Expert Systems With Applications 91 (2018) 187–197

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Detection of evolving concepts in non-stationary data streams: A multiple kernel learning approach

Sajjad Kamali Siahroudi, Poorya Zare Moodi, Hamid Beigy*

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 30 April 2017
Revised 8 August 2017
Accepted 16 August 2017
Available online 24 August 2017

Keywords:
Novel class detection
Data stream
Concept drift
Data stream classification

ABSTRACT

Due to the unprecedented speed and volume of generated raw data in most of applications, data stream mining has attracted a lot of attention recently. Methods for solving these problems should address challenges in this area such as infinite length, concept-drift, recurring concepts, and concept-evolution. Moreover, due to the speedy intrinsic of data streams, the time and space complexity of the methods are extremely important. This paper proposes a novel method based on multiple-kernels for classifying non-stationary data streams, which addresses the mentioned challenges with special attention to the space complexity. By learning multiple kernels and specifying the boundaries of classes in the feature (mapped) space of combined kernels, the required amount of memory will be decreased. These kernels will be updated regularly throughout the stream when the true labels of instances are received. Newly arrived instances will be classified with respect to their distance to boundaries of the previously known classes in the feature spaces. Due to the efficient memory usage, the computation time does not increase significantly through the stream. We evaluate the performance of the proposed method using a set of experiments conducted on both real and synthetic benchmark data sets. The experimental results show the superiority of the proposed method over the state-of-the-art methods in this area.

1. Introduction

In data streams mining, we are faced with challenges that merely or even not addressed in the traditional data mining techniques. The major challenges in data stream mining are *infinite length, concept-drift, recurring concepts* and *concept-evolution*. The first one with regards to storage limitation of current computation devices imposes a restriction on the amount of memory usage. Concept-drift means changes in the underlying distribution of data (Khamassi, Sayed-Mouchaweh, Hammami, & Ghedira, 2016; Lughofer, Weigl, Heidl, Eitzinger, & Radauer, 2016; Shaker & Lughofer, 2014; Sobhani & Beigy, 2011), which results in a decrease in the accuracy of the constructed model by the passage of time (Dehghan, Beigy, & ZareMoodi, 2016; Souza, Silva, Gama, & Batista, 2015). Recurring concepts refer to classes that appear in the stream, then disappear for a relatively long time-span, and then reappear again (Hosseini, Ahmadi, & Beigy, 2013; Hosseini, Gholipour, & Beigy, 2016). Concept-evolution means the emergence of novel classes through the stream which we are not aware of them at the beginning. Concept-evolution is different from novelty detection in several aspects (ZareMoodi, Beigy, & Siahroudi, 2015). The goal of novelty detection methods in non-stream data are detecting abnormal data points as novelty and the concern is not to find a model to classify these novel data points (Angelov, Filev, & Kasabov, 2010; Kasabov, 2007; Lughofer, 2011); while in the data stream that concern is detecting novel classes and building a model for classifying the future data. Existing or known classes are those that learner has been observed their instances in the previously seen chunks. Learner should distinguish between recurring and novel classes. Regardless of the ever-increasing application of data stream mining in real world applications, there are just a few number of researches which address the concept-evolution issue.

In the supervised classification of non-stationary data streams, learners follow a general work-flow. These methods utilize the first thousands points of the stream to build an initial model in a phase called *offline* or *warm-up* phase. After this phase, the online phase begins in which the learner classifies each newly arrived instance. Since detection of concept-evolution requires analyzing a group of instances, in the most of novel class detection methods it is assumed that instances come in groups called *chunks*. Learner first classifies the instances of the newly arrived chunk, then receives their true labels and finally updates its model based on these true labels. This procedure continues until the end of the stream.

* Corresponding author.
E-mail addresses: shak16@mails.tsinghua.edu.cn (S.K. Siahroudi), poorya.zaremoodi@monash.edu (P.Z. Moodi), beigy@sharif.edu (H. Beigy).

http://dx.doi.org/10.1016/j.eswa.2017.08.033
0957-4174/

Before declaring a subset of a newly arrived chunk as a *novel class*, two conditions should be verified (ZareMoodi et al., 2015). First, instances of a novel class are more similar together rather than to instances of the known classes. This condition is based on the *cluster assumption*. Second, if the number of instances in the candidate subset (novel class candidates) is less than a pre-specified threshold, denoted by q, then these instances are *outliers* of the known classes. Several researches have been conducted in this area, however their methods are not memory efficient, and due to infinite-length of data streams, their performance degrades gradually throughout the stream.

In this paper, we propose a novel method based on multi-kernel approach to address the mentioned challenges for detecting novel classes in non-stationary data streams. The proposed method learns multiple kernels and determines the boundary of each class in the feature space. After arrival of a new chunk, each point of the chunk is classified as a member of the previously known classes or as an instance of a novel class. A new instance is classified based on its distance to the previously known classes in the feature space. The algorithm keeps a summary of previously seen instances instead of storing all the previously seen instances, which reduces the required amount of memory. After receiving the true labels of each chunk, the proposed method updates its model based on the true labels. The experimental results on some benchmark data sets show that the proposed method outperforms some of the state of the art methods for both the performance and the space complexity.

