ISSN 2007-9737

Data Stream Classification based on an Associative Classifier

Karen Pamela López-Medina[1], Abril Valeria Uriarte-Arcia[2,*], Cornelio Yáñez-Márquez[1]

[1] Instituto Politécnico Nacional,
Centro de Investigación en Computación,
Mexico

[2] Instituto Politécnico Nacional,
Centro de Innovación y Desarrollo Tecnológico en Cómputo,
México

{auriartea, cyanezm}@ipn.mx, klopezm2021@cic.ipn.mx

Abstract. Currently, the diversity of sources generating data in a massive online manner cause data streams to become part of many real work applications. Learning from a data stream is a very challenging task due to the non-stationary nature of this type of data. Characteristics such as infinite length, concept drift, concept evolution and recurrent concepts are the most common problems that need to be addressed by data stream learning algorithms. In this work an algorithm for data stream classification based on an associative classifier is presented. This proposal combines a clustering algorithm and the Naïve Associative Classifier for Online Data (NACOD) to address this problem. A set of micro-clusters (MCs), a data structure that summarizes the information of the current data, is used instead of storing the whole data. The MCs are continually updated with the arriving data, either to create new MCs or to update existing ones. The added MCs helps to deal with concept drift. To assess the performance of the proposed model, experiments were carried out on 3 data sets commonly used to evaluate data stream classification algorithms: KDD Cup 1999, Forest Cover Type and Statlog (Shuttle). Our model achieved higher accuracies than those achieved with algorithms such as data stream version of Naïve Bayes and Hoeffding Tree, the average accuracies achieved were for KDD Cup 1999: 100%, Statlog (Shuttle): 99.01% and Forest Cover Type 70.44%.

Keywords. Data stream classification, associative classifier, concept-drift.

1 Introduction

Day by day, humans are taking advantage of knowledge that we acquire to solve our tasks more easily, one of them is known in pattern recognition as Classification [10]. With great precision and according to desired parameters we can classify experiences, people, tangible or intangible things thanks to the knowledge we have previously acquired, but what happens when suddenly the nature of the data has changed?

This phenomenon is named *concept-drift*, one of the main challenges of working with data streams [13]. The detection of categories or classes has had an impact on various sectors, both in research and in industry; as an example the companies that manage their sales on Internet.

This has been acquiring greater importance since the data that is generated every day is more and larger, so it is necessary to generate efficient tools for data analysis and decision making process over the data streams [6].

The nature of data streams is dynamic and therefore they are always evolving [2]. Given this situation it is not so easy to treat them with simple and static strategies. The main challenges we face when working with data streams are:

1. *Infinite length:* the analysed data is potentially infinite so it would be impossible to store all this information.

Table 1. Characteristics of data sets used for classification performance evaluation

Data sets			
Data set name	**Number of classes**	**Number of attributes**	**Number of instances**
KDD Cup 1999	10	41	489,795
Forest Cover Type	7	54	581,012
Statlog (Shuttle)	7	9	58,000

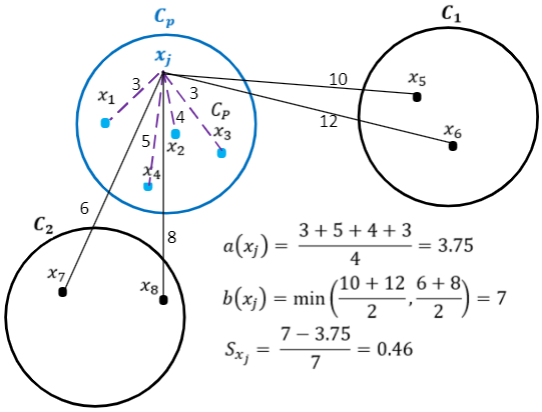

Fig. 1. Example of how the silhouette coefficient is calculated

2. *Concept-evolution:* is the phenomenon that occurs due to the presence of new unknown classes in the data.
3. *Concept-drift:* are changes in input data attribute values that indicate a change in data stream behaviour.
4. *Feature evolution:* occurs when new features appear in the data stream and initial features may disappear.

Data Mining of dynamic data stream is of great importance in the field of Machine Learning. These scenarios require adaptive algorithms that are able to process the input instances in real time, adapt to potential changes in the data, use limited computational resources, and be robust to atypical events that may occur [16]. A great deal of Machine Learning approaches have been developed to target *concept-drift detection*, such as neural networks [28, 18], active learning [12], instance based classifiers [5], Bayesian techniques [15], and ensembles classifiers [12], among others.

Data stream clustering approach is being widely applied in *data stream* scenarios. Clustering naturally adapts to speed and memory restrictions of data stream learning since it can maintain the information summarized of the cluster without the need of storing all the instances observed in a data stream.

In current literature a great deal of work addressing clustering in this type of scenarios can be found [14]. The works mentioned above provide the main ideas for the proposed solution in this document, which aims to implement a model for data stream classification that can handle concept-drift.

The rest of the paper is organized as follows: in section 2 materials and methods are described. In the following section, 3, the proposed solution and its flow chart are introduced. In section 4, the results and discussion are presented and finally in Section 5, the conclusions and proposals for future work are provided.

2 Material and Methods

This section is composed of two subsections necessary for the development of the proposed solution. In Section 2.1 the materials used in this paper are described. These materials consist of three data sets commonly used for data stream classification and the framework used to run some experiments.

In Section 2.2, the NACOD algorithm is described, which is the classifier used in this proposal. In addition, concepts such as: k-means, Silhouette coefficient and Global Hybrid Online Similarity are described.

Table 2. Example of initial training data

Number of instance	Attribute 1	Attribute 2	Attribute 3	Original Class
1	0.85	Yes	5	Class B
2	0.36	Yes	10	Class A
3	0.97	No	7	Class B
4	0.14	Yes	15	Class A
5	'?'	No	6	Class B

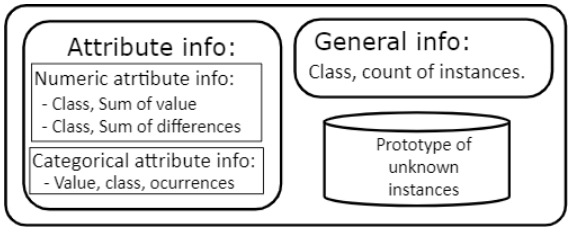

Fig. 2. Archive to store knowledge in NACOD classifier

2.1 Materials

2.1.1 Algorithms and Data Sets

This section describes the algorithms and data sets used during the experimental phase and explains the main reasons why these were selected.

