Unsupervised Change Detection in Data Streams

Rosane M. M. Vallim [*], José A. Andrade Filho, Rodrigo F. de Mello,

André C. P. L. F. de Carvalho

Institute of Mathematical Sciences and Computation

University of São Paulo at São Carlos - Brazil

Jõao Gama

LIADD - INESC Porto

University of Porto - Portugal

Abstract

The ability to detect changes in the data distribution is an important issue in Data Stream mining. Detecting changes in data distribution allows the adaptation of a previously learned model to accommodate the most recent data and, therefore, improve its prediction capability. This paper proposes a framework for non-supervised automatic change detection in Data Streams called M-DBScan. This framework is composed of a density-based clustering step followed by a novelty detection procedure based on entropy level measures. This work uses two different types of entropy measures, where one considers the spatial distribution of data while the other models temporal relations between observations in the stream. The performance of the method is assessed in a set of experiments comparing M-DBScan with a proximity-based approach. Experimental results provide important insight on how to design change detection mechanisms for streams.

Keywords: change detection, unsupervised learning, data streams.

[*]Rosane M. M. Vallim (corresponding author) is with the Department of Computer Sciences, University of São Paulo, Av. Trabalhador São-carlense, 400, Centro, P.O.Box: 668, CEP: 13560-970, São Carlos, SP, Brazil (phone: +55 16 3373-8161; email: rvallim@icmc.usp.br).

1 Introduction

Many applications produce a continuous and time dependent series of data. In this scenario, the batch learning followed by many Machine Learning (ML) algorithms is not appropriate, since it is unpractical, and sometimes even impossible to store all examples in memory [1]. This new situation of high production of data has led to a new research area known as Data Stream mining [12], where, in opposition to conventional ML, algorithms are designed to learn online, processing examples as they become available and discarding them afterwards [9].

Another important distinction between the data stream learning scenario and the conventional, static, learning scenario is that, in the former, data characteristics can change over time because of the continuous operation of the system generating the data. In the conventional scenario, algorithms usually assume that data do not change with time and, therefore, after a learning model has been induced, it does not need to be modified in the future. On the other hand, an algorithm designed to learn over a Data Stream needs to take into account the non-static nature of data. Thus, models may need to be reinduced if changes occur in data characteristics.

Data change detection is an important issue for any activity in Data Stream mining, being frequently been approached under the names of behavior change detection and concept drift [10, 11]. For supervised learning, detecting changes in the behavior of data is crucial. Once the current data distribution is different from what has been seen in the past, the predictive model, induced from past data, will have its predictive performance decreased. When unsupervised learning is required, algorithms need to be able to introduce new data into the current model as to reflect the most recent behavior of the data stream, while maintaining, in an appropriated degree, previous knowledge.

This article is concerned with behavior change detection on unsupervised scenarios. Behavior change detection in such case is performed in a variety of ways [2, 17, 4, 16]. Among the most interesting approaches are the ones the apply clustering and novelty detection techniques [6, 19, 15, 3]. These algorithms are designed in order to continuously adapt clustering structures when variations in the behavior of the stream are observed. Together with a novelty detection mechanism, these techniques usually inform when a new event is considered to differ from the past observed behavior. However, some of these approaches are based on proximity

algorithms, what assumes data is organized in hyper-spherical clusters. Others are based on the assumption that a single outlier example in the stream is a novelty event, which, we believe, does not reflect a real behavior change. Moreover, many algorithms differentiate normal from novelty events based on static threshold values, what is not appropriate for streams of data changing over time.

This article proposes a new approach for automatic behavior change detection in unsupervised data streams based on clustering and novelty detection techniques. Our focus is on changes that alters the clustering structure by either changing the number of clusters, the proportion of examples per cluster or the arrival order of examples to the clusters. This approach is designed as to be independent of the number and format of clusters present in data, what motivated the application of a density based clustering algorithm inspired by DenStream [5]. The novelty detection step is based on the use of two different entropy measures. The spatial entropy considers the partition produced by the clustering algorithm and how examples are distributed among clusters. On the other hand, the temporal entropy models temporal relations between examples and, therefore, captures existing dependencies in the arrival of examples through time. Regardless of the entropy measure used, novelties are detected by changes in entropy levels that surpasses a dynamic calculated threshold, making the approach suitable for processing continuously changing streams, where it is unpractical to consider a constant threshold separating normal from novel events. Finally, a real behavior change is considered to be happening when novelties are detected in a sequence and, therefore, outliers are discarded.

This article is organized as follows: Section 2 presents a brief review of the literature on behavior change for unsupervised scenarios; Section 3 describes the main concepts of the DenStream clustering algorithm; M-DBScan, the proposed method for unsupervised behavior change detection, is explained in Section 4; experimental setup and analysis of the results obtained in a set of datasets are presented in Section 5 and 6, respectively; finally, Section 7 discusses the main conclusions of this work and suggests future research directions.

2 Change Detection in Data Streams: Related Work

Detecting when the data in a stream is changing and sub sequentially incorporating these changes in the learned model is not a trivial task [1]. Changes in an unsupervised stream can

be approached in a variety of ways.

The first approach is to process the raw data in order to build visualization tools that help a human expert analyze data evolution. Works under this approach usually apply visualization techniques in a user-defined time horizon, making it possible for an expert to analyze the evolution of data and find particular trends. Since the objective is to provide the user with a visual tool for following data evolution through time, this approach is not suitable for automatically detecting and reporting behavior changes. The work of Aggarwal [2] is one example of this approach. It presents a framework for the diagnosis of multidimensional data streams that uses a concept called velocity density estimation. Based on this concept, two types of profiles, a spacial velocity profile and a temporal velocity profile, are generated. These profiles can be then visualized and analyzed in a user-defined time horizon.

The second approach makes use of time windows in order to detect data changes. This approaches usually monitor data distribution inside a window of most recent points, or compare successive windows. Some examples are the Adaptive WINdowing algorithm (ADWIN) [4] and the Fixed Cumulative Windows Model algorithm (FCWM) [17]. The window approach works fine for big streams of data. However, there are applications where it is not possible to afford the delay in change detection introduced by time windows. An interesting alternative is the Page Hinkley Test (PHT) [16]. PHT can be applied incrementally over the stream and was designed to detect changes in the average of Gaussian signals. The drawback of PHT is that it uses a constant threshold to report on the occurrence of changes.

The third approach is related to the the task of clustering the unsupervised data, where the algorithm is designed so as to naturally accommodate changes in the clustering structure being built. In this case, new clusters can be created in order to represent the new data arriving in the stream or pre-existing clusters can have their features adjusted incrementally. In this situation, data changes are being introduced into the model, but there is no report on when the behavior of data is presenting such changes. Some examples of clustering algorithms that incrementally update their decision models based on new data are CluStream [1] , DenStream [5], Self-Organizing Maps (SOM) [14] and Recurrent Self-Organizing Maps (RSOM) [20].

CluStream [1], for example, is an algorithm divided in two phases. The first phase uses the concept of micro-clusters in order to summarize the data arriving at the stream. The

second phase applies the K-means algorithm [13] to the center of the micro-clusters, producing a high level clustering structure. DenStream [5] is another algorithm that is divided in two phases. Since this algorithm is the base for our work, it will be described in more details in Section 3. The SOM [14] algorithm, on the other hand, uses the neuron as the basic learning unity. SOM incrementally updates its neurons based on the examples being processed. This way, the network incrementally changes its topology characteristics in order to represent regions of interest in the domain of the problem being learned.

