ClusCTA-MEWMAChart: A new clustering-based technique to detect Concept Drift in the presence of noise

ClusCTA-MEWMAChart: Una nueva técnica basada en clustering para detectar concept drift en presencia de ruido

Sonia JARAMILLO-VALBUENA 1; Jorge Mario LONDOÑO-PELÁEZ 2; Sergio Augusto CARDONA 3

Received: 15/08/2017 • Approved: 12/09/2017

Content
1. Introduction
2. Related word
3. ClusCTA-MEWMAChart
4. Implementation and method
5. Conclusion
Bibliographic references

ABSTRACT:
Many real-world applications generate data streams. Typically, data evolves over time and must be processed on-the-fly without the need for long term storage or reprocessing. In machine learning, the inherent variability or change over time of streams is referred to as concept drift. This phenomenon creates new challenges not present in classical machine learning techniques. In this paper, we present a new clustering-based technique to detect Concept Drift on data streams, named ClusCTA-MEWMAChart. We compare our algorithm experimentally with 4 different methods to detect concept drift from data streams and determine their robustness in the presence of noisy data. We conducted a set of experiments on synthetic datasets. The results show that the proposed approach has good performance.

Key words Concept drift; Classification; Data Stream Mining; adaptive learning

RESUMEN:
Muchas aplicaciones del mundo real generan grandes cantidades de datos que son continuos. A esto se le conoce como streams de datos. Típicamente, los streams de datos evolucionan con el tiempo. Este fenómeno, conocido como concept drift, crea nuevos desafíos que no están presentes en las técnicas clásicas de aprendizaje automático, dado que los datos deben ser procesados sobre la marcha sin la necesidad de almacenamiento a largo plazo o reprocesamiento. En este artículo, se presenta una nueva técnica basada en clusters para detectar Concept Drift en streams de datos, llamada ClusCTA-MEWMAChart. Este algoritmo es comparado experimentalmente con 4 métodos diferentes para detectar concept drift en streams de datos y se determina su solidez en presencia de ruido. Se realizan experimentos en datasets sintéticos. Los resultados muestran que el enfoque propuesto tiene un buen desempeño.

Palabras clave Concept drift; clasificación; minería sobre streams de datos, aprendizaje adaptativo
1. Introduction

In recent years many applications generating huge volumes of data in streamed manner have become commonplace. Network monitoring, phone records, social networks, ATM transactions, sensor data, and biological applications are examples of sources of data streams. Streaming data pose challenging research problems in order to respond to tasks such as statistics maintenance, storage, and real time querying and pattern discovery (Gama J.). Most traditional data mining techniques have to be adapted to analyze streaming data on the fly, normally trying to avoid storing the data in the stream (Le, Stahl, Gomes, Medhat Gaber & Di Fatta, 2014). The processing when the underlying data model changes over time is difficult because it must consider the presence of noise (Fang Chu, 2005) and the evolving nature of the stream, and therefore, should be able to detect and adapt quickly to concept drifting.

The term “concept drift” means that the statistical properties of the target concept change arbitrarily over time (Widmer & Kubat, 1996) (Wang, Yu, & Han, 2010) (Minku & Yao, 2012), causing the previously constructed model to be inconsistent and requiring an update or to be replaced with a new one to prevent accuracy deterioration (Wang, Yu, & Han, 2010) (Dongre & Malik, 2014).

Kelly et al. (Kelly, Hand & Adams., 1999) indicate that depending on the nature of the concept drift, it can occur in three different ways:

- The prior probabilities of classes P(c1),..., P(ck) might change over time.
- Class-conditional probability distributions might change P(X|ci), i = 1,…, k over time. This kind of concept drift is referred to as virtual drift, because it is possible that P(X|ci) change without affecting the previous classes (e.g symmetric movement to opposite directions).
- Posterior probabilities P(ci|X), i = 1,…,k might change. This kind of change is often known as real drift.

The concept drift can also be classified based on its speed. In particular the speed can be: sudden or gradual. The sudden drift (also referred to as concept change) represents irreversible and abrupt changes of samples of respective class. The gradual drift corresponds to slow and gradual changes over time (Gama, Zliobaite, Bifet, Pechenizkiy, & Bouchachia, 2014) (Dongre & Malik, 2014).

An important challenge for Concept Drift Detection algorithms is to differentiate the occurrence of outliers or noise in the data from actual changes in the target concept. The outliers refer to random deviations or statistical anomalies (Chandola, Banerjee & Kumar, 2009). There are some approaches designed to handle concept drift but only very few studies in the literature have documented the performance of these techniques in the presence of noise.