MOA: to evaluate the results obtained, a comparison with other data stream classification algorithms was carried out. The MOA implementation of these algorithms was used. MOA [3] is an open source framework for massive online analysis. This framework provides several tools to deal with data stream scenarios, such as the data stream version of Naïve Bayes [3] and Hoeffding Tree [9]. The proposed methodology has been tested with three real data sets, that are often used in current works of data stream classification mainly because their size is suitable to simulate data stream scenarios. The selected data sets are:

– **KDD Cup 1999**: this is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, includes a wide variety of simulated intrusions in a military network environment. The size of the original data set has more than four million instances but commonly only 10% of them are used to save time and computational resources.

  In this work the 10% version of the data set was used, additionally only the 10 majority classes were used due to the high class imbalance produced if all the classes of this data set would be used. This decision was made since the scope of this work is not addressing the class imbalance problem.

– **Forest Cover Type**: describes seven forest cover types (classes) on a 30×30 meter cell obtained from the United States Forest Service (USFS) Region 2 Resource Information System (RIS) data. In addition, it has been used in several researches for data stream classification. The class refers to the type of forest that covers a defined area.

– **Statlog (Shuttle)**: this data set has 7 classes concerning the behavior of the pressure valve of a rocket. It has $58,000$ instances and 9 attributes, all of which are numerical. Approximately 80% of the data belongs to class 1. Therefore, the default accuracy is around 80%. Characteristics of data sets used for classification are shown in Table 1.

  The number of classes in the selected data sets varies between 7 to 10, number of attributes between 9 to 54, the minimum instance number is $58,000$ and the maximum is $581,012$. KDD Cup has 3 categorical attributes and the rest numerical. On the other hand, the Forest Cover Type and Statlog (Shuttle) attributes are integers. None of these data sets contain missing values.

2.2 Methods

The methods used in this article include clustering, specifically *k-means*, Silhouette coefficient, the associative classifier: *NACOD*, the similarity operator: *GHOS*.

2.2.1 Clustering

Clustering is an unsupervised learning task; it means that it works with data without labels. It consists of distributing a set of instances into groups according to their similarities.

General info:

(Class, count of complete instances per attribute):

Class A 2, 2, 2
Class B 2, 3, 3

Attribute info:

Categorical attribute info

(Attribute, Value, class, ocurrences):
A2, Yes, Class A, 2
A2, Yes, Class B, 1
A2, No, Class A, 0
A2, No, Class B, 2

Numeric Attribute info

(Attribute, Class, Sum of Values)
A1, Class A, 0.50
A1, Class B, 1.82
A3, Class A, 18
A3, Class B, 25

(Attribute, Class, Sum of differences $\sum_{i=1}^{n}(x_i - \tilde{x}\ )^2$)

A1, Class A, 0.0242
A1, Class B, 0.0052
A3, Class A, 13.50
A3, Class B, 2.0

Prototype

Class A: PA = (0.25, Yes, 12.5)
Class B: PB = (0.25, No, 6)

Fig. 3. Archive obtained by NACOD according to the training instances of figure 2

The major difference with respect to classification is that the number of groups is unknown before starting the learning process and the task is to create them [24]. In this work, the *k-means* algorithm will used. Next sub-section introduces theory of this algorithm.

2.2.2 K-Means

Proposed by MacQueen in 1967 [17], it consists of 2 iterative stages: the first one, an observation is assigned to the nearest cluster and in the second one cluster center is calculated according all the observations belonging to that cluster.

The algorithm does not conclude until there is no need to move any observation point to a different cluster [21]. Due to its simplicity, *k-means* [20] is one of the most used algorithms in clustering. It is known for its faster computational speed and superior performance on large data sets in comparison to other clustering techniques [29].

2.2.3 Silhouette Width Criterion

Silhouette width criterion was first defined in [25], also called Silhouette coefficient. It is based on geometrical considerations about separation and compactness of clusters and it helps to validate the micro-clusters and decide if they must be added to the model.

Consider the j^{th} element of a data set x_j which belongs to a given cluster $p \in \{1, \ldots, k\}$ and a_p be the average distance of this element to all the other elements in the cluster p. Also, let $d_{q,j}$ be the average distance from this element to all objects in other cluster, where q be each different cluster, that is $q \neq p$. Finally, let $b_{p,j}$ be the minimum $d_{q,j}$ computed over $q = 1, ..., k$, $q \neq p$, which represents the average dissimilarity of object x_j to its closest neighboring cluster. Then, the Silhouette coefficient of the individual object x_j is defined as:

$$S_{x_j} = \frac{b_{p,j} - a_{p,j}}{\max(b_{p,j}, a_{p,j})}. \quad (1)$$

Figure 1 represents an example of how Silhouette coefficient of an x_j is calculated.

2.2.4 NACOD

Recently, Villuendas-Rey *et al.* [26] introduced the Naïve Associative Classifier for Online Data (NACOD), a decision-making algorithm within the associative approach. This classifier can deal with some data complexity issues such as: missing values, hybrid and incomplete data, which significantly complicate the operation of the learning algorithms. NACOD is divided in three phases: training, classification and updating.

Training phase. It consists on storing relevant knowledge information about an initial set of labeled data.

