PROTEUS

Scalable online machine learning for predictive analytics and real-time interactive visualization

687691

D4.1 Investigative overview of targeted techniques and algorithms

Lead Author: José Barranquero, Hamid Bouchachia,
With contributions from: Alvaro Agea, Daniel Higuero
Reviewer: Asterios Katsifodimos

Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	M04
Actual delivery date:	M04
Version:	2.0
Total number of pages:	33
Keywords:	Data streams, Machine learning, Data mining, libraries, algorithms, big data

Abstract

Machine learning and data mining libraries for data streams are scarce and not mature. To contribute towards the development of new online learning algorithms for high speed data streams, the PROTEUS project will first overview the state of art of technology in this area. In fact, this report draws on this scope by highlighting the stream representation and summarisation techniques, clustering, frequent pattern mining, regression, and classification. It then discusses some potential directions that PROTEUS will take covering some particular areas of machine learning. Towards the end of the report, a brief presentation of library design is given.

[End of abstract]

Executive summary

Learning from high volume and high velocity continuous flow of data is an interesting scientific challenge that needs to be addressed. PROTEUS aims at designing and developing a library of original scalable streaming algorithms for predictive analytics and automatic knowledge discovery called SOLMA, standing for **S**calable **O**nline machine **L**earning algorithms and data **M**ining **A**lgorithms". It is very important to underline the fact that there exist few attempts to develop streaming libraries on big data platform; hence SOLMA will greatly contribute to this area the development of new online learning algorithms for high speed data streams.

To capture the need for this library, the present report tries to draw a picture of the state-of-the-art literature of existing techniques and algorithms covering a number of areas. First an introduction into the area of streaming and the nature of algorithms that suit and a list of machine learning tools is given. Then, stream representation and summarisation techniques are presented such as sampling, sketching, feature selection, windowing. Popular algorithms and techniques related to frequent pattern mining (with a focus on parallel ones), clustering (in particular, one-pass algorithms), regression and classification (tree-based algorithms) are discussed.

It then discusses briefly some potential directions that PROTEUS will take covering some particular areas of machine learning for data streams such as non-parametric models, deep learning, support vector machines and kernel methods, active learning and semi-supervised learning and finally change detection techniques.

Because PROTEUS' aims at develop scalable online machine learning and data mining algorithms for the particular big data platform, Apache Flink, a brief presentation of library design is given towards the end of the report.

Document Information

IST Project Number	687691	**Acronym**	PROTEUS
Full Title	Scalable online machine learning for predictive analytics and real-time interactive visualization		
Project URL	www.proteus-bigdata.com		
EU Project Officer	Martina EYDNER		

Deliverable	**Number**	D4.1	**Title**	Investigative overview of targeted techniques and algorithms
Work Package	**Number**	WP4	**Title**	Scalable online machine learning

Date of Delivery	**Contractual**	M04	**Actual**	M04
Status	version 3.0		final X	
Nature	report ☑demonstrator □ other □			
Dissemination level	public ☑ restricted □			

Authors (Partner)	LMBD, BU			
Responsible Author	**Name**	José Barranquero	**E-mail**	marco@novelti.io
	Partner	LMDP	**Phone**	+34 670460371

Abstract (for dissemination)	Machine learning and data mining libraries for data streams are scarce and not mature. To contribute towards the development of new online learning algorithms for high speed data streams, the PROTEUS project will first overview the state of art of technology in this area. In fact, this report draws on this scope by highlighting the stream representation and summarisation techniques, clustering, frequent pattern mining, regression, and classification. It then discusses some potential directions that PROTEUS will take covering some particular areas of machine learning. Towards the end of the report, a brief presentation of the design of a library that will cover the developed algorithms is given.
Keywords	Data streams, Machine learning, Data mining, libraries, algorithms, big data

Version Log			
Issue Date	**Rev. No.**	**Author**	**Change**
March 4th, 2016	V.0.5	J. Barranquero, H. Bouchachia	Initial draft
March, 6th2016	V.0.6	H. Bouchachia	Initial draft Section 4