The forth way to approach changes in this scenario includes clustering approaches that are linked to a novelty detection mechanism. In this case, the objective is not only updating the clustering structure with the most up to date data, but also reporting on the occurrence of events that cause a variation on the current structure. The novelty detector checks how much the clustering structure has been changed with the arrival of a new example, for example, the creation of a new cluster to accommodate an incoming example. A novelty is declared when the level of change is higher than a pre-specified threshold. One interesting aspect of these methods is that they are incremental change detectors, i.e, consider only the current example in order to make a decision. Some algorithms that are based on this general idea are Growing k-means [6], OLINDDA [19], Sequential Leader Clustering (SLC) [7], Grow When Required Network (GWR) [15] and Self-Organizing Novelty Detection Neural Network (SONDE) [3].

The GWR [15] algorithm, for example, adds neurons to a neural network when the network does not match the data distribution anymore. The algorithm uses the information about how many times a neuron has been activated in order to detect novelties. If a neuron that has not been activated before, i.e, a newly created neuron, or a neuron that has not been activated for some period of time is selected for activation, a novelty event is considered to be occurring. SONDE [3], an algorithm that uses a neural network for data clustering, detects novelties based on estimations of Markov Chains (MCs) and measurements of entropy variation between consecutive MCs. At every new example arriving in the stream, the algorithm uses the information about the neuron activated by this example, together with information about the neuron selected for activation in the last time instant, to update a transition matrix between clusters. This matrix is then used to estimate the transition probabilities of a MC represented by the clusters. Shannon's entropy [18] is used to calculate an entropy level of the MC at every

time instant. The current entropy level is compared to the previous level and, if the difference is larger than a pre-specified threshold, a novelty event is declared.

The majority of algorithms that follows the forth approach present one of three characteristics:

1. Implement proximity-based clustering algorithms. This assumption reduces the application of these methods to real world data stream scenarios, since no assumption can be made on the clustering structure for most real data. Using a proximity-based clustering technique, one is assuming that the clusters have a particular structure, usually hyperspherical. If the clusters can have arbitrary shape, a proximity-based clustering algorithm may not be able to find good clusters in the data and, therefore, novelty detection will be harmed, resulting in poor performance of change detection;

2. Consider that a single outlier example in the stream is a novelty event, which, we believe, does not reflect a real behavior change. Behavior changes should be reported only when the behavior of data is effectively changing and not when an isolated event happens;

3. Assume a constant threshold in order to detect novelties, which, in many cases, is hard to establish *a priori*.

We believe that in order to design an unsupervised algorithm to automatically detect and report on data changes, it is necessary to consider both an arbitrary clustering structure of data and the definition of change as a sequence of consecutive novelty events. Moreover, the threshold separating normal from novel events should be dynamically adjusted. As far as our knowledge goes, ours is the first algorithm two present these characteristics simultaneously. Our method also approachs change detection from both the perspective of spatial distribution (spatial entropy measure) and temporal relationships among data (temporal entropy measure). This makes it possible to compare both types of measures and derive conclusions about the suitability of each measure to different scenarios. Thus, in the next sections we will present the basic concepts in which our approach is based, as well as a detailed explanation on how the algorithm detects changes on unsupervised data.

3 DenStream Clustering

DenStream [5] is a density-based clustering algorithm designed to discover clusters in a data stream scenario. It has an online phase where data is summarized and an offline phase of macro-clustering. The algorithm implements a strategy to find arbitrarily shaped clusters in a data stream without the need to specify the number of clusters *a priori*. It also includes a mechanism to distinguish normal examples from outliers.

In its online phase, the algorithm summarizes data using the concepts of potencial-micro-clusters (p-micro-clusters) and outlier-micro-clusters (o-micro-clusters), which are micro-clusters that contain outlier examples and are kept in the "outlier buffer". The main difference between them is a restriction on the weight w. Any micro-cluster c_p stores information like the weighted linear sum of points that fall into this micro-cluster ($\overline{CF^1}$) , the weighted squared sum of points ($\overline{CF^2}$) and a parameter value w that represents the weight of this micro-cluster. Micro-clusters are incrementally updated. Thus, if a new point p is merged into micro-cluster c_p, the information in c_p is updated by $c_p = (\overline{CF^1} + p, \overline{CF^2} + p^2, w + 1)$. Otherwise, if no point is merged into c_p in the time interval δt, $c_p = (2^{-\lambda \delta t} \cdot \overline{CF^1}, 2^{-\lambda \delta t} \cdot \overline{CF^2}, 2^{-\lambda \delta t} \cdot w)$, where $\lambda \in [0, 1]$. The algorithm also uses two input parameters, μ and β, where μ is the minimum weight of any micro-cluster and $\beta \in [0, 1]$ is the threshold of outlier relative to p-micro-clusters.

Everytime a new point p is available in the stream, the algorithm proceeds as follows:

1. Try to merge p into its nearest p-micro-cluster c_p. If r_p, the new radius of c_p is less or equal to ϵ, where ϵ is an input parameter, p is inserted into c_p;

2. Otherwise, try to merge p into its nearest o-micro-cluster c_o. If r_o, the new radius of c_o is less or equal to ϵ, p is inserted into c_o. Next, the algorithm verifies if w, the new weight of c_o, is greater than $\beta\mu$. If so, c_o has grown into a p-micro-cluster. Therefore, c_o is removed from the outlier buffer an a new p-micro-cluster is created by c_o;

3. Otherwise, create a new o-micro-cluster in the outlier buffer to accommodate p.

Since DenStream applies an exponential forgetfulness to its p-micro-clusters, the algorithm periodically checks the p-micro-clusters to make sure that all of them still hold to the assumption

that $w > \beta\mu$. If the weight of a p-micro-cluster is smaller than $\beta\mu$, the corresponding p-micro-cluster is removed. The o-micro-clusters are also checked to remove those that do not have potential to grown into a p-micro-cluster. The weight of o-micro-clusters is compared with a lower weight limit, ξ. If the weight is smaller than ξ, the o-micro-cluster is removed.

In the offline phase, everytime a clustering request arrives, DenStream applies a modified version of the DBSCAN [8] algorithm, using the p-micro-clusters as virtual points.

4 Density-Based Behavior Change Detection

This section describes the approach proposed in this article for automatic behavior change detection. Like other methods previously proposed for detecting changes in unsupervised scenarios, the proposed method is based on the application of a clustering algorithm to the incoming data, followed by a novelty detection step to detect behavior changes. The proposed method assumes that:

1. **The number of clusters in the stream is unknown:** since in a data stream examples arrive continuously, it is usually impossible to know before hand the number of clusters. This assumption calls for the application of clustering algorithms that estimate the number of clusters;

2. **The format of clusters is arbitrary:** it is not possible in many data stream applications to assume that clusters have a fixed and well-known spatial format, specially when data can change over time. This restriction favours algorithms that do not assume a specific cluster format;

3. **Outliers can occur and do not reflect a real change:** examples that clearly differ from the others can eventually occur in a data stream as a result of, for example, noise in the data collection process; we assume that these examples do not indicate real behavior changes, since they are isolated events. However, many clustering algorithms change their structures because of outliers and, when coupled with a novelty detection mechanism, will end up considering these examples as novelties. To avoid considering outliers as novelties, some type of outlier filter needs to be implemented by the algorithm.

The clustering procedure used in the proposed approach is an evolving method based on the DenStream [5] algorithm. As mentioned before, DenStream is able to find arbitrarily shaped clusters and does not require the number of clusters to be defined before hand. Besides, DenStream implements an efficient mechanism to eliminate outliers, preventing these examples from misleading the clustering structure. Some modifications to the original DenStream formulation were introduced in order to avoid error propagations from the clustering phase to the novelty detection phase. Novelty detection is performed by calculating an entropy level every time a new example is inserted in the clustering structure. Two different entropy measures are used, with different characteristics. Regardless of the entropy measure used, novelties are detected based on the history of entropy values observed during the processing of the stream, and, thus, the process is less dependent on pre-specified thresholds than other similar strategies. The approach proposed in this work will be referred as M-DBScan (from Micro-Clustering DBScan).