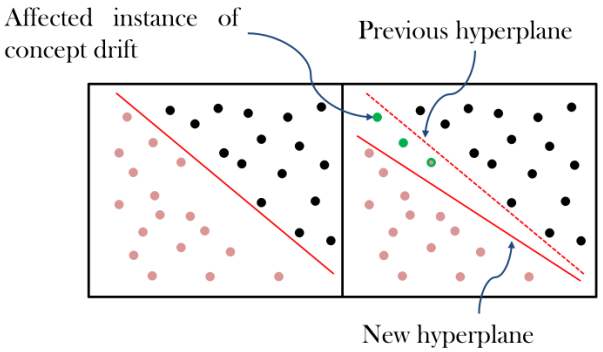

Fig. 4. Concept drift

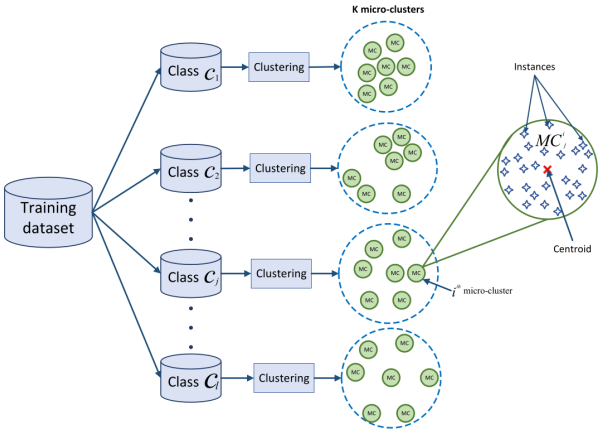

Fig. 5. Train phase

Suppose the following data sets as shown in Figure 2. The main objective in this section is to create an archive to store knowledge about known classes and attributes. Figure 2 shows how this information is handled. This archive also stores a prototype of each of the classes seen so far, like that: the class prototype is selected by using class mean for numeric attributes and class mode for categorical attributes.

For each numeric attribute, the classifier makes the sum of its complete values, and the squared sum of differences between current values and current mean. The current mean is computed by dividing the sum of complete values by the count of complete instances. The squared sum of differences allows the classifier to compute an estimation of the standard deviation of the attribute values, without the need of storing past data.

Figure 3 represents the archive corresponding to the training set described in Figure 2. NACOD uses the prototypes in the archive to compute similarities among instances. Instead of comparing the instance to classify with respect to the entire training set, it compares it with respect to the class prototypes.

Classification phase. To classify a new instance and make a decision, NACOD first computes its overall Global Hybrid Online Similarity (GHOS) with respect to the classes in the current training set (prototypes), equation 2. Then, it will assign the instance to the class with maximum similarity.

Let x be the sample to classify, let $c_j \in Y$ be a decision class, let y be the prototype of the training set of label c_j. The proposed similarity operator, named as Global Hybrid Online Similarity (GHOS) computes the overall hybrid similarity (OHS) of the instance x to the class c_j, with respect to the set of attributes $A = A_1, \ldots, A_n$:

$$\mathrm{GHOS}(x, y) = \sum_{A_i \in A} \mathrm{OHS}(x, y, A_i). \tag{2}$$

$$\mathrm{OHS}(x, y, A_i) = \begin{cases} s_c(x, y, A_i), \\ s_n(x, y, A_i), \end{cases} \tag{3}$$

where:
A_i is a categorical attribute.
A_i is a numerical attribute.

The meaning of $OHS\ (x, y, A_i)$, equation 3 had direct relationship with a relevant property of the OHS similarity immerse on the definition GHOS: OHS analyzes categorical and numeric attributes separately, without the need for codifying either of them.

For categorical attributes, s_c will return zero if the compared values of the samples are different or one of them is missing, and one if the compared values are equals, equation 5. On the other hand, for numeric attributes s_n, equation 5, uses the current standard deviation to determine if two values are close enough for considering them similar. The estimation of the standard deviation σ of numeric attribute A_i for class l_j is computed disregarding the instances with missing values for this attribute.

ISSN 2007-9737

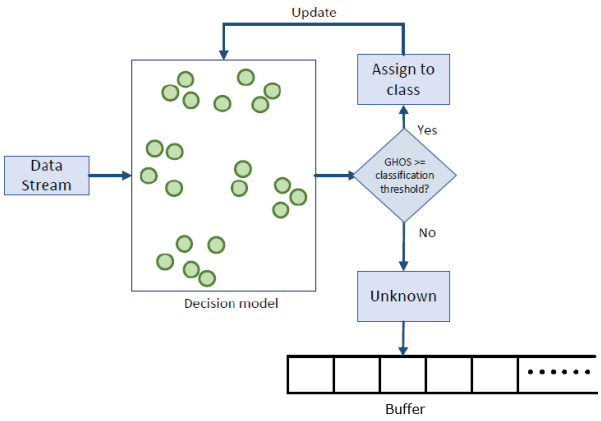

Fig. 6. Classification phase

Table 3. Initial parameters defined by the user

Initial Parameters				
A	B	C	D	E
100	20	2000	10	5

Update phase. NACOD will be classifying as new instances arrive, and then the classified instances will be used for updating the model.

Two possibilities exists: the first is that the instance belongs to a known class, and the other is that the instance does not belong to any of the existing classes. In the first situation, NACOD increases the number of instances seen so far of the corresponding class, as well as updates the maximum, minimum, sum of values and sum of squared differences for each numeric attribute, and the current values and occurrences for categorical attributes in the prototype of the corresponding class.

And in the second case, the instance will be stored by NACOD as a new class prototype; the sum of values and sum of squared differences for the new class will be initialized, as well as the values and appearances of the categorical values. NACOD pseudocode is presented in Algoritmic 1.

2.3 *Concept-Drift*

Given the non-stationary nature of data streams, the concept represented by the data that is being observed can change over time.

When this changes affect the relation between the input variables and the target variable, we are in the presence of *concept-drift* [11, 19, 12]. Formally, *concept-drift* between time point t_0 and time point t_1 can be defined as:

$$\exists X : p_{t_0}(X, y) \neq p_{t_1}(X, y), \tag{4}$$

where p_{t_0} and p_{t_1} denote the joint distributions at time t_0 and time t_1, respectively, between the set of input variables X and the target variable y. *Concept-drift* affects the decision boundaries and therefore it can reduce the performance of the learning algorithms if it is not correctly addressed. Figure 4 depicts *concept-drift*.