The rest of this paper is organized as follows: Section 2 dedicated to discuss related work. The proposed method is introduced in Section 3. Section 4 presents the results of experiments. This paper culminates with conclusions and future works in Section 5.

2. Related work

In this section, we give a brief review of the related works for detection of novel classes in data stream. One of the first well-known methods is ECSMiner (Masud, Gao, Khan, Han, & Thuraisingham, 2011). It uses an ensemble of classifiers equipped with decision boundaries. Decision boundary of a class shows the region that the decision of the associated classifier is valid and instances located outside of the boundary cannot be classified by that classifier. Classification of newly arrived instances is done in two phases. In the first phase, location of instances are specified; and instances located within the boundaries are classified as members of the given class and instances out of boundaries classified as novel class candidates. Then, they are moved to a short-time memory for further analysis. In the second phase, ECSMiner calculates q-neighborhood silhouette coefficient (q-NSC) to distinguish among instances of an emerged novel class with the outliers of existing classes. One major drawback of ECSMiner is forgetting recurrent classes. The reason lies behind its utilization of multi-class classifiers and elimination of obsolete classifiers based on their overall accuracy on the recent chunks. Thus, it is probable that all classifiers detecting a specific class are omitted and ECSMiner will miss-classify the instances of that class as *novel*.

Several methods have been proposed to improve the performance of ECSMiner. Multi class miner (MCM) (Masud et al., 2013) is an extension of ECSminer, which is designed to handle feature-evolving data streams by utilizing a feature set homogenization technique. Al-Khateeb et al. proposed CLAM (Al-Khateeb et al., 2012) to overcome the forget-recurring-classes issue in ECSMiner. It keeps an ensemble of one-class classifiers for each existing class instead of one ensemble of multi-class classifiers for all classes. Also, an updating procedure is designed for memorizing any of existing classes. In this mechanism, a classifier is eliminated when the corresponding ensemble is full and a newly trained classifier should be added. However, sometimes it is necessary to forget some classes due to the change in the underlying distribution of data. Other steps of this algorithm are similar to ECSMiner.

These methods have two drawbacks. First, they may declare emergence of a novel class with the number of instances below than the given threshold; specially when more than one novel class are emerged in a chunk (ZareMoodi et al., 2015). Second, at most one classifier can be eliminated per each ensemble while adaptation to sudden or rapid concept-drift may need several updating steps which may temporarily decrease the accuracy. Local classifier ensemble (LOCE) was proposed to overcome the drawbacks of the mentioned methods (ZareMoodi et al., 2015). To address the weak-verification of novel classes, LOCE constructs a neighborhood graph utilizing the novel class candidates and analyses this graph to detect emergence of probable novel classes. In order to address the slow adaption to concept drift issue, a prune phase is dedicated to eliminate obsolete classifiers as soon as possible. Hence, new metrics for determining obsolete classifiers are defined. Utilizing local patterns in LOCE improves the accuracy of both detecting novel class candidates, as well as classifying the instances of the known classes. However, the amount of memory usage and computational complexity of LOCE is relatively high. Moreover, throughout the stream the amount of memory usage increases, which leads to an increase in the computation time. SVSCLASS (ZareMoodi, Siahroudi, & Beigy, 2016) is another method in this area that used a support vector based method to maintain boundaries of known classes. This method labels a new instance based on its distance from class boundaries. If a new instance is in boundaries of known classes, the instance is labeled as member of the given class. When an instance is out of the boundaries, SVSCLASS constructs neighborhood graph to detect emergence of a novel class. SVSCLASS keeps hyper-spheres for updating its model. It dynamically updates its model by shrinking, enlarging and merging hyper-spheres in the kernel space and by this updating its model is capable to adapt to changes of underlying distribution of data. In SVSCLASS, the amount of memory usage increases throughout the stream, which leads to an increase in the computation time. There are also some other related works that close to our proposed method. Lughofer et al. proposed an extension of evolving fuzzy classifier with Gaussian-shaped rules to adapt with novel classes (Lughofer, Weigl, Heidl, Eitzinger, & Radauer, 2015). Their method is relying on all-pairs techniques which is decompose a multi-class classification problem into some binary classification problems. Their model can adapt with new classes by adding new rules. It should be noted that these method is unable to detect novel classes, and is only able to adapt its model with them. Lemos et al. proposed another extension for evolving fuzzy classifiers (Lemos, Caminhas, & Gomide, 2013). It can detects novel operation condition of a process (usually monitored industrial process). Then, an operator is notified to identify the corresponding operation mode. Eventually a new rule will be added to the model.

3. The proposed method

One major difference between stream data and batch data is that stream data arrives with time passage. Hence, we can not model the whole underlying distribution of data at first. Therefore, we build an initial model by using the first several thousand labeled instances of data. This model specifies the boundary of regions that instances located within and having the same class label. Then, we complete or change (if concept drift occurs) the model by observing newly arrived instances. If concept-drift does not occur, there are three possible condition for the location of each newly arrived instance: 1) the instance is located within the built boundary of a class, 2) it is located outside and near the boundary of a class, 3) it is located far from the boundaries of