March, 10th. 2016	V.0.8	J. Barranquero, H. Bouchachia	First full draft
March 15th 2016,	V 1.0	J. Barranquero, H. Bouchachia D. Higuero	Assessing streaming and scalable, non-scalable, non-streaming algorithms
March 17th, 2016	V 1.1	J. Barranquero, H. Bouchachia D. Higuero	Front information
March 21st , 2016	V 1.5	H. Bouchachia	Restructuring the document, introductory text to the sections
March 21st , 2016	V 1.7	H. Bouchachia	Front material
March 22nd, 2016	V. 1.8	H. Bouchachia D. Higuero	Change detection rewritten
March 23rd, 2016	V. 2.0	A. Agea H. Bouchachia D. Higuero	First version for review
March 25th. 2016	V. 2.5	J. Barranquero H. Bouchachia	Reviewed references and minor changes
March 30th, 2016	V. 3.0	A. Agea H. Bouchachia D. Higuero J. Barranquero A. Katsifodimos	Reviewed version.

Table of Contents

Abbreviations

ARM: Rule Association Mining

BBT: Balanced Binary Trees

BIRCH: Balanced Iterative Reducing and Clustering using Hierarchies

DBNs: Deep Belief Networks

DL: Deep Learning

FP: Frequent Pattern

GP: Gaussian Process

HWT: Haar-wavelet Transform

ML: Machine Learning

MOA: Massive Online Analytics

OGD: Online Gradient Descent

OMD: Online Mirror Descent

PARMA: Parallel randomized approximate association rule mining

RBMs: Restricted Boltzmann Machines

SOLMA: Scalable Online Machine Learning and Data Mining Algorithms

SVM: Support Vector Machines

VFDT: Very Fast Decision Trees

VHT: Vertical Hoeffding Trees

1. Introduction

The advent of big data has emerged along with new analytics challenges due to its inherent requirements related to volume, velocity and variety. Huge volume of multi-modal data is being generated every day. Most of it comes in the form of continuous streams, especially with the spread of Internet of Things and more generally the cyber-physical systems that involve astronomic numbers of networked devises. Data stream processing is therefore shaping the big data landscape and sparking an increasing interest from industry as well as from academia. Stream analytics tools are being investigated and developed. PROTEUS intends to contribute to the effort of creating efficient analytics technologies and tools.

However, analytics systems are traditionally built from data samples in an off-line one-shot experiment [Bouchachia, 2007]. Once the learning phase is exhausted, the learning system is no longer capable of learning further knowledge from new data nor is it able to update itself in the future. In PROTEUS, we consider the problem of online and incremental learning where learning is continuous and is from data streams that evolve over time. The basic idea motivating online learning is that as new data points arrive, new knowledge elements may be created and existing ones may be modified allowing the knowledge base (respectively, the system) to evolve over time. Thus, the acquired knowledge becomes self-corrective in light of new evidence. This update is of paramount importance to ensure the adaptivity of the system. However, it should be meaningful (by capturing only interesting events brought by the arriving data) and sensitive (by safely ignoring unimportant events). This is known as plasticity-stability dilemma (see Fig. 1).

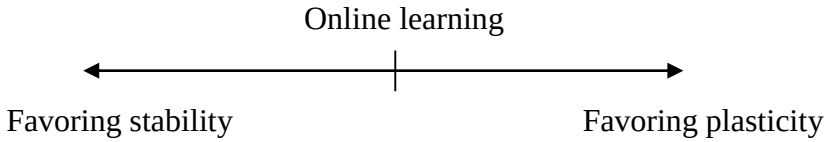

Figure 1: Learning spectrum

Perceptually, online learning is a fundamental problem of cognitive development. Indeed, the perceiver usually learns how to make sense of its sensory inputs in an incremental manner via a filtering procedure.

The current state of the art of machine learning algorithms for Big Data is dominated by offline learning algorithms that process data stored in the cloud. Data can be revisited as much as desired. In online learning however, the algorithms see data only once. Moreover, the traditional meaning of "online" is that data is processed sequentially one by one but for many epochs. Although there have been many developments related to online learning (for data streams), the scalability remains still an issue. There is currently, nonetheless, an increasing interest from the machine learning community to develop scalable online machine learning and similarly from the data mining community to propose streaming algorithms (mostly for sensor networks and security applications) that are capable of efficiently handling continuous high-speed streams. More detail will follow in the next sections. It is however worthwhile, to draw a representative picture of the exiting streaming technologies, mostly from big data platforms be it machine learning or data mining[1], before delving into further details.