3 Proposed Solution

The proposal is divided into an offline phase and an online phase. During the offline phase (training) a clustering process is performed, therefore, Subsection 2.2.1 introduces the clustering algorithm used. In Subsection 2.2.3, the Silhouette coefficient was presented, which is used to determine the cohesiveness of a cluster.

The online phase uses the NACOD associative classifier to classify the arriving instances; the theory behind this classifier was presented in Section 2.2.4. Examples not described by the current model are marked as outliers and stored in a buffer to be used later to address one of the main challenges in data stream classification: *concept-drift*, explained in section 2.3, or perhaps discard them if they are outliers.

3.1 Problem Formalization

We first formalize the problem of our data sets for data stream classification here. A data stream D can be defined as an infinite sequence of multi-dimensional instances $x^1, x^2, \ldots, x^N$ arriving at time $t_1, t_2, t_3, \ldots, t_N$, where each instance is a vector of dimension n, denote by $x^i = \{x_1^i, x_2^i, \ldots, x_n^i\}$. Each instance x^i is associated with a class label y^i. Te class label $y^i \in Y$, where Y is the set of classes in the data stream $Y = \{c_1, c_2, \ldots, c_L\}$.

Algorithm 1 NACOD Classifier Algorithm

procedure NACOD(T, A, W, L, x)
  Inputs:
  T : Training set of data points with labels.
  A : Set of attributes describing each data point. $A = \{A_1, \cdots, A_n\}$
  W : Set of weights for attributes (optional). $W = \{w_1, \cdots, w_n\}$
  L : Set of possible class labels. $L = \{l_1, \ldots, l_k\}$
  x : Instance to classify.
  Training:
  Creating the archive.
  Compute and store the prototype of each label into a prototype set $P = \{p_1, \cdots, p_k\}$.
  For each numeric attribute and for each class, store its maximum and minimum values (max_i, min_i)
  For each categorical value and for each class, store its values and number of appearances.
  Classification:
  $l \leftarrow$ FindMostSimilarLabel(x, P)
  (Find the label of the prototype most similar to x)
  Updating:
  if x does not belong to any class in L **then**
    AddClassLabel(L, new label)
    UpdatePrototypeSet(P, x, new label)
    UpdateArchiveInfo(P, A)
  else
    UpdateClassInfoInArchive(P, x, L)
    UpdatePrototypeSet(P, x, GetLabelOf(x))
  end if
  Outputs:
  Class label assigned to x.
  return x
end procedure

As each instance arrives, we can collect p sequential instances into a data chunk, denoted as $D_k = \{d^{k,1}, d^{k,2}, \ldots, d^{k,t}, \ldots, d^{k,p}\}$, where $d^{k,t}$ indicates an instance that arrives at time t in data chunk k, namely $d^{k,t} = \{x^{k,t}, y^{k,t}\}$.

The training data set $D_1 = (x^{1,1}, y^{1,1}), (x^{1,2}, y^{1,2}), ..., (x^{1,t}, y^{1,t}), ..., (x^{1,p}, y1, p)$ is a set of p tuples where each $x^{1,t}$ is an input vector arriving at time t in first data chunk and $y^{1,t}$ is its target variable (class label).

3.1.1 Offline Phase

In this first phase, performed offline, that can be considered the training phase of the algorithm, a set of labeled instances will be needed to create the initial model.

As suggested in [1] an $InitNumber = 2,000$ was defined as the number of instances for the initial training set. In this stage the training data set will be divided in subsets $D_1, D_2, \ldots, D_S$ where D_k is the set of instances belonging to class c_j. A clustering algorithm is applied to each D_k to create q micro-cluster (MC) for each class. In this work, the k-means algorithm [17] will be used, as also suggested in [1]. Figure 5 depicts the general initial training process.

During the update step, which will further explained in Subsection 3.2.4, these initial micro-clusters will be adjusted to reflect the changes in the data stream. Each micro-cluster will be denoted by MC_j^i where j shows the class of the micro-cluster and i indicates the number of the micro-cluster in the class.

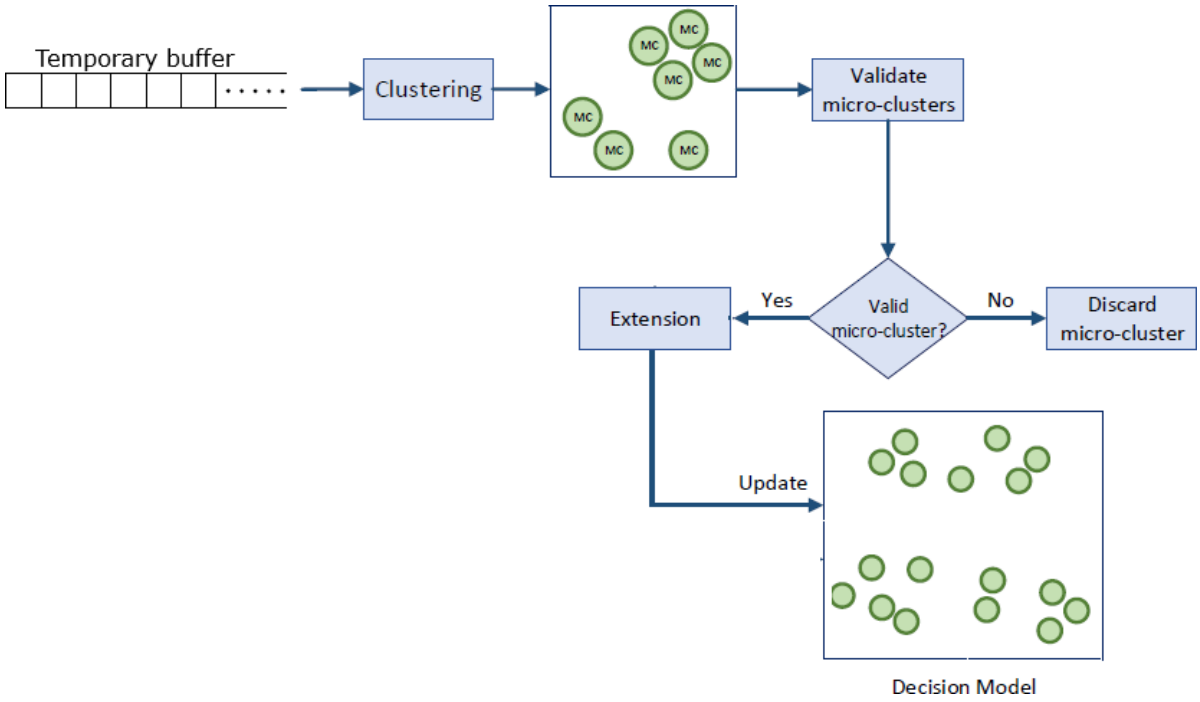

Fig. 7. *Concept-drift* detection process

Also the micro-clusters need to have an indicator of the timestamp when it was last updated. To this end, the index of the last aggregated instance will be used as the micro-cluster timestamp:

