Expert Systems With Applications 67 (2017) 228–238

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

An evolutionary algorithm for clustering data streams with a variable number of clusters

Jonathan de Andrade Silva[a,1,*], Eduardo Raul Hruschka[b,2], João Gama[c,3]

[a] Computer Science Department, The University of Mato Grosso do Sul (UFMS) at Ponta Porã, Brazil
[b] Computer Science Department, The University of São Paulo (USP) at São Carlos, Brazil
[c] INESC-TEC Department, The University of Porto, Portugal

ARTICLE INFO

Article history:
Received 2 May 2016
Revised 15 August 2016
Accepted 12 September 2016
Available online 22 September 2016

Keywords:
Evolutionary algorithms
Clustering
Data streams
Concept drift

ABSTRACT

Several algorithms for clustering data streams based on *k*-Means have been proposed in the literature. However, most of them assume that the number of clusters, *k*, is known a priori by the user and can be kept fixed throughout the data analysis process. Besides the difficulty in choosing *k*, data stream clustering imposes several challenges to be addressed, such as addressing non-stationary, unbounded data that arrive in an online fashion. In this paper, we propose a Fast Evolutionary Algorithm for Clustering data streams (FEAC-Stream) that allows estimating *k* automatically from data in an online fashion. FEAC-Stream uses the Page–Hinkley Test to detect eventual degradation in the quality of the induced clusters, thereby triggering an evolutionary algorithm that re-estimates *k* accordingly. FEAC-Stream relies on the assumption that clusters of (partially unknown) data can provide useful information about the dynamics of the data stream. We illustrate the potential of FEAC-Stream in a set of experiments using both synthetic and real-world data streams, comparing it to four related algorithms, namely: CluStream-OMR*k*, CluStream-B*k*M, StreamKM++-OMR*k* and StreamKM++-B*k*M. The obtained results show that FEAC-Stream provides good data partitions and that it can detect, and accordingly react to, data changes.

1. Introduction

Advances in both hardware and software have enabled large-scale data acquisition. Currently, enormous amounts of data are being collected in dynamic environments, at high speeds. Such data are usually referred to as *data streams*. A data stream is an unbounded, ordered sequence of objects that must be accessed in order and that can be read only once or a small number of times (Guha, Meyerson, Mishra, Motwani, & O'Callaghan, 2003). In recent years, data streams have attracted significant attention because of relevant applications, e.g., see Gama (2010); Lughofer, Macian, Guardiola, and Klement (2010); Mouchawe (2010); Zhang, Zhu, Shi, Guo, and Wu (2011).

Data streams must be intelligently transformed into meaningful and actionable information, which can then be used to enable more effective decision-making. To accomplish that goal, machine learning algorithms that are capable of continuous learning over time play a pivotal role. Specifically, data streams require learning algorithms that can adapt models, eventually forgetting data samples that become obsolete. In this context, incremental algorithms are of great relevance because they can avoid the computationally intensive task of re-training the whole model while accounting for dynamic patterns in the data that change over time. Additionally, the data stream must be processed in a single-pass-like manner, i.e., the data stream cannot be read again due to storage limitations. Usually, the data objects are discarded after being processed.

A useful form of analyzing data streams involves clustering (Aggarwal, Han, Wang, & Yu, 2004; Ailon, Jaiswal, & Monteleoni, 2009; Gama, 2010; Shindler, Wong, & Meyerson, 2011; Silva et al., 2012). The literature on clustering is very large. Of the many algorithms that is available is *k*-Means, which is very popular for data mining due to its simplicity, scalability, and empirical success in many real-world applications (Jain, 2009; Wu et al., 2007). Several *k*-Means variants have been proposed to address data streams, e.g., see Ackermann et al. (2012); Aggarwal, Han, Wang, and Yu (2003,2004); Guha et al. (2003);