4.1 Clustering of incoming data

As M-DBScan is based on DenStream, the concepts of p-micro-clusters and o-micro-clusters are used and they store the same information as the original DenStream, i.e., $\overline{CF^1}$, $\overline{CF^2}$ and w. The micro-cluster structures are maintained incrementally. When a new point p arrives at the stream, if an existing p-micro-cluster c_p is selected for receiving p, the information in c_p is updated as $c_p = (\overline{CF^1} + p, \overline{CF^2} + p^2, w + 1)$. For all remaining p-micro-clusters, the values of $\overline{CF^1}$, $\overline{CF^2}$ and w are not updated. On the other hand, if no p-micro-cluster is selected to receive p, the outlier buffer is checked to see if there is any o-micro-cluster that can receive the example. If so, the values of $\overline{CF^1}$, $\overline{CF^2}$ and w are updated as for the p-micro-clusters. All remaining o-micro-clusters remain unchanged. If no o-micro-cluster is chosen, a new o-micro-cluster is created to receive p, with $w = 1$ as initial weight.

The algorithm uses the input parameters μ, β and ϵ, where μ is the minimum weight of a micro-cluster, $\beta \in [0, 1]$ is the threshold of outlier relative to p-micro-clusters and, finally, ϵ is the maximum boundary on the radius of a micro-cluster. The algorithm is detailed next.

Every time a new example p is available to the algorithm:

1. Try to merge p into its nearest p-micro-cluster c_p. If r_p, the new radius of c_p, is less than

or equal to ϵ, p is inserted into c_p.

2. Otherwise, try to merge p into its nearest o-micro-cluster c_o. If r_o, the new radius of c_o, is less than or equal to ϵ, p is inserted into c_o. Next, the algorithm verifies if w, the new weight of c_o, is larger than $\beta\mu$. If so, c_o has grown into a p-micro-cluster. Therefore, c_o is removed from the outlier buffer and a new p-micro-cluster is created by c_o.

3. Otherwise, create a new o-micro-cluster in the outlier buffer to accommodate p.

4. If T_p mod t_c, where t_c is the current time instant and T_p is given by Equation 1, check all o-micro-clusters in order to remove those that do not have potential to grow into a p-micro-cluster. For an o-micro-cluster c_o, if the weight of c_o is less than its lower limit of weight, denoted by ξ, then c_o is removed from the outlier buffer. The lower limit of weight is given by Equation 2, where t_o is the creation time of the o-micro-cluster.

$$T_p = \left\lceil \frac{1}{\lambda} log(\frac{\beta\mu}{\beta\mu - 1}) \right\rceil \tag{1}$$

$$\xi(t_c, t_o) = \frac{2^{-\lambda(t_c - t_o + T_p)} - 1}{2^{-\lambda T_p} - 1} \tag{2}$$

Equations 1 and 2 are used in the original DenStream algorithm. Also like DenStream, the algorithm proceeds applying the DBScan algorithm [8] on top of the micro-clustering structure, using the p-micro-clusters as virtual points. The macro-clustering procedure is called for every new example, differently from the original definition of DenStream where the macro-clustering is performed on an interval basis, where the interval is defined by the user.

By using this clustering procedure, no assumption needs to be done *a priori* on the number and formats of clusters present in data. Besides, the outlier buffer represents a filter that discards the majority of outliers in the stream. Other remaining outliers are taken care by the novelty detection procedure.

4.2 Novelty detection based on entropy levels

Entropy is a measure of the uncertainty associated with a random variable. The higher the value of the entropy measure, the higher the uncertainty about the outcome of the random variable, while small values of entropy stand for high degrees of certainty [18]. As an example, consider Figure 1, where $H(X)$ stands for the entropy associated with a coin flip and $P(X = 1)$ is the probability of the random variable X assuming value one. In this example, X is the outcome of a coin flip and can, therefore, assume two possible values. A value of zero means a tails outcome, while a value of one represents a heads outcome. If the coin is fare, the probability of the outcome being heads is 0.5, leading to the highest value of uncertainty about the outcome of a coin flip. On the opposite side, if the coin is unfair and, for example, only outcomes heads, the probability $P(X = 1)$ is equal to one and the entropy value is zero, meaning total certainty about the outcome of the coin flip.

This work proposes the use of two different entropy measures, one with a temporal bias and another with a more spatial bias. The first is based on the entropy measure used by the SONDE algorithm [3]. The second is based on the purity of different sets, like, for example, the purity of leaf nodes in a decision tree.

4.2.1 Temporal Entropy Measure

This measure estimates a Markov Chain (MC) from the clusters found in data at time instant t. Every cluster is considered a state for the MC and the transition probabilities between states are updated according to the cluster selected to received the current example and the cluster selected at the previous time instant.

Let M be a MC with N states and $p_{i,j}$ be the transition probability between states i and j. State i represents the cluster selected to receive the example at time instant $t - 1$, while state j represents the cluster selected at the current time instant t. In this work, the transition probabilities are estimated by an exponentially-weighted moving average, with a weighting factor $\eta_t \in [0, 1]$. The weighting factor has a control influence on the transition probabilities of the chain. The higher the value of η_t, the more the transition probabilities are affected by current updates. On the other hand, if a small value of η_t is used, the probabilities are changed more slowly, giving more weight to the past history of the chain. The transition corresponding

to states i and j in the chain are updated as in Equation 3.

$$p_{i,j} = \frac{(1 - \eta_t) \cdot p_{i,j} + \eta_t}{\sum_{k=0}^{N} p_{i,k}} \tag{3}$$

All other transitions $(a, b) \neq (i, j)$ at time instant t are updated as in Equation 4.

$$p_{a,b} = \frac{(1 - \eta_t) \cdot p_{a,b}}{\sum_{k=0}^{N} p_{a,k}} \tag{4}$$

The entropy level of a chain is calculated using its updated transition probabilities. This entropy level is based on Shannon's entropy [18] (Equation 5).

$$\mathbb{H}(M) = -\sum_{i=0}^{N} \sum_{j=0}^{N} P_{i,j} \cdot \log_2(P_{i,j}) \tag{5}$$

As an example, consider Figure 2 where a MC is represented at instants t and $t+1$. In this example, η_t is considered to be 0.05. The arrival of the example at time instant $t+1$ creates a new transition in the chain, while all other existing transitions are updated in order to reflect the change. The entropy of the chain is modified as a consequence of the updated transition probabilities.

Since the entropy level is calculated using the transition probabilities of a MC, it has a temporal bias, i.e, this measure is sensitive to changes in the order of occurrence of events. Therefore, the use of this measure allows for the detection of temporal relations between consecutive examples in the stream.

4.2.2 Spatial Entropy Measure

The second entropy measure considers the spatial distribution of data into clusters and estimates the proportion of examples belonging to each cluster during the stream's processing. The proportion of examples in a cluster i, denoted as P_i, is update every time a new example is inserted in the clustering structure. The cluster i that receives the example at time instant t has it's proportion updated as in Equation 6, where $\eta_e \in [0, 1]$ is a weighting parameter for the moving average.

$$P_i = (1 - \eta_e) \cdot P_i + \eta_e \tag{6}$$

All other remaining clusters have their proportions updated as in Equation 7, where $j \neq i$ represents the cluster index.

$$P_j = (1 - \eta_e) \cdot P_j \tag{7}$$

Afterwards, the entropy is calculated as in Equation 8

$$\mathbb{H}(G) = -\sum_{i=0}^{N} P_i \cdot \log_2(P_i) \tag{8}$$

where G is the set of clusters at the current time instant and N is the number of clusters.

Figure 3 presents a simple example where the proportion of examples per cluster at time instant t differs from the proportion at time instant $t + 1$, consequently modifying the entropy value. In this example a value of $\eta_e = 0.05$ is considered.