$$s_c(x, y, A_i) = \begin{cases} 0 & \text{if } \Big((x_i \neq y_i) \vee (x_i = '?') \\ & \vee (y_i = '?') \Big), \\ 1 & \text{otherwise.} \end{cases} \tag{5}$$

$$s_n(x, y, A_i) = \begin{cases} 0 & \text{if } \Big(\left(|x_i - y_i| > \sigma_j^i\right) \vee (x_i = '?') \\ & \vee (y_i = '?') \Big), \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

Each micro-cluster is described by 6 components $(N_j^i, LS, SS, SD, c_j, t_j^i)$, where N_j^i is the number of instances in the i^{th} micro-cluster of the j^{th} class; LS and SS are the linear and squared sum of the examples in the micro-cluster, respectively; SD is the sum of squared differences between the centroid of the micro-cluster and its instances; c_j is the label of the class to which the micro-cluster belongs; and t_j^i is the timestamp which value is the index of the last instance that updated the microcluster.

This representation is an extension of the one used in [8]. Using these measures it is possible to calculate other important metrics required for the implementation of this proposal, such as the standard deviation of each feature, the distance between micro-clusters, and the standard deviation of the distances between a centroid and its instances. To calculate the centroid of a micro-cluster the following equation was used, as proposed in [22]:

$$\text{Centroid}_{MC_j^i} = \frac{LS_j^i}{N_j^i}. \tag{7}$$

The equations used to calculated each of the other elements that describe a micro-cluster are the following:

$$LS_j^i = \sum_{x \in MC_j^i} x, \tag{8}$$

Fig. 8. Test-then-train or prequential with 4 blocks

$$SS_j^i = \sum_{x \in MC_j^i} (x)^2, \tag{9}$$

$$SD_j^i = \sum_{x \in MC_j^i} (x - \text{Centroid}_{MC_j^i})^2, \tag{10}$$

$$\sigma = \sqrt{\frac{SD_j^L}{N_j^L}}. \tag{11}$$

Other statistics must be calculated to use during the classification step. The NACOD classifier, uses standard deviation to calculate similarity of numeric features. Assuming the possible infinite length of a data stream, it is impossible to store all the instances that have been seen; instead summarise of the data are stored. Therefore, an estimation of the standard deviation will be calculated using the squared sum of differences. For this purpose, the squared sum of differences (SD) was calculated and stored for each micro-cluster.

3.2 Online Phase

Two possibilities exist when an instance arrives:

1. The instance is classified (correctly or incorrectly).
2. The instance is sent to a temporary buffer, because is not explained by the current model.

3.2.1 Classification

During this phase, data will be read in chunks of $2,000$ instances and the label of the new incoming instances of the data stream will be predicted. In this proposal, the NACOD algorithm [26] will be used. Unlike the original NACOD proposal [27], which only handles one standard deviation per class, our proposal uses several standard deviations per class, *i.e.* one for each micro-cluster that describes the class.

This, allows a better description of the distribution of the instances in the feature space. A classification threshold based on the GHOS operator was used to decide if a new instances should go through the classification process or should be sent to the temporary buffer. Several values were tested for this threshold, but the best results were obtained with values between $r/2$ and r, where r is the number of attributes.

The Global Hybrid Online Similarity (GHOS) is computed between the new instance and the centroids of all current micro-clusters. If the obtained value of GHOS is greater than or equal to the classification threshold proposed, the instance is classified and assigned to the class of the micro-cluster whose centroid has the maximum similarity with the instance and its information will be used to update the model. Otherwise, the instance will be store in a temporary buffer for further analysis. The general idea of the classification phase is shown in Figure 6.

3.2.2 Buffer Analysis

The instances that were not explained by the current model, *i.e.* instances stored in the temporary buffer, will be studied to determine if they are outliers or hints of the beginning of a *concept-drift*. In the second case, those instances will be used to model new micro-clusters that allow the model to incorporate this new knowledge. The process on *concept-drift* detection will be executed at the final of each data chunk of $2,000$ instances. Figure 7 shows the novel class detection process. Two possibilities exist when temporary buffer is analysed:

1. Single or multiple instances can be outliers.
2. A representative number of instances can be declared as a class extension, known as a *concept-drift*.

To determine which one of the previous cases is, is necessary to have cohesive sets of representative instances. To identify these sets, a cluster process will be applied to the temporary buffer when a considerable number of instances stored in it is reached.

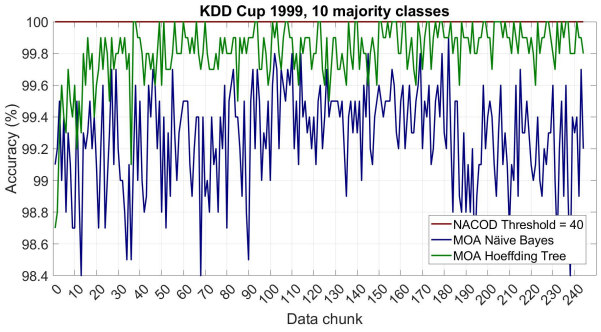

Fig. 9. KDD cup 1999 with its 10 majority classes experiment

Table 4. Accuracies averages

Using NACOD	Näive Bayes	Hoeffding Tree
100%	99.27%	99.81%
99.01%	90.02%	97.81%
70.44%	59.59%	67.78%

This number $(NumInstances)$ is a user-defined parameter of the algorithm, in the experimental section this number is called *representative number*. In this process, the *k-means* algorithm will be used again to generate these new clusters. To identify if a cluster is cohesive, a modified Silhouette coefficient will be used, as proposed in [8]:

$$\text{Silhouette} = \frac{b-a}{(a,b)}, \tag{12}$$

where b is the Euclidean distance between the new micro-cluster centroid and its closest micro-cluster centroid, and a represents the standard deviation of the distances between the instance in the new micro-cluster and its centroid. To determine if a micro-cluster is representative it has to contain a minimum number of instances.