In this paper we propose a new clustering technique to detect Concept Drift on data streams, it is referred as ClusCTA-MEWMAChart. This approach uses multiple sliding windows and centroid tracking for maintaining enough knowledge about centroid behavior, and multivariate EWMA to report the occurrence of concept drift.


The paper is organized as follows. Section 2 reviews related works on concept drift detection. Section 3 describes the technique we propose. Section 4 shows implementation details and model evaluation. The last section provides a summary of the findings in this work.
2. Related word

There are several approaches designed to detect concept drift. In this section we describe 4 known algorithms in the literature: DDM, EDDM, GeometricMovingAverageDM and EWMAChartDM.

2.1. Drift Detection Method (DDM)

The Drift Detection Method (DDM) proposed by (Gama & Rodrigues, 2004) and described in (Jaramillo, Londoño & Cardona) uses a binomial distribution to describe the behavior of a random variable that gives the number of classification errors in a sample of size n. DDM calculates for each instance i in the stream, the probability of misclassification (p_i) and its standard deviation s_i; where s_i = \sqrt{p_i (1 - p_i) / i}. If the distribution of the samples is stationary, p_i will decrease as sample size increases. A stationary process (on mean and variance) is one whose statistical properties do not change over time (Nason, 2010). If the error rate of the learning algorithm increases significantly, it suggests changes in the distribution of classes, causing the current constructed model to be inconsistent with current data, and thus providing the signal to update the model.

DDM calculates the values of p_i and s_i for each instance and when p_i+s_i reaches its minimum value, it stores p_{min} and s_{min}. Then, DDM checks two conditions to detect whether the system is in the warning or drifting level:
- The warning level is activated when p_i+s_i ≥ p_{min} + 2s_{min}. Beyond this level, it stores the instances anticipating a possible change of context.
- The drift level is triggered when p_i+s_i ≥ p_{min} + 3s_{min}. In this level, DDM resets the variables (p_{min} and s_{min}) and the induced model. Later a new model is learnt using the instances stored since the warning level was triggered.

DDM shows good performance for detecting gradual changes (if they are not very slow) and abrupt changes, but has difficulties detecting drift when the change is slowly gradual. In that case, it is possible that many samples are stored for a long time, before the drift level is activated and there is the risk of overflowing the sample storage space (Baena-Garcia, et al., 2006).

2.2. Early Drift Detection Method (EDDM)

(Baena-Garcia, et al., 2006) proposed the Early Drift Detection Method as a modified version of DDM for improving the detection in presence of gradual concept drift, while retaining good performance with abrupt change. EDDM calculates average distance between two errors classification (p') and its standard deviation s'. The values p'_\text{max} and s'_\text{max} are maintained when p'+2.s' reaches its maximum value. EDDM considers two thresholds:
- Warning level. This level is triggered when (p'+2.s') / (p'_\text{max} + 2.s'_\text{max}) < \alpha. From this point on the samples are stored in advance of a possible change of context.
- Drift level. This level is triggered when (p'+2.s') / (p'_\text{max} + 2.s'_\text{max}) < \beta. This alarm triggers EDDM to learn a new model using the instances stored since the warning alarm. It also resets the variables p'_\text{max} and s'_\text{max} (Baena-Garcia, et al., 2006) (Jaramillo, Londoño & Cardona, 2018).

2.3. GeometricMovingAverage (GMA) Detection Method
This method, described in (Sadhukha, 2003) and (Jaramillo, Londoño & Cardona, 2018), uses two key ideas: the log-likelihood ratio and the exponential weighting of observations. The log-likelihood ratio is defined as:

$$s(y) = \ln \frac{p_{\theta_1}(y)}{p_{\theta_0}(y)}$$  \hspace{1cm} (1)$$

The key statistical property of this ratio is expressed in (2):

$$E\theta_0 (s) < 0 \text{ and } E\theta_1 (s) > 0$$  \hspace{1cm} (2)$$

Where $E\theta_0$ and $E\theta_1$ are the expectations of the random variables under the two distributions $p_{\theta_0}$ and $p_{\theta_1}$ respectively. From this, it is possible to define a change detector using the Kullback information (Eq. 3), to calculate the information between the two models (the mean values before and after change). This is shown in (4).

