of the classifier. Regarding the Hyperplane data stream, it can be seen that

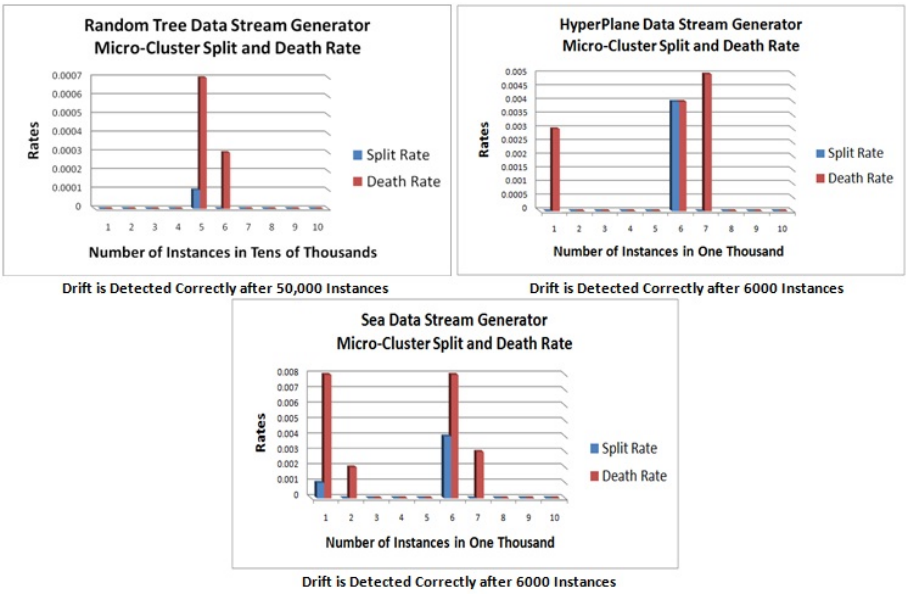


Fig. 12 The Micro-Cluster Split and Death Rates.

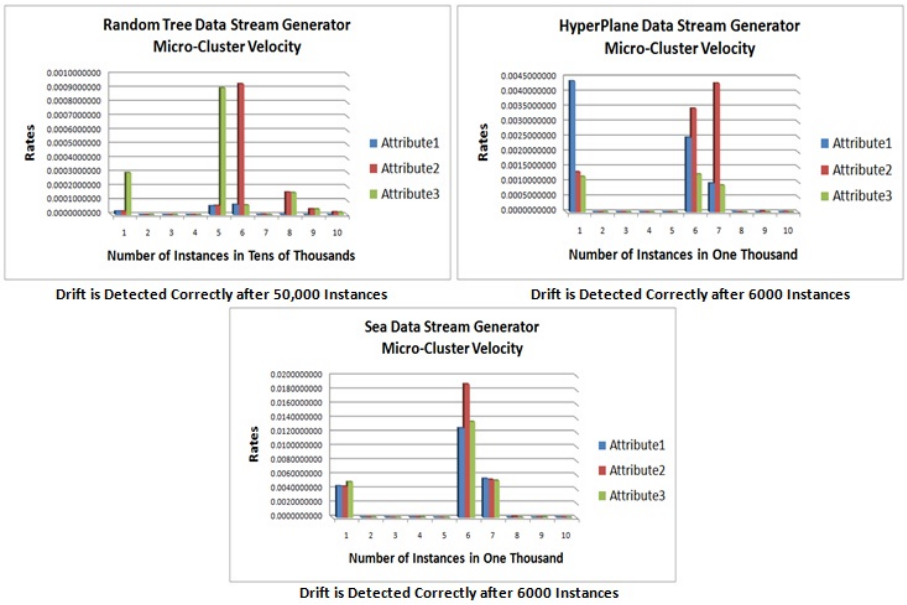


Fig. 13 The Micro-Cluster *Velocity* Rate.