[1] In the rest of this document, we will make distinction between machine and data mining algorithm when appropriate, otherwise, we call both classes of algorithms "streaming algorithms" to facilitate the explanation. Note that existing machine learning libraries for big data platforms contain data mining algorithms as well.

Table 1 shows an illustrative (none exhaustive) sample of existing libraries. Column "Framework" indicates that there are readily available streaming platform such as: Apache Storm, Apache Spark, Apache S4, Apache Samza and Apache Flink which is adopted in PROTEUS. From the perspective of machine learning algorithms, there exist many libraries dedicated to batch processing. Some are for big data such as GraphLab, Giraph, Pegasus, MLib and the famous Mahout, and others are not. These libraries are compared on the basis of the following characteristics: data stream processing, scalability, simplicity of use, framework, and maturity (coverage of various classes of ML algorithms). Such libraries offer different machine learning, data mining and graph-oriented algorithms. Almost all of them are developed for batch processing.

Library	Streaming/online	Scalability	Simplicity of use	Framework	Maturity
SAMOA http://samoa-project.net/	+	+	+	S4, Storm, Samza	-
MOA http://moa.cms.waikato.ac.nz/	+	-	+	Not for big data	~
MLI/MLbase http://www.mlbase.org/	-	+	+	Spark	-
GraphLab http://www.cs.cmu.edu/~Pegasus/	-	+	-	Spark, Hadoop	~
Mahout http://mahout.apache.org/	-	+	+	Hadoop	-
RapidMiner https://rapidminer.com/products/studio/	-	-	+	independent	+
Liblinear http://www.csie.ntu.edu.tw/~cjlin/liblinear/	-	-	+	Independent	-
Vowpal-Wabbit (https://github.com/JohnLangford/vowpal_wabbit/wiki)	+	+	+	Multi-core machines	-
Shogun (http://www.shogun-toolbox.org/)	-	-	+	independent	-
MLPack (http://www.mlpack.org/)	-	-	+	independent	-
SystemML (http://researcher.watson.ibm.com/researcher/view_group.php?id=3174)	-	+	-	Hadoop	-
Pegasus (http://www.cs.cmu.edu/~pegasus/faq.htm)	-	+	-	Hadoop	+
Giraph (http://grafos.ml/#realTime)	~	+	-	Hadoop	+
OptiML (http://ppl.stanford.edu/main/optiml.html)	-	+	+	multi-core	-
MADlib (http://madlib.net/)	-	-	+	independent	+
scikit-learn (http://scikit-learn.org/stable/)	-	-	+	independent	+
Weka http://www.cs.waikato.ac.nz/ml/weka/)	-	-	+	independent	+
Matlab (http://mathworks.com/)	-	+	+	independent	+
R (http://www.r-project.org/)	-	-	+	independent	+
PRtools	-			independent	+

For online stream processing, there is no mature framework yet. To the best of our knowledge, there has been only one attempt, called SAMOA that tries to convert the Massive Online Analytics (MOA), originally developed for non-distributed architecture, to a distributed equivalent. This library works on top of Storm and S4, but contains very few machine learning and data mining algorithms: one classifier (vertical Hoeffding tree), one clustering algorithm (CluStream), one regression algorithm (decision-rule regression algorithm), and one ensemble classifier (bagging). Vowpal-Wabit also contains some of the online scalable algorithms that work based on stochastic gradient descent and its variants which will be presented later.

1.1. Document objectives

This document contains an overview of most common streaming algorithms. The intended objective if this overview is to capture the existing machine learning and data mining algorithms that could be considered to be enhanced toward scalability (re-designed) and implemented for inclusion in SOLMA, the Streaming library of the PROTEUS project. As a part of the algorithms review, an acceptable level of detail is provided as this document is not meant to reveal the technical (mathematical) details in depth. Mostly the document collects information on the algorithms, their main purpose and their current status in terms of availability and implementation for parallel processing and in particular for continuous data streams.

A further objective is to discuss the implementation choices of SOLMA, thus at the end we have clear understanding about the design of SOLMA and the candidate algorithms that go in.

1.2. Document structure

The document consists mainly of three parts. The first part describes the state-of-the-art online streaming algorithms, both scalable and non-scalable. The second part presents some of the techniques and algorithms that are currently either non-streaming or non-scalable and that could be potentially investigated in the context of PROTEUS. The last part briefly highlights the design of SOLMA.

