Future Generation Computer Systems 75 (2017) 187–199

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Scalable real-time classification of data streams with concept drift

Mark Tennant [a], Frederic Stahl [a,*], Omer Rana [b], João Bártolo Gomes [c]

[a] University of Reading, Whiteknights, PO Box 225, RG6 6AY, Reading, UK
[b] Cardiff University, Computer Science & Informatics, Queen's Buildings, 5 The Parade, Roath, CF24 3AA, Cardiff, UK
[c] Institute for Infocomm Research (I2R), A*STAR, 1 Fusionopolis Way Connexis, Singapore 138632, Singapore

HIGHLIGHTS

- A real-time data stream classifier adaptive to concept drift and robust to noise.
- A parallel implementation of the real-time data stream classifier.
- A discussion about using open source Big Data technologies for data stream mining.

ARTICLE INFO

Article history:
Received 30 November 2015
Received in revised form
3 June 2016
Accepted 22 March 2017
Available online 9 April 2017

Keywords:
Parallel data stream classification
Adaptation to concept drift
High velocity data streams

ABSTRACT

Inducing adaptive predictive models in real-time from high throughput data streams is one of the most challenging areas of Big Data Analytics. The fact that data streams may contain concept drifts (changes of the pattern encoded in the stream over time) and are unbounded, imposes unique challenges in comparison with predictive data mining from batch data. Several real-time predictive data stream algorithms exist, however, most approaches are not naturally parallel and thus limited in their scalability. This paper highlights the Micro-Cluster Nearest Neighbour (MC-NN) data stream classifier. MC-NN is based on statistical summaries of the data stream and a nearest neighbour approach, which makes MC-NN naturally parallel. In its serial version MC-NN is able to handle data streams, the data does not need to reside in memory and is processed incrementally. MC-NN is also able to adapt to concept drifts. This paper provides an empirical study on the serial algorithm's speed, adaptivity and accuracy. Furthermore, this paper discusses the new parallel implementation of MC-NN, its parallel properties and provides an empirical scalability study.

1. Introduction

The 4 main aspects of Big Data are [1]: data generated at a fast rate (*Velocity*), very large and potentially unknown data quantities (*Volume*), uncertainty in the data (*Veracity*) and different forms of data such as text, structured data etc. (*Variety*). Other aspects of Big Data have been added over the years, i.e. *Volatility*, referring to how long the data is valid for, which is particularly relevant when referring to real-time data streams; and *Value*, referring to potential insights that can be derived by analysing the data. Regarding *Velocity*, data arriving at a very high speed challenges our computational hardware processing capabilities [2,3]. This paper presents an algorithm that addresses the overlap of the *Velocity* and *Volume* aspects of Big Data Analytics through a parallel and adaptive real-time data stream classifier. In data stream classification a classifier is trained in real-time on incoming labelled data instances. This classifier is then used in real-time to predict the class label of previously unseen data instances. The classifier is required to adapt to changes of concepts that can occur over time (known as concept drift [4]), in order to keep an accurate classification model over time.

The growing importance of data stream classification techniques is reflected through many commercial applications, such as: sensor networks; Internet traffic management and web log analysis [5]; TCP/IP packet monitoring [6]; intrusion detection [7]; and credit card fraud detection [8]. Due to high throughput of data and potentially infinite data streams, it is often not feasible to capture, store and process the data. In the past two decades this has led to the development and publication of data stream classifiers that can analyse the data in real-time as it is being generated. For example,

* Corresponding author.
E-mail addresses: M.Tennant@pgr.reading.ac.uk (M. Tennant), F.T.Stahl@reading.ac.uk (F. Stahl), RanaOF@cardiff.ac.uk (O. Rana), bartologjp@i2r.a-star.edu.sg (J.B. Gomes).

http://dx.doi.org/10.1016/j.future.2017.03.026

data stream classifiers such as Hoeffding Trees [9], G-eRules [10], Very Fast Decision Rules (VFDR) [11] only need one pass through the data and thus train and adapt to concept drifts in real-time scenarios. Nonetheless, their scalability is limited to the utilisation of one processing node at a time.

Few attempts have been made to combine parallelism and real-time data stream classification. Parallel binning is used by the SPDT [12] algorithm. However, the updating of SPDT classifier is not performed in parallel. Vertical Hoeffding Trees (VHDT) [13] partition the stream instances in terms of attributes, to support parallel processing. However, VHDTs scalability is limited by the number of attributes, as the attributes are distributed evenly over the number of processors utilised. In addition there exists a parallel method for concept drift detection termed Online MapReduce Drift Detection Method (OMR-DDM) [14], which makes use of the error rate of a collection of classifiers executed concurrently.

This paper proposes an inherently parallel adaptive data stream classifier termed MC-NN. The classifier is based on Nearest Neighbour (NN) classification and statistical summaries of the data and recency. The statistical summary is structured in the form of a set of variance based Micro-Clusters (MCs). Micro-Clusters continuously adapt to concept drifts through absorbing new data instances (updating statistics). An empirical evaluation [15] shows that the serial implementation of MC-NN is already very fast and robust to noise and concept drifts. However, it is limited by the throughput of a single computational node.