* Corresponding author.
E-mail addresses: jonathan.andrade@ufms.br (J.d. Andrade Silva), erh@icmc.usp.br (E.R. Hruschka), jgama@fep.up.pt (J. Gama).
[1] Present Address: Campus of Ponta Porã, The University of Mato Grosso do Sul (UFMS-CPPP)
[2] Present Address: Department of Computer Science, The University of São Paulo (USP) at São Carlos, São Paulo, Brazil
[3] Present Address: Laboratory of Artificial Intelligence and Decision Support, University of Porto (UP), Porto, Portugal

http://dx.doi.org/10.1016/j.eswa.2016.09.020
0957-4174/

O'Callaghan, Meyerson, Motwani, Mishra, and Guha (2002). Despite the successful application of these algorithms to many real-world problems, they have a major limitation: the number of clusters, k, must be defined a priori.

From an optimization perspective, clustering can be formally considered to be a specific type of NP-hard grouping problem (Falkenauer, 1998). Evolutionary algorithms are meta-heuristics that are widely believed to be able to effectively produce sub-optimal solutions on NP-hard problems in a reasonable amount of time. Under this assumption, a large number of evolutionary algorithms for solving clustering problems have been proposed in the literature (see Hruschka, Campello, Freitas, and de Carvalho (2009) for an overview). More specifically, the *Fast Evolutionary Algorithm for Clustering* (FEAC) (Alves, Campello, & Hruschka, 2006) has shown to be especially efficient for automatically estimating k from data (Naldi, Campello, Hruschka, & Carvalho, 2011). However, this algorithm was not designed to address data streams. Aiming at circumventing such a limitation, we extend the FEAC in such a way that it can address data streams. The resulting algorithm is called the FEAC-Stream. To the best of our knowledge, this method is the first evolutionary algorithm for clustering data streams that addresses the estimation of k from the data.

In data stream scenarios, ideally the clustering algorithms should be able to update the data partition in an online fashion (Silva et al., 2012). This alternative can save computational resources when clusters do not change significantly over time. In order to determine if there is a change in the data partition, it is necessary to perform a change detection test. Among the alternatives in the literature, the Page–Hinkley (PH) Test (Mouss, Mouss, Mouss, & Sefouhi, 2004) is an efficient method to detect changes in the normal behavior of a process (Gama, Žliobaitė, Bifet, Pechenizkiy, & Bouchachia, 2014). Bearing this property in mind, we propose a change detection procedure that is based on the PH Test (Mouss et al., 2004). Specifically, the PH Test was adapted to detect whether the assignment of an object to the closest cluster increases the intra-cluster distances significantly.

The potential of the proposed FEAC-Stream is illustrated by comparing it to the framework proposed in de Andrade Silva and Hruschka (2011), which is based on three state-of-the-art algorithms for clustering data streams, namely, Stream LSearch (O'Callaghan et al., 2002), CluStream (Aggarwal et al., 2003), and StreamKM++ (Ackermann et al., 2012), combined with two algorithms for estimating the number of clusters, which are Ordered Multiple Runs of k-Means (OMRk) (Naldi et al., 2011) and Bisecting k-Means (BkM) (Steinbach, Karypis, & Kumar, 2000).

The remainder of this paper is organized as follows. In Section 2, we briefly review related approaches. Section 3 presents the proposed evolutionary algorithm for clustering data streams (FEAC-Stream). Experimental results are reported in Section 4. Finally, Section 5 concludes the paper.