Different from the temporal entropy measure, the spatial entropy models changes in the concentration of examples per clusters. The spatial entropy is, therefore, not concerned with temporal dependencies that might be presented in data.

4.2.3 Using Entropy Levels to Detect Novelties

Whether the temporal or spatial measure is being considered, entropy levels are used in order to decide if the arrival of an example produces a significant variation in the clustering structure.

The entropy calculated at time instant t, expressed as $\mathbb{H}_t(\cdot)$, is compared to a dynamically calculated threshold τ. To define the value of τ, a moving average on historical entropy values Φ (Equation 9), and a moving standard deviation Ω (Equation 10), are calculated, where γ and $\delta \in [0, 1]$ are weighting factors.

$$\Phi_t = (1 - \gamma) \cdot \Phi_{t-1} \ + \ \gamma \cdot \mathbb{H}_t(\cdot) \tag{9}$$

$$\Omega_t = (1 - \delta) \cdot \Omega_{t-1} \ + \ \delta \cdot |\mathbb{H}_t(\cdot) - \Phi_t| \tag{10}$$

To estimated τ, we will assume a normal distribution on entropy values. Therefore, we will include a constant parameter θ that represents the number of standard deviations from the average to consider in order to define the threshold. Equation 11 presents the final calculation.

$$\tau = \Phi + (\Omega \cdot \theta) \tag{11}$$

If $\mathbb{H}_t(\cdot) > \tau$, then the entropy level $\mathbb{H}_t(\cdot)$ at time instant t is considered to diverge from the average entropy level observed, indicating the occurrence of a novel event.

This procedure for detecting novelties automatically adjusts the threshold separating normal from novel events during the processing of the stream. As a consequence, it is more appropriate for detecting novelties in changing streams than other approaches that assume a constant threshold defined by the user.

5 Experiments

This section presents the experiments carried out to access the performance of M-DBScan for change detection. For such, we evaluated our density-based approach for a set of data sets using both the temporal and spatial entropy measures. We compared the results obtained for each entropy measure with the results of a proximity-based method for clustering, using the same entropy measures for novelty detection. In the experiments, our focus was on detecting three different types of changes. The first two are changes that modify the clustering structure (structural change) by either changing the number of clusters or changing the format and proportion of examples per cluster. The last is a change in the arrival order of examples to belonging clusters (temporal change). This type of change is important for data streams with a time series behavior. Our hypothesis is that M-DBScan will present higher performance in the task of automatically detecting different types of changes when compared with the proximity based approach. The results of these experiments will also provide useful insights on the effectiveness of each entropy measure to detect different types of changes occurring in data.

5.1 Datasets used

Eight synthetical datasets were constructed to test the performance of the proposed approach. Each of these datasets has two numerical attributes and represents a data stream. Also, each dataset contains three behavior changes, occurring at the 1000^{th}, 2000^{th} and 3000^{th} examples. Further, one dataset was created using data from the 10% partition from the KDD99 dataset

on intrusion detection. This dataset contains 3 behavior changes, occurring at the 1000^{th} and 2000^{th} examples. Next, we present the main characteristics of each dataset.

1. FixedDenSpatial: presents four arbitrarily shaped clusters from the beginning to the end of the stream. Changes are of the structural type and modify the proportion of examples per cluster and the formats of clusters;

2. FixedDenTemporal: presents four arbitrarily shaped clusters from the beginning to the end of the stream. Changes are of the temporal type and modify the arrival order of examples to existing clusters;

3. FixedSphericalSpatial: presents four hyper-spherical clusters from the beginning to the end of the stream. Changes are of the structural type and modify the proportion of examples per cluster and the radius of the spheres that forms each cluster;

4. FixedSphericalTemporal: presents four hyper-spherical clusters from the beginning to the end of the stream. Changes are of the temporal type and modify the arrival order of examples to existing clusters;

5. VarDenAddCluster: presents a variable number of arbitrarily shaped clusters. Changes are of the structural type and modify the number of clusters by adding new clusters. The initial number of clusters is 4. A new cluster appears at the stream at every behavior change. Thus, at the end of the stream processing there are 7 clusters;

6. VarDenMergeCluster: presents a variable number of arbitrarily shaped clusters. Changes are of the structural type and modify the number of clusters by merging of two existing clusters into a single cluster (reduction in the number of clusters). The initial number of clusters is 5. Two clusters merge at every behavior change. Thus, at the end of the stream processing there are 2 clusters;

7. VarSphericalAddCluster: presents a variable number of hyper-spherical clusters. Changes are of the structural type and modify the number of clusters. It has the same characteristics as the VarDenAddCluster dataset, except for the format of the clusters;

8. VarSphericalAddSubCluster: presents a variable number of hyper-spherical clusters. Changes are of the structural type and modify the number of clusters by adding a new cluster followed by the disappearance of an existing cluster. The initial number of clusters is 4;

9. StreamKDD99: this dataset was constructed using data from 3 different attacks from the KDD99 dataset. The chosen attacks were Satan (probe attack), Smurf (DOS attack) and Neptune (dos attack). The dataset contains 3000 examples, divided in three sets of 1000 examples, each one containing data from one type of attack. Also, the data used was pre-processed in order to remove all categorical attributes, and, therefore, the remaining data contains 34 attributes of types numerical and binary.

Table 1 summarizes the characteristics of each synthetical dataset. For the StreamKDD99 dataset, some columns are marked as unknown. Since this dataset was constructed from data used primarily for classification tasks, it is hard to define correctly the number of clusters present in the data used in our experiments, as well as the format of this clusters. In what regards the type of change, we can safely assume that all the 3 changes can be categorized as both temporal and structural, because data from each attack is well differentiated from other attacks. Therefore, data from each type of attack will form a different set of clusters. When changing from one attack type to the other, changes will both alter the arrival order of examples to clusters and the proportion of examples per cluster.

5.2 Experimental Setup

M-DBScan uses a clustering step that is based on the DenStream algorithm. Therefore, on the macro-clustering phase, a density-based clustering algorithm is used. This algorithm is a variation of the DBScan algorithm that uses the centers of the p-micro-clusters, built on the previous micro-clustering phase, as virtual points that will be divided into clusters. Our hypothesis is that using a density-based clustering algorithm for the final clustering will produce better results when the data present clusters of arbitrary shapes. In order to evaluate this assumption, we implemented another clustering strategy that is proximity-based, i.e., builds spherical format clusters.

The proximity-based strategy used works as follows. Instead of using the DBScan variation on top of the p-micro-clusters, we applied the K-means [13] algorithm. The examples submitted to K-means were virtual points represented by the centers of the p-micro-clusters. Since each p-micro-cluster c_p has a center and a corresponding weight w_{c_p}, we created N_{c_p} copies of the example that represents c_p. The number of copies of c_p added to the set of virtual points is calculated as in Equation 12, where c is an integer constant.

$$N_{c_p} = \lceil w_{c_p} \rceil * c \tag{12}$$

In the experiments that follow, we set up the constant c with the same value as k, where k is the number of clusters for K-means. All examples in the set of virtual points where normalized in the interval $[0, 1]$ before being turned in to K-means. Regarding K-means centroid initialization, at the first execution of the algorithm we randomly selected k points in the interval $[0, 1]$. We then stored these initial values in order to initialize the centroids always at the same positions every time K-means is executed.

It is important to notice that this implementation allows us to make a fair comparison between density-based and proximity-based approaches, since both of them are based on the same micro-clustering process. Therefore, differences are a consequence of the macro-clustering step only. Moreover, K-means is a well known clustering algorithm, being frequently used as reference. For the sake of simplicity this approach will be called, from this point forward, as M-Kmeans (from Micro-Clustering Kmeans).