$$K(\theta_1, \theta_0) = E\theta_1 (s)$$  \hspace{1cm} (3)$$

$$E\theta_1 (s) - E\theta_0 (s) = K(\theta_1, \theta_0) + K(\theta_0, \theta_1) > 0$$  \hspace{1cm} (4)$$

The other underlying idea behind GMA refers to the exponential weighting ($y_i$) of observations (this means that higher weights are assigned on recent observations and lower weights on past ones) (Roberts, 2000) (Basseville & Nikiforov, 2012). The decision function is defined by eq. (5):

$$g_k = \sum_{i=0}^{\infty} y_i \ln \frac{p_{\theta_1}(y_{k-i})}{p_{\theta_0}(y_{k-i})} = \sum_{i=0}^{\infty} y_i s_{k-i}$$  \hspace{1cm} (5)$$

Where $y_i$ is defined as $y_i = \alpha(1 - \alpha)^i$, $0 < \alpha \leq 1$ and $\alpha$ is the forgetting factor ($\alpha$ gives more or less weight to the last received data).

The decision function, in a recursive manner, can be rewritten as shown in (6):

$$g_k = (1 - \alpha)g_{k-1} + \alpha s_k \text{ with } g_0 = 0$$  \hspace{1cm} (6)$$

The alarm of drift is defined by the stopping rule $t_\alpha = \min\{k: g_k \geq h\}$, where $h$ is a threshold chosen to tune the sensitivity and false alarm rate of the detector.

### 2.4. Exponentially Weighted Moving Average Chart Detection Method
3. ClusCTA-MEWMAChart

In this section, we introduce a new technique based on clustering and multivariate EWMA to detect Concept Drift on data streams. The method proposed is referred as ClusCTA-MEWMAChart. ClusCTA-MEWMAChart is a component of ClusCTA. ClusCTA is a new clustering method.

3.1. Clustering based on Centroid Tracking (ClusCTA)

Jaramillo et al. proposed ClusCTA. ClusCTA uses multiple sliding windows and centroid tracking to detect concept drift. The EWMA chart assumes knowledge of $\mu_t$ and $\sigma_X$ (the standard deviation of the stream). The EWMA estimator is a way of getting a new estimate of $\mu_t$, with older data. Roberts indicates that independent of the distribution of the $X_t$ variables, the mean and standard deviation of $Z_t$ can be calculated using:

$$\mu_{Z_t} = \mu_t, \quad \sigma_{Z_t} = \sqrt{\frac{\lambda}{2-\lambda}} (1 - (1 - \lambda)^{2t}) \sigma_X$$

Before a change occurs, $\mu_t$ is equal to $\mu_0$ and it is assumed that the value of $Z$ will fluctuate around it. After a change, $\mu_t$ will change to $\mu_1$, and the value of $Z_t$ will react to this, reporting that a change has occurred (it is when $Z_t > \mu_0 + L\sigma_{Z_t}$, where $L$ is a control limit and indicates how far $Z$ must diverge from $\mu_0$ before a change is triggered).

The EWMA chart can be used to detect changes in a stream, considering that the error can be seen as a sequence of Bernoulli trials, where $p_t$ is the probability of misclassifying an example at time $t$. An increase in the parameter $p_t$ indicates the occurrence of drifting concept. This detector, shown in (Yeh, 2008), assumes that $p_t$ has 2 possible values: $p_0$ and $p_1$, ($p_0$ before the change point and $p_1$ after the change) and assumes that $p_0$ and $\sigma_X$ are known. Since it uses the Bernoulli distribution, $\sigma_X$ depends on $p_{t_0}$ so that any change in the $p_t$ will also modify the standard deviation. To make this explicit, it is assumed that $\sigma_{Z_t} = \sigma_0$ before the drift point, and $\sigma_{Z_t} = \sigma_1$ after that. The EWMA estimator is defined as:

$$\sigma_{Z_t} = \sqrt{p_0(1-p_0)\frac{\lambda}{2-\lambda} (1 - (1 - \lambda)^{2t})}$$

The EWMA Chart DM improves this last approach obviating the need of knowing $p_0$, and introducing a second estimator of $p_0$ called $\hat{p}_{0,t}$, given by eq. (11).

$$\hat{p}_{0,t} = \frac{1}{t} \sum_{t=1}^{t} X_t = \frac{t-1}{t} \hat{p}_{0,t-1} + \frac{1}{t} X_t$$