If the new micro-cluster is valid, *i.e.* representative and cohesive, the distance between the centroid of the new micro-cluster MC_{new} and all the $MC's$ centers belonging to the decision model will be calculated. The new micro-cluster will be labeled as an extension of the same class of its closest micro-cluster, $MC_{closest}$, and its instances will be eliminated from the temporary buffer.

That means the decision boundary has changed, this explains why NACOD did not classify well. Otherwise, the temporary buffer is not modified and the instances are kept for further analysis. This may mean a *concept-drift* or *outliers*. To prevent a memory overflow, the user must define a limit number of observations to be stored in the temporary buffer.

3.2.3 Buffer Update

The temporary buffer needs to be updated: instances that have been too long in this buffer are removed. For this, the timestamp of the instances is also stored.

At the end of each data chunk, the temporary buffer is checked and the instances whose difference between their timestamp and the current timestamp is greater than the defined data chunk size are considered outdated instances and will be eliminated.

3.2.4 Model Update

When new instances $\tilde{x}$ arrive, the model needs to be updated. If an instance is explained by one of the micro-clusters, *i.e.* is correctly classified, the knowledge provided by this instance needs to be added to the corresponding micro-cluster. It means that when an instance is added to its corresponding cluster, implies that the micro-cluster statistics with greater similarity to the instance have to be updated.

Linear (LS) and squared sums (SS) are updated using equations 13 and 14, increase by one the number of instances in the micro-cluster, and the centroid needs to be recalculated using the new LS and N. The squared difference between the instance and the centroid is added to SD. The timestamp is set using the instance index. The statistics of the corresponding micro-cluster will be updated using the following equations:

$$LS_j^i = LS_j^i + \tilde{x}, \tag{13}$$

$$SS_j^i = SS_j^i + (\tilde{x})^2, \tag{14}$$

$$SD_j^i = SD_j^i + (\text{Centroid}_{MC_j^i} - \tilde{x})^2. \tag{15}$$

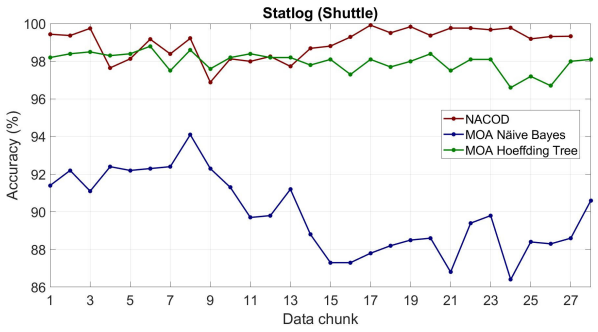

Fig. 10. Statlog (shuttle) experiment

Notice that before calculating equation 15, $\text{Centroid}_{MC_j^i}$ needs to be recalculated using the new value of LS_j^i. On the other hand, if the instance is not classified by any of the micro-clusters, it has to be added to the temporary buffer. At the end of each data chunk, the number of instances in the temporary buffer is checked to determine if it is time to analyse them and execute the *concept-drift* detection procedure.

Then a process of clustering using the *k-means* algorithm has to be executed and the Silhouette coefficient is calculated for each resulting cluster in order to assess if there are valid clusters to be added to the model. If a micro-cluster is valid, it has to be added to the model as an extension of a class. If the micro-cluster is not valid, it is discarded but the instances are kept in the temporary buffer.

4 Experimental Results

This Section explains what evaluation approach and measure metric were used. Also shows the results obtained from the experiments performed with the proposed solution presented in Section 3 and a discussion of them. Accuracy vs data chunks graphs to observe the performance of the algorithm as time goes by are provided.

Exist several approaches that are commonly used to estimate the performance of models when the data are not stationary. The objective of these approaches is to leave the last part of the data for testing and preserve the temporal order of observations although they do not make full use of all data.

In this article, we use Prequential approach, or Interleaved Test-Then-Train [7], because is one of the most common evaluation approaches in incremental data streams [11], in which first an observation (or a set of observations) is used to test and after, this observation is used to update and train the model. Figure 8 depicts Prequential approach, the data are split into 4 data splits.

As classification of data stream is not a static task, researchers evaluate over every data chuck how the classifier is working [4]. The measure implemented in this research is *accuracy*, is one of the easiest and simplest performance measure. It is determined as the ratio of correct predictions to the total number of test patterns [23].

$$\text{Accuracy} = \frac{\text{Correctly predicted values}}{\text{Total number of predictions}}. \quad (16)$$

As mentioned before, Table 3 refers to the parameters used in the experiments, those parameters must be defined by the user. The meaning of each column is explained next:

– Column A: The minimum number of instances required to start the analysis of the temporary buffer.

– Column B: The minimum number of instances to consider a MC representative.

– Column C: The Forgetting Factor, this number indicates how long an instance is kept in the temporary buffer. The forgetting process of the temporary buffer is performed by subtracting the Time Stamp of the instances in the temporary buffer to the current Time Stamp, if the result is greater that the Forgetting Factor the instance is discarded from the temporary buffer.

– Column D: The number of MC's to be generated from the temporary buffer.

– Column E: The number of MC's generated for each class in the training phase, this number refers to the *k* value for *k-means*.