Our first set of experiments compares M-DBScan with M-Kmeans, using the temporal entropy as a measure for novelty detection, in the nine datasets presented. The second experiment compares both approaches using the spatial entropy measure. The parameters used by M-DBScan were $\mu = 10$, $\beta = 0.105$, $\lambda = 0.03$. This values configure the algorithm to promote o-micro-clusters to p-micro-clusters when the o-micro-cluster contains more than one example. It also establishes that, at every 102 examples, the algorithm will check for the possible removal of o-micro-clusters from the outlier buffer. The value of parameter ϵ, which defines a maximum boundary for the micro-clusters, varies with each dataset used and is expressed in Table 2. These values were established empirically for each dataset. For detailed explanation on the meaning of each parameter, see Section 4.1. Table 2 also presents the values used for k (the

number of cluster to M-Kmeans), since different values where used for different datasets. As a rule of thumb, we selected k as the initial number of clusters present in each dataset, except for StreamKDD99 where k was set to 4.

For the experiments using the temporal entropy measure we used $\eta_t = 0.005$, $\gamma = 0.05$, $\delta = 0.002$ and $\theta = 3$. The experiments using the spatial entropy measure applied $\eta_e = 0.005$, $\gamma = 0.05$, $\delta = 0.02$ and $\theta = 3$. These values were chosen in order to apply a more conservative approach to novelty detection, where the past history of events have a stronger influence on the definition of the threshold separating normal from novel events. In all experiments, examples were considered to arrive at the stream at every 1 second.

A behavior change is considered detected when more than one novelty occur in a sequence. These consecutive novelties are considered to represent a single change. Single novelties events are not considered as behavior changes.

In order to compare the results, three measures were used. The first one was the number of times a change was detected where it was actually not happening (false positive - FP). The second one was the number of times a change was detected, but with a significant delay (delayed detection - DD). A change detection is considered to be delayed (DD), if the delay is greater than 100 examples but lesser that 300 examples. Finally, the last measure was the number of times the change was not detected (non-detected change - ND). This measure includes the changes that were not detected at all and the changes that were detected with a delay greater than 300 examples. The lower the values in each one of these measures, the better the results.

Our hypothesis is that M-DBScan will produce better change detection results than M-Kmeans in the majority of the datasets used, specially on those containing clusters of arbitrary shapes and the ones with a changing number of clusters.

6 Results

The first experiment compares M-DBScan with M-Kmeans using the temporal entropy measure. Table 3 summarizes the results of this experiment. The results show that M-Kmeans was not able to detect 4 out of 27 existing behavior changes, while M-DBScan did not miss any

change. It was observed that M-DBScan was capable of finding a much more stable and close to optimum clustering of the points. This happened because M-DBScan can find an arbitrary number of clusters, while M-Kmeans produced a fixed number of clusters during the entire stream processing. This is particularly important when the number of clusters in the stream is changing, what is reflected in the ND results for M-Kmeans, where all the ND happened in datasets with a varying number of clusters.

Building a more stable clustering structure during the stream's processing also means building a more stable MC through time, which is directly linked to the FP occurrence. If the clustering structure is poor, there will be a high probability of occurrence of misleading transitions in the MC. These transitions will be reflected as false temporal novelties even for the simplest datasets, as, for example, FixedSphericalTemporal dataset. As can be seen in the results in Table 3, M-DBScan produced much less FP than M-Kmeans.

For the StreamKDD99 dataset both methods were able to detect the 2 existing behavior changes. This is due to the fact that data representing a particular network attack is reasonably well defined, presenting small variation. Moreover, different attacks produce data that significantly differ from data produced by other types of attacks. Regarding FP, M-DBScan detected the occurrence of one FP in the middle of the data representing the first attack (Satan). Taking a closer look to the results, we observed that this FP was actually a variation of the same attack, with different characteristics of what has been observed before in the stream. So the detection of this event would be actually desired, since it is a different behavior of the same type of intrusion.

Figures 4 and 5 present the temporal entropy variation with corresponding standard deviation and occurrence of novelties for dataset VarDenAddCluster. Behavior change detection is represented by peaks in the entropy curve that cause a sequence of novelties to occur.

Table 4 presents the delay for detection of each one of the three behavior changes. The delay for detecting a behavior change is important because, the sooner the change is detected, the sooner appropriate measures to deal with the results of this change can take place. As well as being able of detecting more behavior changes, M-DBScan also presented similar delays to M-Kmeans.

The next experiment compares M-DBScan with M-Kmeans using another entropy measure for novelty detection. This time, the spatial entropy measure was used. Table 5 presents the results obtained by both methods on the datasets used.

Results on Table 5 suggest that both methods were not able to detect a high number of behavior changes (ND). However, these results bring important insights on how the spatial entropy measure works. First of all, the results using both methods indicate that the spatial entropy measure was not appropriate when the change in data is of the temporal type (datasets FixedDenTemporal and FixedSphericalTemporal). This was expected, since the spatial measure does not measure the order of arrival of examples to clusters. It was also observed that the spatial entropy was very sensitive to the creation of new clusters, producing a high variation in entropy.

Results on the datasets with an increasing number of clusters suggest that M-Kmeans performance was worse than M-DBScan. While M-DBScan was able to accurately detect the majority of changes in these datasets, M-Kmeans or did not detect changes or detected them with a delay, as can be observed in Table 6. M-Kmeans also presented a high number of FP, while M-DBScan did not produce any FPs. Figures 6 and 7 present the spatial entropy variation with corresponding standard deviation and occurrence of novelties for dataset VarDenAddCluster.

For the StreamKDD99 dataset, again both methods performed similarly. The same conclusion draw from the temporal entropy applies to the spatial entropy.

Results in both the first and second experiments indicate that M-DBScan is more suitable to the task of detecting behavior changes than M-Kmeans. In order to better understand the influence of the two entropy measures used, we compared the results of M-DBScan gathered in the first and second experiments, i.e., using, respectively, the temporal and spatial entropy measures. Table 7 summarizes these results and simplifies comparisons between entropies.

As can be seen in Table 7, ND was reasonably higher in the spatial entropy. The majority of ND events happened in datasets containing temporal changes and changes in the format of clusters. As for the temporal changes, this was actually expected since the spatial entropy does not measure changes in the arrival order of examples. For the datasets containing changes in the format of clusters, we observed that ND occurrence was more related to the micro-clustering step than the novelty detection step. Since the micro-clustering step depends on input parameter ϵ to proceed, misclassifications of examples to wrong micro-clusters can occur if the value of ϵ is not appropriate. This happens because ϵ defines a maximum boundary for the radius of the micro-clusters (see Section 4.1). Errors in the micro-clustering phase propagate to the macro-clustering phase, resulting in a wrong clustering of the data as a whole. This errors can cause the spatial entropy not to detect variations in the proportion of examples per clusters, missing the true changes. The spatial entropy, however, was capable of detecting the majority of changes in the number of clusters occurring in the datasets.

As for the temporal entropy, this measure was more appropriate to detect the two types of changes where the spatial entropy fails. This is because these changes cause a modification in the transition probabilities of the Markov Chain (MC) that are almost instantaneous, even in some cases where the clustering structure is not optimal. Modifications in the MC are reflected in the temporal entropy measure, producing significant differences and consequently leading to novelty detection. The temporal entropy measure, however, presented a drawback in relation to the spatial entropy. Because the temporal entropy is less conservative in declaring novelties, the occurrence of FP in the temporal entropy was higher than in the spatial entropy. Since the spatial entropy requires more examples to produce a significant variation in entropy, it was not as affected by false positives.

7 Conclusions

In data stream mining, being capable of detecting and informing when the behavior of the stream is changing is an important consideration of many learning algorithms. In unsupervised learning, behavior change detection is performed mostly by the conjunction use of clustering algorithms and novelty detection strategies. However, some of these approaches are only capable of producing hypher-spherical clusters [19, 15, 3], while others consider a single event as a

novelty [15, 3]. Moreover, many of these approaches use a constant threshold separating normal from novel events [15, 7]. None of them is ultimately appropriate to detecting real behavior changes in arbitrary streams of data because: i) data can organize itself in clusters of arbitrary formats; ii) a true behavior change is characterized by a sequence of novel events; iii) the threshold separating normal from novel events should be dynamically adjusted during the stream's processing.