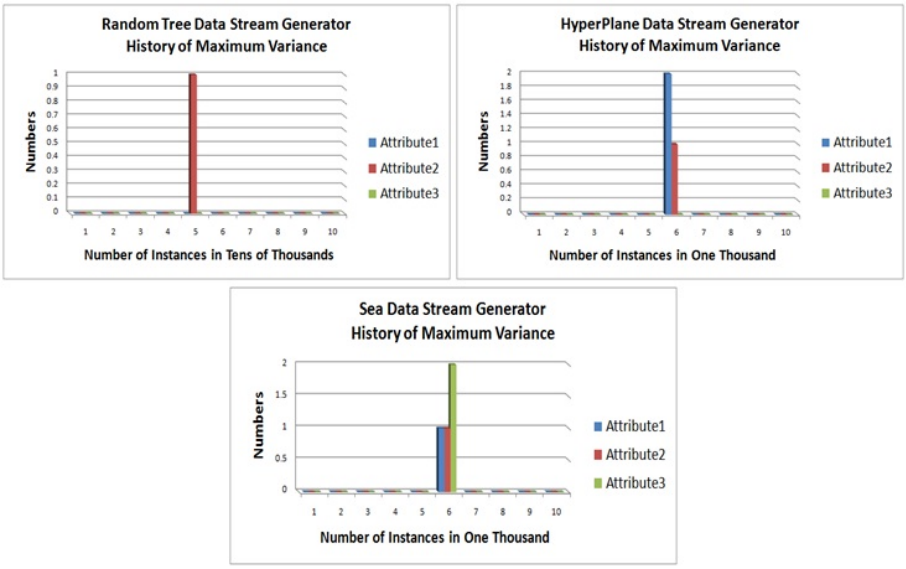


Fig. 14 History of Maximum $Variance$.

the $Velocity$ and $Variance$ increase rapidly for the swapped attributes as we would expect. This gives a good indication that attributes 1 and 2 should be reexamined for use for the adaptation of the classifier.

Regarding SEA we know that attributes 2 and 3 were involved in the concept drift, and they also exhibited the highest velocities. However, the $Velocity$ of attribute 1 is also very high, so no conclusive decision can be reached as to which attribute to reexamine. However, the $Variance$ of attribute 3 is much higher compared with the $Variance$ of the remaining attributes, hence it can be concluded that attribute 3 should be reexamined for use for the adaptation of the classifier. Thus, the approach did not find conclusive evidence for attribute 2 being involved in the concept drift.

In addition we compared the method with existing methods in terms of adaptation to concept drift and computational efficiency. The existing methods used were CUSUM, DDM, EDDM, EWMA, and ADWIN. We recorded the number and time of drift detections by each method and thus determined if the actual (real) drift had been correctly detected. This information can be found in Table 3. As can be seen, our technique always detected the drifts at the correct times.

If a method detects a false positive, then this would require unnecessary adaptation by the classifier using the drift detection method. We compiled in Table 4 a number of experimental runs (each with one actual drift crafted in it) that resulted in many true and false positives as detected by each method. As can be seen, our method did not detect any false positives and thus would only require minimum adaptation of the classifier using our method.

Table 3 Adaptation to Drift using The Proposed and Previous Techniques.

Generator	Technique	Number of Drift Detections	Time of Detected Drift	Drift Detected	Window Size
Sea	**The Proposed Technique**	1	6	**Correctly**	1000
	CUSUM	1	5	Incorrectly	
	DDM	2	5 and 8	Incorrectly	
	EDDM	2	5 and 8	Incorrectly	
	EWMA	9	1 to 9	**Correctly**	
	ADWIN	1	5	Incorrectly	
HyperPlane	**The Proposed Technique**	1	6	**Correctly**	
	CUSUM	1	5	Incorrectly	
	DDM	1	6	**Correctly**	
	EDDM	0	-	**None detected**	
	EWMA	7	2 to 8	**Correctly**	
	ADWIN	1	5	Incorrectly	
RandomTree	**The Proposed Technique**	1	5	**Correctly**	10,000
	CUSUM	1	4	Incorrectly	
	DDM	1	4	Incorrectly	
	EDDM	1	4	Incorrectly	
	EWMA	3	4,5, and 6	**Correctly**	
	ADWIN	4	4,5,6, and 7	**Correctly**	

Table 4 Summary of Adaptation to Drift.

Technique	Number of Experiments	True Positives	False Positives
The Proposed Technique	3	3	0
CUSUM	3	0	3
DDM	3	1	2
EDDM	3	0	2
EWMA	3	3	0
ADWIN	3	1	2
Total:	18	8	9

6 Conclusion

This paper introduced a novel Micro-Cluster based methodology for drift detection in data streams. This approach is to be distinguished from existing drift detection techniques in that, the proposed method is also capable of detecting the features implicated (deemed casually responsible for) the emerging concept drift evidenced by the velocity and variance of the Micro-Clusters in different dimensions. The method can thus be exploited for the implementation of real-time feature selection techniques thus powerfully supporting real-time Data Stream Mining eco-systems. The experimental proof of concept shows that the methods can successfully detect concept drifts and identify drifting features. Ongoing and future work comprises of more

in depth evaluation of the method on real data streams in combination with feature selection and a classification algorithm.