EWMA Chart DM reports a change when $Z_t > \hat{p}_{0,t} + L\sigma_{Z_t}$. Finally, the pre-change standard deviation can then be calculated by $\hat{\sigma}_{0,t} = \hat{p}_{0,t} (1 - \hat{p}_{0,t})$ and the standard deviation of the EWMA estimator as shown in (12):

$$\sigma_{Z_t} = \sqrt{\hat{p}_{0,t}(1-\hat{p}_{0,t})\frac{\lambda}{2-\lambda} (1 - (1 - \lambda)^{2t})}$$

It is suggested that the value of $\lambda$ is chosen in the range $\lambda \in [0.1; 0.3]$ (Ross, Adams, Tasoulis & Hand, 2012).
for maintaining enough knowledge about centroid behavior. At the beginning, ClusCTA constructs a clustering model by running 5 times the k-Means++ algorithm on the first arrival sample window. It keeps the best of the five models, and based on the estimated radius of the obtained clusters, configures MCOD (Tran, Fan & Shahabi, 2016) (Kontaki, Gounaris, Papadopoulos, Tschilas & Manolopoulos, 2011) to perform an outlier or noise filtering process. To conclude the initialization phase, ClusCTA invokes ClusTree on the filtered window and it returns the clustering model of the initial sample window. ClusTree (Kranen, Assent, Baldauf & Seidl, 2011) is a compact and self-adaptive index structure, which maintains stream summaries and reports novelty and outliers. The application of the ClusTree on the filtered data allows obtaining a set of good quality initial centroids. It is important for ClusCTA to start from a good quality set of centroids (even if there is noise in the data stream).

The first centroids calculated are stored at an array of current centroids. After this, ClusCTA maintain a sliding window of samples per cluster with a maximum size of \(n/k\). When a new instance arrives, it is processed individually, associated with a cluster (using a Nearest Neighbor Criteria) and recorded in the sliding window of the corresponding cluster. Additionally, ClusCTA have a motion window per cluster (of size \(m\)). In this ClusCTA record the arrival time of the new sample and the impact that it has on the movement of its centroid. The computation of the tracking model uses the samples stored in the centroid motion window.

The centroid tracking is calculated using multivariate polynomial regression to fit points corresponding the centroid movements. In order to start the centroid tracking, ClusCTA needs to have good initial centroids. The centroids can be obtained with any method of clustering. The polynomial model is given by (11), where \(y(t)\) is the centroid of the \(i\)th cluster at time \(t\). The coefficients \(a, b, c, d\ldots\) describe the derivatives of order 0, 1, 2 (offset, velocity, acceleration) of the centroid.

\[
y(t) = a_i + b_i t + c_i t^2 + d_i t^3 + \ldots + z_i t^n + \varepsilon_i \quad (11) (13)
\]

The equation (13) can be expressed in matrix form, as it is shown in (14).

\[
y = X \cdot C \quad (14)
\]

\(X\) is a matrix, in which each row corresponds to a time vector \([1 \quad t_i \quad t_i^2 \ldots t_i^n]\), \(C\) is the matrix of coefficients (in this the coefficient vectors \(a, b, c, d.\ldots\) for each problem \(i\) are stacked horizontally) and \("y"\) is a matrix where each row is the estimated centroid at the time instant \(t\) (Rossi, Allenby & McCulloch, 2012).

The coefficients of the polynomial regression (15) are obtained by using a standard least squared error estimator (Pollock, 2007):

\[
C = (X'X)^{-1}X'y \quad (15)
\]

The centroid tracking process is repeated for every new instance. A polynomial regression model is computed for the centroid of each cluster and updated on new instance arrivals. When the model for a cluster is calculated ClusCTA get a new centroid, which is used to update the array of current centroids. The clustering resulting from centroid tracking is the set of points stored in this array.

ClusCTA only resets the current centroids when it detects that a centroid is lagging behind the actual cluster. After a reset, the clustering is restarted running again ClusTree. The centroid lags behind when clusters overlap or when a cluster abruptly changes its speed in a same execution. In order to detect if a centroid is lagging behind, ClusCTA use two strategies: a cluster size estimator and a Multivariate Exponentially Weighted Moving Average MEWMA filter (Hotelling, 1931) (Lowry, Woodall, Champ & Rigdon, 1992). The cluster size estimator, considers the number of instances that belong to the cluster (this is, the cluster queue), when it falls below a given threshold (for example, in a 60% of the sample sliding window size), ClusCTA concludes that cluster moved, but its centroid is lagging behind. To detect concept drift, ClusCTA applies multivariate EWMA on centroid movements.
3.2. ClusCTA-MEWMAChart