With this in mind, this article proposed a new method for detecting and informing when a behavior change is happening in the stream of data. This method uses a density-based clustering algorithm and a novelty detection mechanism based on entropy measures. Two different types of entropy measures were presented and explored in this work.

Results of experiments comparing our approach with a proximity-based method suggested the density-based approach proposed in this work was more appropriate for detecting different types of behavior changes in several datasets. Useful insight was also gathered about the two entropy measures used. The temporal entropy measure proved to be more appropriate for detecting the three types of changes presented in the datasets, while the spatial entropy is more effective for detecting changes in the number of clusters. The temporal entropy, however, is more sensitive to false positives than the spatial entropy when the number of examples in the stream is high. It was also observed that the delay for change detection was shorter for the temporal entropy than for the spatial entropy because changes produce modifications in the Markov Chain that are almost instantaneous.

Future work focus on the entropy measures used for novelty detection. First of all, our focus will be on how to make the temporal entropy measure less sensitive to false positives. For the spatial entropy measure, we intend to explore ways in order to make the measure appropriate to detect clusters changing in shape, but with no modification on the proportion of examples per cluster.

Finally, a complete system for behavior change detection will be developed, with the possibility of using both entropies for making decisions on novelty events. We believe that such a behavior changing detection technique can be useful to many applications ranging from sensor monitoring in industries, to network traffic monitoring, user profile creation, health care applications, and many others.

Acknowledgments

Rosane M. M. Vallim is funded by Fapesp, process number 2010/11250-0. João Gama is funded by the Portuguese Foundation for Science and Technology, project KDUS ref. PTDC/EIA-EIA/098355/2008. The authors would also like to thank the brazilian research councils CNPq and Capes.

References

[1] C. Aggarwal, J. Han, J. Wang, and P. Yu. A framework for clustering evolving data streams. In *Proceedings of the Twenty-Ninth International Conference on Very Large Data Bases*, pages 81–92. Morgan Kaufmann, 2003.

[2] C. C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 575–586. ACM, 2003.

[3] M. K. Albertini and R. F. Mello. A Self-Organizing Neural Network to Approach Novelty Detection. In *Intelligent Systems for Automated Learning and Adaptation: Emerging Trends and Applications*, pages 49–71. IGI Global, 2010.

[4] A. Bifet and R. Gavaldà. Learning from Time-changing Data with Adaptive Windowing. In *SIAM International Conference on Data Mining*, pages 443–448, 2007.

[5] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over and evolving data stream with noise. In *Proceedings of the SIAM Conference on Data Mining*, 2006.

[6] M. Daszykowski, B. Walczak, and D. L. Massart. On the optimal partitioning of data with K-Means, Growing K-Means, Neural Gas, and Growing Neural Gas. *Journal of Chemical Information and Computer Sciences*, 42:1378–1389, 2002.

[7] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

[8] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[9] J. Gama. *Knowledge Discovery From Data Streams*. CRC Press, 2010.

[10] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues. Learning with drift detection. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*, pages 286–295. Springer, 2004.

[11] J. Gama and P. P. Rodrigues. Data Stream Processing. In *Learning from Data Streams: Processing Techniques in Sensor Networks*, pages 25–38. Springer, 2007.

[12] J. Gama and P. P. Rodrigues. An Overview on Mining Data Streams. In *Studies in Computational Intelligence*, pages 29–45. Springer Berlin/Heidelberg, 2009.

[13] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[14] T. Kohonen. *Self-organizing maps*. Springer Verlag, 1997.

[15] S. Marsland, J. Shapiro, and U. Nehmzow. A self-organising network that grows when required. *Neural Networks*, 15(8–9):1041–1058, 2002.

[16] E. S. Page. Continuous Inspection Schemes. *Biometrika*, 41:100–115, 1954.

[17] R. Sebastião, J. Gama, P. Rodrigues, and J. Bernardes. Monitoring incremental histogram distribution for change detection in data streams. In *Knowledge Discovery from Sensor Data*, volume 5840 of *Lecture Notes in Computer Science*, pages 25–42. Springer Berlin / Heidelberg, 2010.

[18] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, 1948.

[19] E. Spinosa, A. C. P. L. F. de Carvalho, and J. Gama. Olindda: a cluster-based approach for detecting novelty and concept drift in data streams. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, pages 448 – 452, 2007.

[20] M. Varsta, J. D. R. Millán, and J. Heikkonen. A recurrent self-organizing map for temporal sequence processing. In *Proceedings of the 7th International Conference on Artificial Neural Networks*, pages 421 – 426. Springer Verlag, 1997.

Table 1: Datasets characteristics.

Name	**#Ex**	**#Dim**	**#Clusters**	**Format**	**Change Type**
FixedDenSpatial	4000	2	4	Arbitrary	Structural
FixedDenTemporal	4000	2	4	Arbitrary	Temporal
FixedSphericalSpatial	4000	2	4	Spherical	Structural
FixedSphericalTemporal	4000	2	4	Spherical	Temporal
VarDenAddCluster	4000	2	4 to 7	Arbitrary	Structural
VarDenMergeCluster	4000	2	2 to 5	Arbitrary	Structural
VarSphericalAddCluster	4000	2	4 to 7	Spherical	Structural
VarSphericalAddSubCluster	4000	2	4 to 6	Spherical	Structural
StreamKDD99	3000	34	Unknown	Unknown	Temporal and Structural

Table 2: Values of parameters ϵ and k used in the experiments.

Dataset	ϵ	k
FixedDenSpatial	80	4
FixedDenTemporal	80	4
FixedSphericalSpatial	100	4
FixedSphericalTemporal	100	4
VarDenAddCluster	60	4
VarDenMergeCluster	80	5
VarSphericalAddCluster	35	4
VarSphericalAddSubCluster	35	4
StreamKDD99	120	4

Table 3: Results for the temporal entropy measure: comparing M-DBScan with M-Kmeans. FP stands for the number of false positives, DD is the number of delayed detections and ND is the number of non-detected changes.

Dataset	**M-DBScan**			**M-Kmeans**		
	FP	**DD**	**ND**	**FP**	**DD**	**ND**
FixedDenSpatial	0	0	0	10	0	0
FixedDenTemporal	0	0	0	3	0	0
FixedSphericalSpatial	1	0	0	0	0	0
FixedSphericalTemporal	0	0	0	0	0	0
VarDenAddCluster	3	0	0	1	0	2
VarDenMergeCluster	1	0	0	5	0	0
VarSphericalAddCluster	2	0	0	1	0	1
VarSphericalAddSubCluster	0	0	0	0	0	1
StreamKDD99	1	0	0	0	0	0
Total	**8**	**0**	**0**	**20**	**0**	**4**

Table 4: Delay for detecting behavior changes with respect to the temporal entropy measure. C1, C2 and C3 represent, respectively, the first, second and third change happening in data. A dashed line represents non detection of the respective behavior change, while a *** signal represents a change that does not apply to this dataset.

Dataset	**M-DBScan**			**M-Kmeans**		
	C1	**C2**	**C3**	**C1**	**C2**	**C3**
FixedDenSpatial	4	3	9	10	3	9
FixedDenTemporal	0	8	6	1	4	8
FixedSphericalSpatial	4	3	9	4	3	9
FixedSphericalTemporal	0	1	3	0	1	3
VarDenAddCluster	10	11	20	14	–	–
VarDenMergeCluster	5	33	16	19	25	26
VarSphericalAddCluster	5	10	13	–	10	14
VarSphericalAddSubCluster	3	7	13	–	7	14
StreamKDD99	12	1	***	12	1	***

Table 5: Results for the spatial entropy measure: comparing M-DBScan with M-Kmeans. FP stands for the number of false positives, DD is the number of delayed detections and ND is the number of non-detected changes.