References

1. Ebbers, M., Abdel-Gayed, A., Budhi, V.B., Dolot, F., Kamat, V., Picone, R., Trevelin, J., et al.: Addressing Data Volume, Velocity, and Variety with IBM InfoSphere Streams V3. 0. IBM Redbooks (2013)
2. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM (2002) 1–16
3. Kadlec, P., Gabrys, B., Strandt, S.: Data-driven soft sensors in the process industry. Computers & Chemical Engineering **33**(4) (2009) 795–814
4. Jadhav, A., Jadhav, P., Kulkarni, P.: A novel approach for the design of network intrusion detection system (nids). In: Sensor Network Security Technology and Privacy Communication System (SNS & PCS), 2013 International Conference on, IEEE (2013) 22–27
5. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2001) 97–106
6. Le, T., Stahl, F., Gomes, J.B., Gaber, M.M., Di Fatta, G.: Computationally efficient rule-based classification for continuous streaming data. In: Research and Development in Intelligent Systems XXXI. Springer (2014) 21–34
7. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys (CSUR) **46**(4) (2014) 44
8. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: SDM. Volume 7., SIAM (2007) 2007
9. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Advances in artificial intelligence–SBIA 2004. Springer (2004) 286–295
10. Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., Morales-Bueno, R.: Early drift detection method. In: Fourth international workshop on knowledge discovery from data streams. Volume 6. (2006) 77–86
11. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2000) 71–80
12. Aggarwal, C.C.: Data streams: models and algorithms. Volume 31. Springer Science & Business Media (2007)
13. Gama, J.: Knowledge discovery from data streams. CRC Press (2010)
14. Fong, S., Wong, R., Vasilakos, A.: Accelerated pso swarm search feature selection for data stream mining big data. (2015)
15. Houle, M.E., Nett, M.: Rank-based similarity search: Reducing the dimensional dependence. Pattern Analysis and Machine Intelligence, IEEE Transactions on **37**(1) (2015) 136–150
16. Loo, H., Marsono, M.: Online data stream classification with incremental semi-supervised learning. In: Proceedings of the Second ACM IKDD Conference on Data Sciences, ACM (2015) 132–133
17. Shao, J., Ahmadi, Z., Kramer, S.: Prototype-based learning on concept-drifting data streams. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2014) 412–421

18. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: On demand classification of data streams. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2004) 503–508
19. Law, Y.N., Zaniolo, C.: An adaptive nearest neighbor classification algorithm for data streams. In: Knowledge Discovery in Databases: PKDD 2005. Springer (2005) 108–120
20. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2003) 226–235
21. Gama, J., Kosina, P., et al.: Learning decision rules from data streams. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence. Volume 22. (2011) 1255
22. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: an efficient data clustering method for very large databases. In: ACM Sigmod Record. Volume 25., ACM (1996) 103–114
23. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: Proceedings of the 29th international conference on Very large data bases-Volume 29, VLDB Endowment (2003) 81–92
24. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for projected clustering of high dimensional data streams. In: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment (2004) 852–863
25. Udommanetanakit, K., Rakthanmanon, T., Waiyamai, K.: E-stream: Evolution-based technique for stream clustering. In: Advanced Data Mining and Applications. Springer (2007) 605–615
26. Rodrigues, P.P., Gama, J., Pedroso, J.P.: Hierarchical clustering of time-series data streams. Knowledge and Data Engineering, IEEE Transactions on **20**(5) (2008) 615–627
27. Kranen, P., Assent, I., Baldauf, C., Seidl, T.: Self-adaptive anytime stream clustering. In: Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, IEEE (2009) 249–258
28. Meesuksabai, W., Kangkachit, T., Waiyamai, K.: Hue-stream: Evolution-based clustering technique for heterogeneous data streams with uncertainty. In: Advanced Data Mining and Applications. Springer (2011) 27–40
29. Tennant, M., Stahl, F., Gomes, J.B.: Fast adaptive real-time classification for data streams with concept drift. In: Internet and Distributed Computing Systems. Springer (2015) 265–272
30. Page, E.: Continuous inspection schemes. Biometrika **41**(1/2) (1954) 100–115
31. Ross, G.J., Adams, N.M., Tasoulis, D.K., Hand, D.J.: Exponentially weighted moving average charts for detecting concept drift. Pattern Recognition Letters **33**(2) (2012) 191–198
32. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive online analysis. Journal of Machine Learning Research **11**(May) (2010) 1601–1604
33. Street, W.N., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2001) 377–382