2. Related Work

In general, the data stream clustering problem is defined as to maintain continuously consistent good clustering of processed objects using a small amount of memory and time (Guha et al., 2003). Ideally, the algorithms should incrementally process the data objects, rapidly detect and react to cluster evolution, provide a model representation that does not grow with the number of objects processed and handle outliers (Silva et al., 2012). Bearing these issues in mind, several clustering algorithms have been proposed in the literature (Ackermann et al., 2012; Aggarwal et al., 2003; 2004; Ailon et al., 2009; Broder, Garcia-Pueyo, Josifovski, Vassilvitskii, & Venkatesan, 2014; Cui et al., 2014; Guha et al., 2003; Hadian & Shahrivari, 2014; O'Callaghan et al., 2002; Shindler et al., 2011). Not surprisingly, most of them are based on k-Means and its variants. For example, in O'Callaghan et al. (2002), the authors proposed the Stream LSearch algorithm. This algorithm uses the divide and conquer strategy, which divides the data streams into chunks of data and, then, discovers clusters in each of these chunks and, finally, finds clusters from the centroids of each chunk. Aggarwal et al. (2003) point out that Stream LSearch is implemented as a continuous version of k-Means and assumes that the clusters are to be induced over the entire data stream. One of the most influential clustering data stream algorithms is the CluStream (Aggarwal et al., 2003). This algorithm considers that the data evolves over time and allows the exploration of clusters over different portions of the stream. The clustering process is divided into two components: (i) an online component that summarizes the data stream with a specific data structure (micro-clusters) and (ii) an offline component that uses these micro-clusters to induce clusters via a variant of k-Means. The authors in Ackermann et al. (2012) proposed the StreamKM++ algorithm, which summarizes the data stream by extracting a small set of objects (coreset) (Agarwal, Har-Peled, & Varadarajan, 2004; Bădoiu, Har-Peled, & Indyk, 2002) and finds k clusters with the k-Means++ algorithm. The k-Means++ can be viewed as a seeding procedure for the k-Means. Despite the successful application of these algorithms to many real-world problems, they have a major limitation: the number of clusters, k, must be specified in advance by the user. We note that some algorithms assume that the value of k is not required as input but a careful look reveals that another indirect parameter that controls it is required in advance. This is actually much worse than specifying the number of clusters (which is a more intuitive parameter to be set by the user) in advance — *e.g.,* setting the radius of clusters or their minimum number of points.

A few algorithms are able to estimate the number of clusters from data (Albertini & de Mello, 2013; de Andrade Silva & Hruschka, 2011; Beringer & Hüllermeier, 2006; Lughofer, 2012). In particular, a fuzzy c-means algorithm was proposed by Beringer and Hüllermeier (2006) where the k value is adjusted by choosing the best value between $k+1$ and $k-1$ according to a variant of the Xie–Beni index (Xie & Beni, 1991). In Lughofer (2012), split-and-merge strategies were proposed to estimate the number of clusters by increasing or decreasing the value of k according to threshold values. In Albertini and de Mello (2013), the Ordered Multiple Runs of k-Means algorithm is combined with Markov Models to estimate the number of clusters. The algorithms proposed in de Andrade Silva and Hruschka (2011) extend three state-of-art k-means based algorithms to estimate the number of clusters. Basically, the algorithms for clustering data streams can be described by means of two components, namely: a *streaming* step and a *clustering* step. To estimate the number of clusters from the data, the framework proposed in de Andrade Silva and Hruschka (2011) uses an intermediate $\hat{k}$-step, as illustrated in Fig. 1. More precisely, the *streaming* algorithm creates and keeps updated stream summary data structures whose size is smaller than the whole data stream (Han & Kamber, 2000). After building this summary, the k value is estimated via well-known algorithms for estimating the number of clusters from *steady-state* datasets, namely, Ordered Multiple Runs of k-Means (OMRk) (for a recent reference, see Vendramin, Campello, & Hruschka (2010)) and Bisecting k-Means (BkM) (Steinbach et al., 2000). Then, not only the estimated number of clusters but also the obtained cluster centers serve as inputs to the *clustering step*.

The framework illustrated in Fig. 1 can make use of a variety of algorithms. Among the available alternatives, we chose the popular CluStream (Aggarwal et al., 2003) and StreamKM++ (Ackermann et al., 2012). In the following, we address how they can be adapted to be used in the framework of Fig. 1 with respect