The first experiment carried out was KDD Cup 1999 with its 10 majority classes. As the data set has $489,795$ instances, the data chunk length is $2,000$ and the first data chunk was used to train

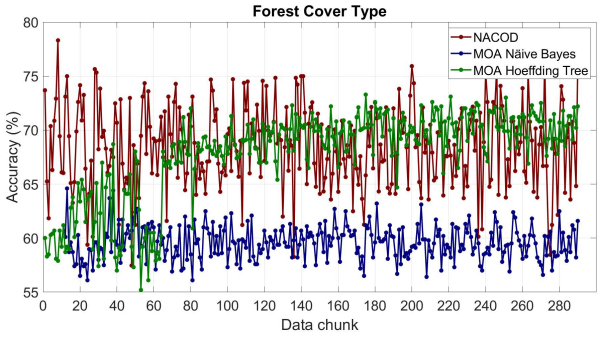

Fig. 11. Forest cover type experiment

the model, the experiment had in total 244 data chunks. Figure 9 shows 3 experimental results on this corpus. The algorithm proposed here is plotted in red, the Moa-Näive Bayes classifier is represented by the blue line and finally, the green line refers to the MOA-Hoeffding Tree.

Our proposal reaches in all the data chunks the 100% of the accuracy while Hoeffding Tree sometimes reaches the 100% accuracy and Näive Bayes never reached an 100% accuracy. It is important to mention that our proposal managed to detect *concept-drift* in 2 classes and therefore created 1 micro-cluster for the smurf class and 28 micro-clusters for the normal class.

Figure 10 corresponds to the second one experiment, which was Statlog (Shuttle), having $581,012$ observations, meaning in total 28 data chunks. The red line graphs refers to our proposal while the blue and green lines refer to the Näive Bayes and Hoeffding Tree classifiers respectively. Näive Bayes Classifier presents the lowest accuracies during the whole experiment; while out of 28 data chunks only in 6 data chunks it beats our proposal.

In Table 4 we observe the average accuracies of the whole experiments, for the second time our proposal manages to obtain the highest average accuracy compared to Näives Bayes and Hoeffding Tree. Here *concept-drift* was identified in 3 classes. 13, 6 and 4 MC's were added to the model for classes 1, 4 and 5 respectively. The main objective with this corpus was achieved with our proposal; as 80% of the data belong to class 1, the default accuracy is about 80%.

The aim here is to obtain an accuracy of 99 - 99.9%, we achieved 99.01%. Forest Cover Type data set was the last experiment carried out, its length is $581,012$ observations, if we consider each data chunk of length equal to $2,000$; it had 290 data chunks. The red line plotted corresponds to our proposal while the Näive Bayes classifier is plotted in blue and the Hoeffding Tree classifier in green.

As in the previous experiment, our proposal also identified *concept-drift* in 2 classes: class 1 and 2 ended up with 26 and 73 more MC's respectively. Näive Bayes classifier loses and it seems to be very close between Hoeffding Tree and our proposal but our proposal won again. In Table 11 the average accuracies of all the experiments are available.

5 Conclusion and Future Work

5.1 Conclusions

In this work an algorithm for classification of data streams using a classifier with an associative approach was designed, implemented and tested. This proposal combines a clustering algorithm, the Naïve Associative Classifier for Online Data (NACOD), a method to detected *concept-drift* and an updating process to incorporate the new discovered knowledge.

An extensive documentary research was carried out to study the issues related to the processing of massive data streams that change dynamically. As part of the study of current topics in data stream learning, a study of available real data sets was performed, in order to select those that were appropriate to perform the tests of the proposed algorithm. This led us to the selection of only 3 data sets for our tests.

The proposed solution was developed with the aim of classify, but also addresses *concept-drift* and noisy data. Our proposal beats the other classifiers in the evaluation measure. Because of the design of the solution, it is our belief that it is also capable of handling recurrent concepts, but given the scope, tests to verify this affirmation were not performed.

5.2 Future Work

The proposal presented in this paper fulfilled the main objective of classifying into data streams but several issues are still open to improvements to extend the scope of this proposal. Among them, we can mention the following:

– Use a different clustering algorithm to create the MC's. A clustering algorithm specially design for data stream, such as CluStream, could improve the results.
– Test the proposed solution with synthetic generated data sets.
– Design experiments to test the ability of the algorithm to handle recurring concepts and concept evolution.

Acknowledgments

The authors would like to thank the Instituto Politécnico Nacional (Secretaría de Investigación y Posgrado, Centro de Investigación en Computación and Centro de Innovación y Desarrollo Tecnológico en Cómputo), Consejo Nacional de Humanidades, Ciencia y Tecnología and Sistema Nacional de Investigadores for their support to develop this work.