Dataset	**M-DBScan**			**M-Kmeans**		
	FP	**DD**	**ND**	**FP**	**DD**	**ND**
FixedDenSpatial	0	0	2	2	0	2
FixedDenTemporal	0	0	3	0	0	3
FixedSphericalSpatial	0	0	1	0	0	1
FixedSphericalTemporal	0	0	3	0	0	3
VarDenAddCluster	0	0	0	1	0	3
VarDenMergeCluster	2	0	1	4	1	1
VarSphericalAddCluster	0	0	0	3	0	2
VarSphericalAddSubCluster	0	0	0	3	0	1
StreamKDD99	1	0	0	1	0	0
Total	**3**	**0**	**10**	**14**	**1**	**16**

Table 6: Delay for detecting behavior changes with respect to the spatial entropy measure. C1, C2 and C3 represent, respectively, the first, second and third change happening in data. A dashed line represents non detection of the respective behavior change, while a *** signal represents a change that does not apply to this dataset.

Dataset	**M-DBScan**			**M-Kmeans**		
	C1	**C2**	**C3**	**C1**	**C2**	**C3**
FixedDenSpatial	–	4	21	–	–	–
FixedDenTemporal	–	–	–	–	–	–
FixedSphericalSpatial	–	4	9	–	9	9
FixedSphericalTemporal	–	–	–	–	–	–
VarDenAddCluster	44	58	20	–	951	–
VarDenMergeCluster	–	8	16	–	258	25
VarSphericalAddCluster	19	40	20	14	413	480
VarSphericalAddSubCluster	15	23	14	15	–	8
StreamKDD99	12	0	***	12	0	***

Table 7: Results for M-DBScan: comparing the temporal and spatial entropy measures. FP stands for the number of false positives, DD is the number of delayed detections and ND is the number of non-detected changes.

Dataset	**Temporal Entropy**			**Spatial Entropy**		
	FP	**DD**	**ND**	**FP**	**DD**	**ND**
FixedDenSpatial	0	0	0	0	0	2
FixedDenTemporal	0	0	0	0	0	3
FixedSphericalSpatial	1	0	0	0	0	1
FixedSphericalTemporal	0	0	0	0	0	3
VarDenAddCluster	3	0	0	0	0	0
VarDenMergeCluster	1	0	0	2	0	1
VarSphericalAddCluster	2	0	0	0	0	0
VarSphericalAddSubCluster	0	0	0	0	0	0
StreamKDD99	1	0	0	1	0	0
Total	**8**	**0**	**0**	**3**	**0**	**10**

Figures

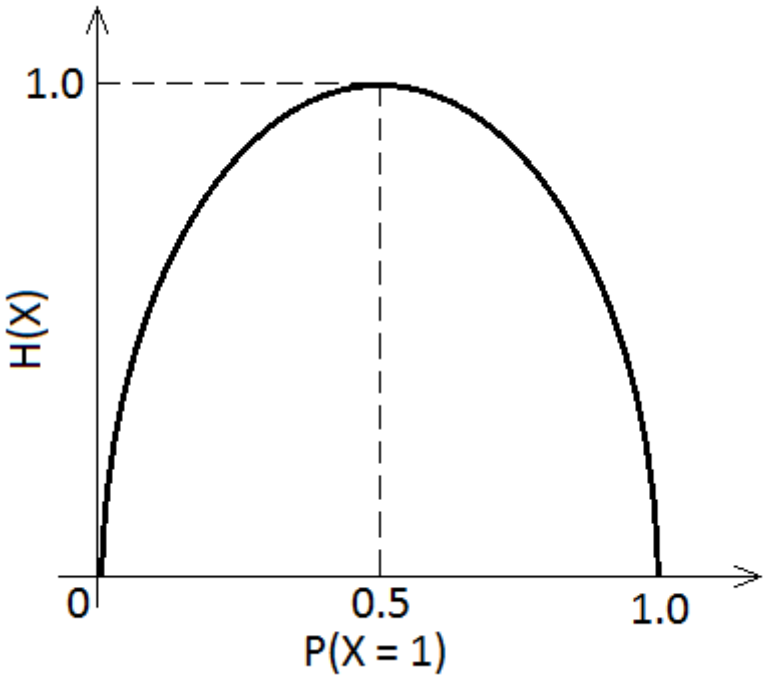

Fig. 1: Entropy of a random variable.

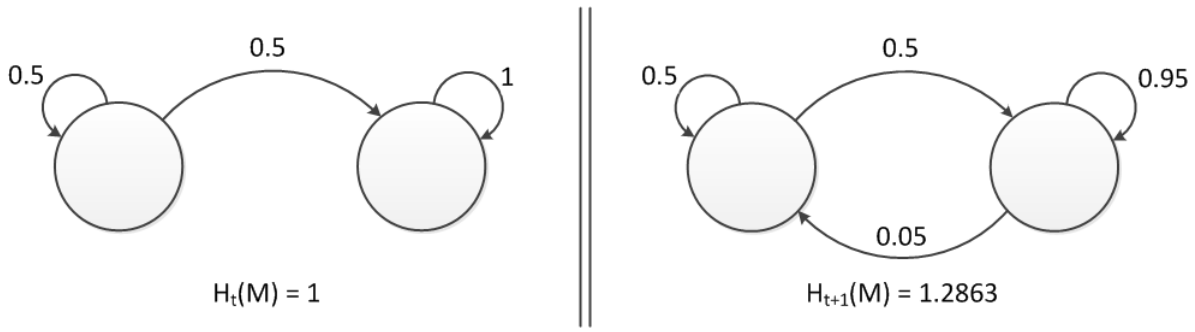

Fig. 2: Two consecutive states of a Markov Chain and respective entropy levels.

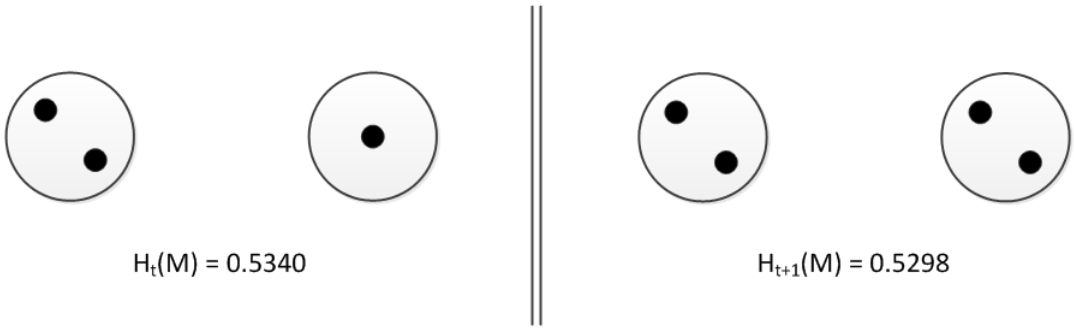

Fig. 3: Distribution of examples per cluster at two consecutive time instants with corresponding entropy values.