A parallel implementation of MC-NN is presented, along with a critical appraisal of implementation mechanisms that can be used to support parallel analysis of real-time data. A scalability evaluation is also carried out, identifying insights, difficulties and solutions in implementing parallel real-time data stream classifiers.

This paper is organised as follows: Section 2 summaries some related work. Section 3 describes the developed naturally parallel MC-NN algorithm and provides an empirical study comparing it with its serial competitors in terms of classification accuracy, adaptation to concept drifts and speed. Section 4 discusses the parallel implementation of MC-NN and provides an empirical scalability evaluation. It further discusses issues and experiences in implementing real-time data processing algorithms. Concluding remarks are provided in Section 5.

2. Related work

In the more general area of data mining an algorithm would iterate over the data several times in order to generate a model that fits the concepts (patterns) in the data. In each iteration the model is altered in order to better fit the concepts. However, as data streams are inherently infinite in length, iterative processes cannot be used. If left un-monitored, the algorithms would try to fit the concepts encoded on the whole stream and not account for *'Concept Drifts'*. 'Concepts' can be thought of as blocks of homogeneous/statistically similar data in a linear time frame. As the length and number of 'Concepts' is unknown the data stream must be monitored in real-time. An interesting area of research is the development of 'standalone' Concept Drift Detectors that monitor data streams in real-time. Typically, when a Drift Detector algorithm detects a concept drift (correctly or incorrectly) the current model is deleted and a new model is created. Examples of Concept Drift detectors are DDM [16], ECDD [17] and ELM[18]. Typically Concept Drift detectors work independently and concurrently from the underlying data mining algorithm (i.e. a classifier). Thus, both the data mining algorithm and the concept drift detector have to be computationally efficient in order not to hinder the computational performance. The data mining algorithm used in conjunction with the drift detector does not need to be adaptive — batch algorithms such as C4.5 [19], Support Vector Machines [20], N-Prism [21], KNN, etc. can also be used. This would require buffering enough data after the concept drift has occurred and then applying the batch data mining algorithm on the buffer. However, as mentioned earlier, batch algorithms typically require several passes through the data and thus may be too slow if data is arriving at a high speed. Further techniques exist to adapt non adaptive data mining algorithms to streaming data, such as sliding window [2] and reservoir sampling [22]. Reservoir sampling maintains an unbiased and representative fixed sized sample of the data instances retrieved from the stream, whereas sliding window based algorithms, such as G-eRules [10], consider only the most recent instances from the stream to build the data mining model. However, these techniques would require to re-train batch algorithms and thus may be too slow and impractical to use for data arriving at high speed.

Other techniques such as Hoeffding bound based techniques [23] and Micro-Clusters [24] have been used to create inherently adaptive data stream mining algorithms. Hoeffding based techniques aim to create and adapt data mining models based on a statistical upper bound on the probability that the so far received attribute values deviates from its expected value. The Hoeffding bound has been successfully used to create various data stream mining algorithms known as Very Fast Machine Learning (VFDT [25]). Micro-Cluster based techniques aim to create a statistical summary in terms of feature values, value distribution and time-stamps of the data retrieved from the stream (CluStream[24], On Demand Classification of Data Streams [26]).

A number of systems exist to support parallel stream processing, the most notable of these include Esper [27]. However, Esper makes use of a centralised architecture that runs on a single node and keeps everything (states, operators, and so on) in memory (although support is provided for multi-threading). However, if the continuous queries have a large window size and might require processing of a large number of data items/sec, Samza [28] provides a better alternative [29]. Samza can be though of as a simplified 'pilot job' [30]. In the pilot terminology a Samza job is an 'early bound' container that will process an unknown future workload. The key difference is that a pilot task is identified as an individual with its own workload to process, and therefore has no interaction with other tasks. Samza containers bound to a data stream are static, they have access to all data passed through the underlying stream until they are stopped externally. Other alternatives with similar functionality include: Apache Storm, Spark Streaming and Apache S4 — a comparison can be found in [29]. A number of messaging systems exist that Samza is capable of utilising. Broadly speaking they can be split into three groups: *Message Queue Systems*, such as Kestrel and RabbitMQ [31], *Publish Subscribe Systems*, such as Kafka [32] and Kestrel [33], and *Log Systems*, such as Flume [34] and Scribe [35]. The work presented in this paper develops an inherently parallel and adaptive data stream classifier that makes use of parallel stream processing technologies.

3. Adaptive Micro-Cluster nearest neighbour data stream classification

3.1. Micro-Cluster based nearest neighbour

In the authors' previous feasibility study [36], a real-time classifier was implemented based upon KNN. In KNN a data instance is assigned the class that is most common amongst its K Nearest Neighbours. The basic approach of the real-time KNN is to keep a sliding fixed sized time window of the most recent data instances and execute KNN from the sliding window set. Real-time KNN retrains on recent instances whilst older instances are deleted. However, real-time KNN is computationally limited