EWMA, described in the state of the art, can be extended as is shown in (16), for the multivariate case.

\[ Z_t = (1 - \lambda)Z_{t-1} + \lambda X_t \quad \text{for } t = 1, 2, \ldots, n \]  

(16)

where \( Z_t \) is the \( t \)th EWMA vector, \( X_t \) is the the \( t \)th observation array, \( \Lambda \) is a diagonal matrix with \( \lambda_1, \lambda_2, \ldots, \lambda_p \) on the main diagonal, and \( p \) is the quantity of variables (Lowry, Woodall, Champ & Rigdon, 1992) (NIST Sematech, 2014).

A well known method for identifying shifts in the multivariate case is the Hotelling’s \( T^2 \) control chart (Hotelling, 1931) (Lowry, Woodall, Champ & Rigdon, 1992). \( T^2 \) gives an out-of-control signal, as soon as:

\[ T^2_t = \sum_{i=1}^{n} Z_i^2 > h \]  

(17)

where \( h > 0 \) is a value defined to achieve a specified in control average run length or ARL and \( \Sigma_z \) is the inverse covariance matrix of \( Z \). The \((k,l)\) th element of the covariance matrix of the \( Z_i \), can be calculated as is show in (18). The average run length is the expectancy of the time before the control chart throws a false alarm that an in-control process going to out-of-control. Usually the appropriate values of \( h \) is obtained using a Monte Carlo Simulation or a Markov chain approach (Lowry, Woodall, Champ & Rigdon, 1992) (Patel & Divecha, 2013).

\[ \Sigma_{z_{i}}(k,l) = \lambda_k \lambda_l \left[ \frac{(1-(1-\lambda)^2i^2)(1-\lambda)^i}{(\lambda_k + \lambda_l - \lambda \lambda_k \lambda_l)} \right] \sigma_{k,l} \]  

(18)

Where \( \sigma_{k,l} \) is the element in the position \((k,l)\) of \( \Sigma \) (the covariance matrix of the observations). If \( \lambda_1 = \lambda_2 = \ldots = \lambda_p = \lambda \), then the above expression simplifies to that shown in (19):

\[ \Sigma_{z_i} = \left\{ \frac{i(1-(1-\lambda)^2)}{2-\lambda} \right\} \Sigma \]  

(19)

When \( i \) becomes large \((i \to \infty)\), \( \Sigma_{z_i} \) may be expressed as is presented in (20):

\[ \Sigma_{z_i} = \left\{ \frac{i}{i(2-\lambda)} \right\} \Sigma \]  

(20)

To report the occurrence of drift we obtain an estimate of coordinates of the centroids after passing through the filter EWMA. Before a change occurs, \( \mu_i \) is equal to \( \mu_0 \) and EWMA value will oscillate around it. After a change in the distribution of the arriving samples, it is detected that the metric \( T^2_t \) exceeds the threshold (17), which allows to generate a change report.

4. Implementation and method

We implement ClusCTA-MEWMAChart on top of the Massive On-line Analysis MOA framework (The University of Waikato, 2015), a widely known framework for data stream mining written in Java.
RandomRBFGeneratorEvents is a generator based on the random Radial Basis Function that adds drift to samples in a stream (Bifet A., 2012). The random radial basis function, described in (Bifet, Holmes, Pfahringer, & Gavalda, 2009) generates a fixed number of random centroids. Each centroid has a random position, a class label, a standard deviation and a weight. The instances are generated by selecting a centroid at random. For this process, the weights of the centroids are taken into consideration, so centroids with larger weight are more likely to be chosen. The chosen centroid determines the class label of the instance. The random radial basis function gives rise to a normally distributed hypersphere of instances enclosing each centroid. Drift is added by moving the centroids at a constant rate.

For the evaluation with DDM, EDDM, GeometricMovingAverageDM and EWMAChartDM, we use DriftDetectionMethodClassifier (Baena, 2012), a class for detect concept drift with a wrapper on a classifier. The chosen classifier is a Naive Bayes classifier and the concept Drift Detection Technique can be any of the 4 methods (DDM, EDDM, GeometricMovingAverageDM and...
EWMAChartDM. ClusCTA-MEWMAChart does not use DriftDetectionMethodClassifier.