References

1. **Aggarwal, C. C., Yu, P. S., Han, J., Wang, J. (2003).** A framework for clustering evolving data streams. Proceedings 2003 VLDB Conference, pp. 81–92. DOI: 10.1016/B978-012722442-8/50016-1.
2. **Bianchini, D., De-Antonellis, V., Garda, M. (2023).** A big data exploration approach to exploit in-vehicle data for smart road maintenance. Future Generation Computer Systems, Vol. 149, pp. 701–716. DOI: 10.1016/j.future.2023.08.004.
3. **Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B. (2010).** MOA: Massive online analysis. The Journal of Machine Learning Research, Vol. 11, pp. 1601–1604. DOI: 10.5555/1756006.1859903.
4. **Brzezinski, D., Stefanowski, J. (2017).** Prequential AUC: Properties of the area under the ROC curve for data streams with concept drift. Knowledge and Information Systems, Vol. 52, pp. 531–562. DOI: 10.1007/s10115-017-1022-8.
5. **Chen, D., Yang, Q., Liu, J., Zeng, Z. (2020).** Selective prototype-based learning on concept-drifting data streams. Information Sciences, Vol. 516, pp. 20–32. DOI: 10.1016/j.ins.2019.12.046.
6. **Chen, F. (2023).** Anomaly recognition method of network media large data stream based on feature learning. The International Conference on Cyber Security Intelligence and Analytics, pp. 20–28. DOI: 10.1007/978-3-031-31860-3_3.
7. **Dawid, A. P. (1984).** Present position and potential developments: Some personal views statistical theory the prequential approach. Journal of the Royal Statistical Society: Series A (General), Vol. 147, No. 2, pp. 278–290. DOI: 10.2307/2981683.
8. **de-Faria, E. R., Ponce-de-Leon-Ferreira-Carvalho, A. C., Gama, J. (2016).** MINAS: Multiclass learning algorithm for novelty detection in data streams. Data Mining and Knowledge Discovery, Vol. 30, pp. 640–680. DOI: 10.1007/s10618-015-0433-y.
9. **Domingos, P., Hulten, G. (2000).** Mining high-speed data streams. Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 71–80. DOI: 10.1145/347090.347107.
10. **Duda, R. O., Hart, P. E., Stork, D. G. (2000).** Pattern classification. Chapter Nonparametric Techniques, pp. 177–178.
11. **Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A. (2014).** A survey on concept drift adaptation. ACM computing surveys (CSUR), Vol. 46, No. 4, pp. 1–37. DOI: 10.1145/2523813.
12. **Halder, B., Hasan, K. A., Amagasa, T., Ahmed, M. M. (2023).** Autonomic active learning strategy using cluster-based ensemble classifier for concept drifts in imbalanced data stream. Expert Systems with Applications, Vol. 231, pp. 120578. DOI: 10.1016/j.eswa.2023.120578.
13. **Hewage, U. H. W. A., Sinha, R., Naeem, M. A. (2023).** Privacy-preserving data (stream) mining techniques and their impact on data mining accuracy: A systematic literature review. Artificial Intelligence Review, Vol. 56, pp. 10427–10464.

14. **Islam, M. Z., Lin, Y., Vokkarane, V. M., Yu, N. (2023).** Robust learning-based real-time load estimation using sparsely deployed smart meters with high reporting rates. Applied Energy, Vol. 352, pp. 121964. DOI: 10.1016/j.apenergy.2023.121964.
15. **Jasiński, M., Woźniak, M. (2022).** Employing convolutional neural networks for continual learning. International Conference on Artificial Intelligence and Soft Computing, pp. 288–297. DOI: 10.1007/978-3-031-23492-7_25.
16. **Korycki, Ł., Krawczyk, B. (2023).** Adversarial concept drift detection under poisoning attacks for robust data stream mining. Machine Learning, Vol. 112, No. 10, pp. 4013–4048. DOI: 10.1007/s10994-022-06177-w.
17. **MacQueen, J. (1967).** Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Vol. 1, No. 14, pp. 281–297.
18. **Mahajan, E., Mahajan, H., Kumar, S. (2024).** EnsMulHateCyb: Multilingual hate speech and cyberbully detection in online social media. Expert Systems with Applications, Vol. 236, pp. 121228. DOI: 10.1016/j.eswa.2023.121228.
19. **Mehmood, H., Khalid, A., Kostakos, P., Gilman, E., Pirttikangas, S. (2024).** A novel edge architecture and solution for detecting concept drift in smart environments. Future Generation Computer Systems, Vol. 150, pp. 127–143. DOI: 10.1016/j.future.2023.08.023.
20. **Mohanapriya, K., Sangavi, N., Kanimozhi, A., Kiruthika, V. R., Dhivya, P. (2023).** Optimized feed forward neural network for fake and clone account detection in online social networks. 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), pp. 476–481. DOI: 10.1109/ICSCDS56580.2023.10104616.
21. **Pala, O. (2024).** Assessment of the social progress on European Union by logarithmic decomposition of criteria importance. Expert Systems with Applications, Vol. 238, pp. 121846. DOI: 10.1016/j.eswa.2023.121846.
22. **Souza, V. M., Silva, D. F., Batista, G. E., Gama, J. (2015).** Classification of evolving data streams with infinitely delayed labels. 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), pp. 214–219. DOI: 10.1109/ICMLA.2015.174.
23. **Suryawanshi, S., Goswami, A., Patil, P. D., Mishra, V. (2022).** Adaptive windowing based recurrent neural network for drift adaption in non-stationary environment. Journal of Ambient Intelligence and Humanized Computing, Vol. 14, No. 8, pp. 1–15. DOI: 10.1007/s12652-022-04116-0.
24. **Tao, Z., Huang, S., Wang, G. (2023).** Prototypes sampling mechanism for class incremental learning. IEEE Access. DOI: 10.1109/ACCESS.2023.3301123.
25. **Vendramin, L., Campello, R. J., Hruschka, E. R. (2010).** Relative clustering validity criteria: A comparative overview. Statistical analysis and data mining: the ASA data science journal, Vol. 3, No. 4, pp. 209–235. DOI: 10.1002/sam.10080.
26. **Villuendas-Rey, Y., Hernandez-Castaño, J. A., Camacho-Nieto, O., Yañez-Marquez, C., Lopez-Yañez, I. (2019).** NACOD: A naïve associative classifier for online data. IEEE Access, Vol. 7, pp. 117761–117767. DOI: 10.1109/ACCESS.2019.2936366.
27. **Villuendas-Rey, Y., Rey-Benguría, C. F., Ferreira-Santiago, A., Camacho-Nieto, O., Yáñez-Márquez, C. (2017).** The naïve associative classifier (NAC): A novel, simple, transparent, and accurate classification model evaluated on financial data. Neurocomputing, Vol. 265, pp. 105–115. DOI: 10.1016/j.neucom.2017.03.085.
28. **Wu, Y. M., Chen, L. S., Li, S. B., Chen, J. D. (2021).** An adaptive algorithm for dealing with data stream evolution and singularity. Information Sciences, Vol. 545, pp. 312–330. DOI: 10.1016/j.ins.2020.07.010.
29. **Yao, B., Ling, G., Liu, F., Ge, M. F. (2023).** Multi-source variational mode transfer learning for enhanced PM2.5 concentration forecasting at data-limited monitoring stations. Expert Systems with Applications, Vol. 238, pp. 121714. DOI: 10.1016/j.eswa.2023.121714.

Article received on 07/11/2023; accepted on 14/12/2023.
**Corresponding author is Abril Valeria Uriarte-Arcia.*