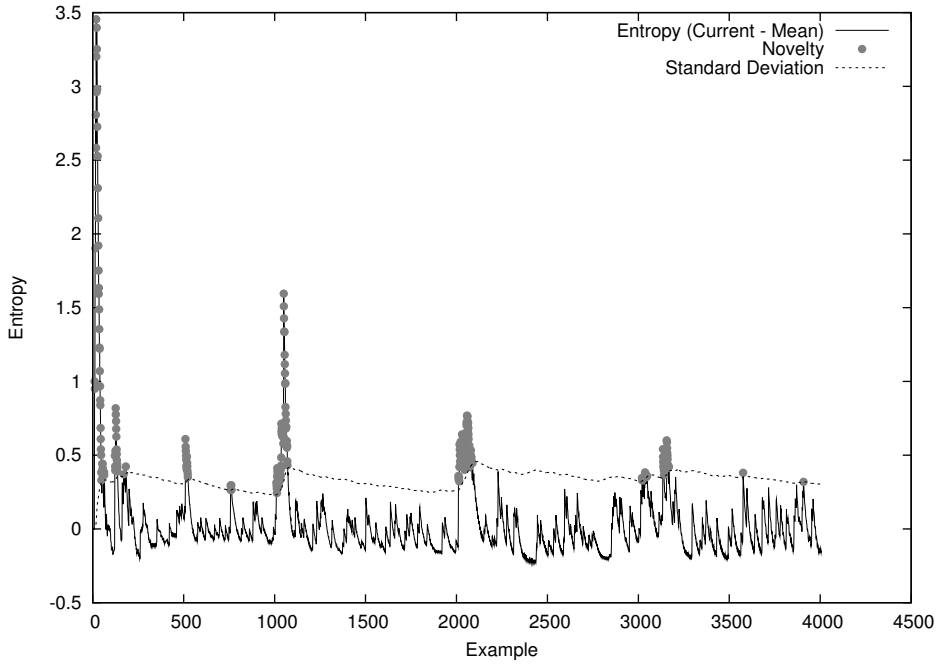

Fig. 4: Results using the temporal entropy measure for M-DBSCAN.

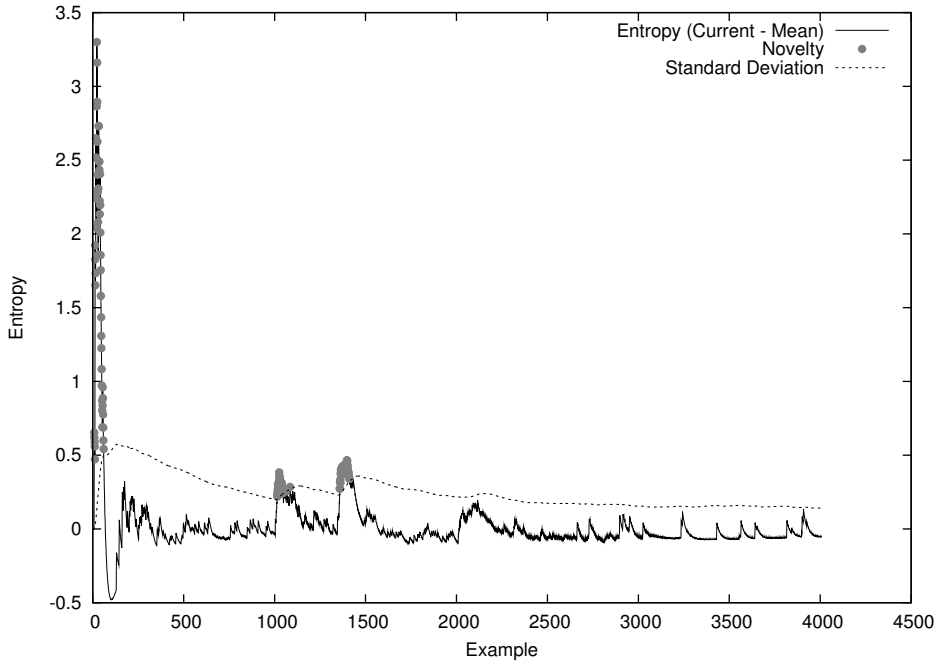

Fig. 5: Results using the temporal entropy measure for M-Kmeans.

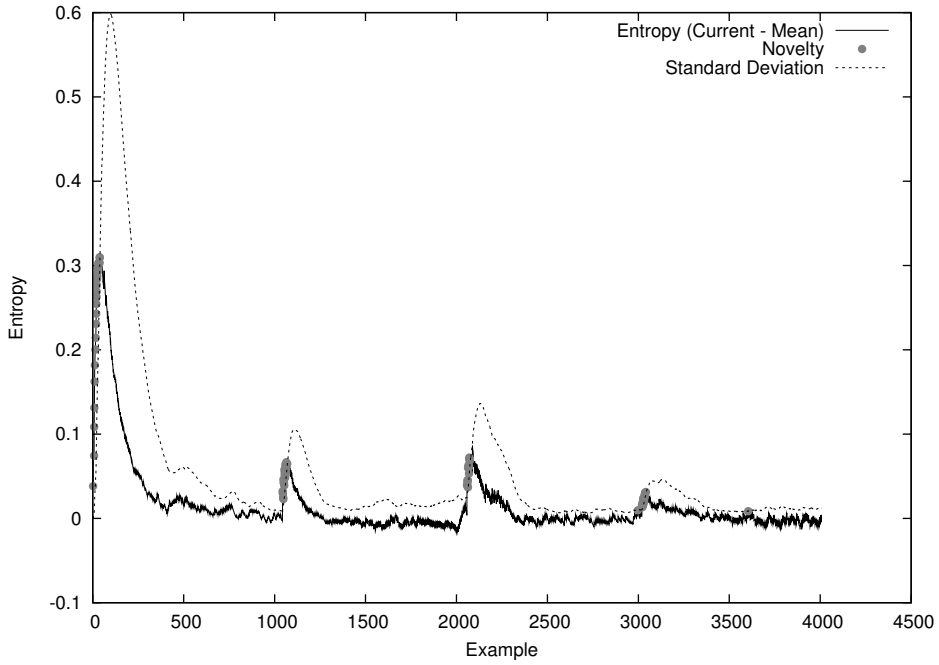

Fig. 6: Results using the spatial entropy measure for M-DBSCAN.

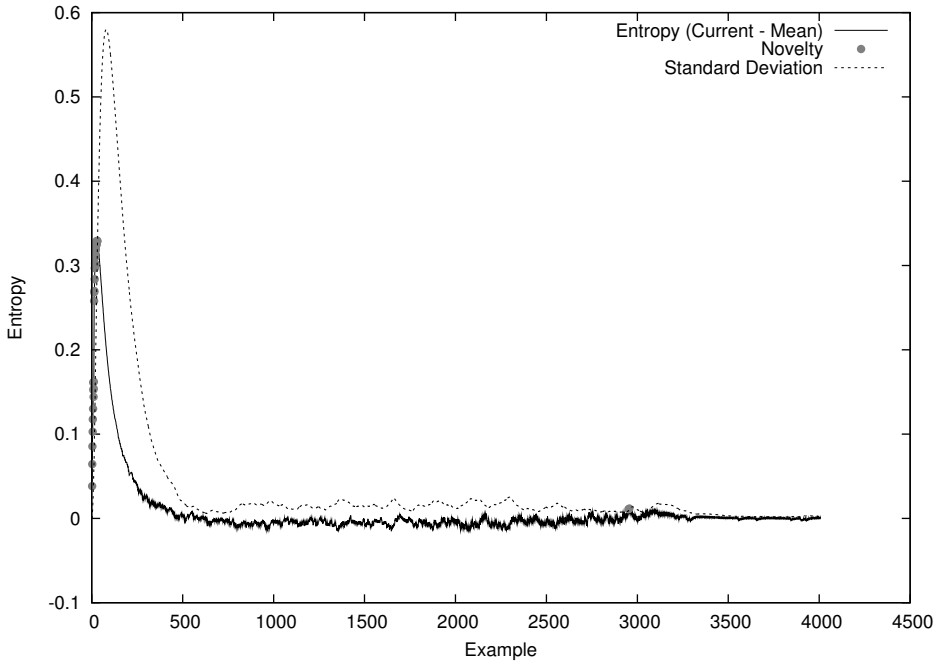

Fig. 7: Results using the spatial entropy measure for M-Kmeans.