We configure the RandomRBFGeneratorDrift, so that it creates five centroids of which 2 have no movement (speedOption=0) and 3 do have. In a same execution, a cluster with movement can change its speed (speedOption may take 3 different values: 0.01/500, 0.001/500 and 0.1/500).

In data streams, it is possible to have imbalanced classes, and the accuracy can be misleading when this happens. In this case, it may be desirable to have a model with a lower accuracy but with greater predictive power (Brownlee, 2014). For this reason, to assess the accuracy of the methods to identify concept drift, in addition to accuracy, we use Youden's J statistic.

<table>
<thead>
<tr>
<th>Concept drift detection method</th>
<th>Parameters used for evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusCTA-MEWMAChart</td>
<td>( \lambda = 0.10 )</td>
</tr>
<tr>
<td></td>
<td>h = 23%</td>
</tr>
<tr>
<td></td>
<td>WMcovariance=200 (movements of each centroid to calculate the covariance matrix of the ( \mathbf{Z}_i ) per cluster)</td>
</tr>
<tr>
<td></td>
<td>Wpolynomial=350 (sliding window of 350 samples to calculate the polynomial regression model).</td>
</tr>
<tr>
<td>EWMACheckDM</td>
<td>( \lambda = 0.10 )</td>
</tr>
<tr>
<td></td>
<td>minNumInstancesOption=30 (minimum number of stored instances before permitting the detection of a change)</td>
</tr>
<tr>
<td>DDM</td>
<td>minNumInstancesOption=30</td>
</tr>
<tr>
<td>EDDM</td>
<td>minNumInstancesOption=30</td>
</tr>
<tr>
<td>Geometric Moving Average Test</td>
<td>minNumInstancesOption=30</td>
</tr>
<tr>
<td></td>
<td>h=1</td>
</tr>
<tr>
<td></td>
<td>( \alpha = 0.99 )</td>
</tr>
</tbody>
</table>

To identify the best parameters for each concept drift detection method, we try several values. We use multiobjective optimization to obtain the best parametrization. For this purpose, we select the configuration by maximizing 2 classification quality indices, accuracy and Youden's J. The parameter MinNumInstancesOption did not show influence quality assessment of the methods. The best configuration of each method was selected to compare them against each other method. The results obtained are summarized in Figures 1-5.

![Figure 1](image_url)

Performance comparison for concept drift detection, noise level=0%
Figure 2
Performance comparison for concept drift detection, noise level=5%

Source: Own image

Figure 3
Performance comparison for concept drift detection, noise level=10%

Source: Own image
Figure 4
Performance comparison for concept drift detection, noise level=15%

Figure 5
Performance comparison for concept drift detection, noise level=20%
Figures 1 to 5 report the experimental result for noise level between 0% and 20%. From the observation, we can find that the classification accuracy and Youden’s J is always higher with ClusCTA-MEWMAChart, than with the 4 others methods of drift detection. When ClusCTA-MEWMAChart detects drift, a new model is induced by applying the clustering algorithm ClusTree in the samples stored.

5. Conclusion

In this paper, we present ClusCTA-MEWMAChart as a concept drift detector on centroids, and compare this with other similar approximations with respect to how they performed in the presence of concept drifts, with different velocities of change. ClusCTA-MEWMAChart shows good performance and the centroid tracking proves to be effective to react to concept drift. ClusCTA-MEWMAChart was the best concept drift detection method considering both the accuracy and Youden’s J.

ClusCTA-MEWMAChart is designed to detect gradual changes, that is, slow and gradual changes over time. Therefore, how to modify our method to detect abrupt changes is a subject of future research.

Future work includes to extend ClusCTA-MEWMAChart to consider novelty detection (new emergent classes) in data streams scenarios and online class imbalance learning. The process of concept drift detection in imbalanced data is a more complicated task than in the balanced classes, leading to a degradation in performance (Wang, Minku, Ghezzi, Caltabiano & Tino, 2012).

Bibliographic references


PAGE, E. S. **Continuous Inspection Schemes.** Biometrika, 41:100-115, 1954.


1. PhD in Engineering. Computer Engineering Dept., Universidad del Quindío, UQ. Armenia, (Colombia), sjaramillo@uniquindio.edu.co
2. PhD in Computer Science. Engineering Dept. Universidad Pontificia Bolivariana, UPB. Medellín, (Colombia), jorge.londono@upb.edu.co
3. PhD candidate in Engineering. Computer Engineering Dept., Universidad del Quindío, UQ, Armenia, (Colombia), sergio_cardona@uniquindio.edu.co