2. Stream representation and summarisation

The following section contains an overview of the most common streaming algorithms that cover sampling, pattern mining, and clustering. To meet the purpose of the deliverable, we will highlight the algorithms, their distributed implementation if any and discuss their scalability in general.

2.1. Sampling

Sampling is the natural way of summarizing a data stream as it allows to compute a representative set of stream elements so that an efficient processing. In general the sample is continuously maintained in order to accommodate anytime approximate query answering, selectivity estimation, query planning, or any other mining task. The sample fits in the RAM, hence various standard offline algorithms can be applied. The challenge is to develop sampling techniques that provide unbiased estimate of the underlying stream with provable error guarantees. In one-pass stream sampling, the probability of including a data point in the sample may depend on its value. Sampling becomes difficult, in particular, when the data is split across multiple distributed sites or when the processing environment uses distributed computing approaches. The main challenge is to ensure that a sample is drawn uniformly across the union of the data while minimizing the communication needed to run the protocol on the evolving data. At the same time, it is also necessary to make the protocol lightweight, by keeping the space and time costs low for each stream [Cormode et al., 2011].

Random sampling is the most common approach to approximate data management and querying, and it had been studied in the streaming setting long before formal models of data streams were first introduced. The most classical approach is *reservoir sampling* [Knuth, 1981]. The algorithm maintains a random sample of size *s* without replacement over a stream. It is initialized with the first *s* elements; when the *i-th* element arrives for $i > s$, with probability $1/i$ the model adds the new element, and replaces an element uniformly chosen from the current sample. There have been various extensions to the basic reservoir sampling algorithm.

Using an appropriate distribution, it is possible to determine how many new elements to "skip" over until the next sample will be drawn [Olken, 1997]. In a different approach, Gibbons and Matias introduced concise sampling and counting sampling, to make better use of the space [Gibbons and Matias, 1998]. "Distinct samples" aim to draw a sample from the support set of the multi-set of items in a stream, possibly containing deletions [Frahling et al., 2005][Gibbons, 2001]. Priority Sampling is another sampling technique that aims to reduce the variance on subset-sum queries [Duffield et al., 2003]. There has also been work on sampling from streams which include deletions [Gemulla, 2007].

Random Sampling algorithms have been repeatedly implemented for distributed stream scenarios [Cormode et al., 2011], and parallel implementations are available in most of more common processing frameworks [Chung, 2013].

2.2. Histograms

Histograms are structures that provide a summarized view of the data capable of aggregating the distribution of values in a dataset [Gibbons et al., 1997] [Datar et al., 2002] [Guhan, 2006]. They are very common in tasks such as query size estimation, approximate query answering, and data mining.

The structure of histograms allows building simple graphical representations of the distribution of numerical data used for:

- The estimated probability distribution of random variable
- Approximate query answering with error guarantees or estimations
- Obtaining an approximate value of the frequency distribution of element values
- Partitioning data into a set of contiguous buckets

The most common types of histograms for data streams are: *V-optima*l histograms, *equal-width* histograms, *end-biased* histograms. V-optimal histograms approximate the distribution of a set of values *v1, ..., v2* by a piecewise constant function *vb(i),* with the scope of minimizing the sum of squared error.

Several parallel implementations [Jung et al., 2014] [Milic et al., 2013] [Demetrescua et al., 2011] for stream processing are available. For a review in depth, please see [Ross, 2014].

2.3. Sliding windows

As sliding windows is a very common tool for various streaming applications and algorithms, there has been much interest in understanding how to maintain a sample efficiently over a sliding window [Datar et al., 2002] [Braveman et al., 2009] [Gemulla, 2008]. There are two commonly accepted models for sliding windows samples: *sequence-based windows* and *time-based windows*. In sequence-based windows, the sample keeps a summary of the last *N* elements in the stream. In time-based windows, every element arrives at a particular time and the scope is to maintain a sample summary over the elements that have arrived in the time interval $[t - w, t]$ for a window length *w* from the current time, *t.*

Sampling from a sliding window is similar to sampling from the distinct items in a dataset in that we do not know the size of the underlying window or dataset. For some applications, however, it is important to be able to at least estimate the window size in order to make effective use of the sample. Most of the parallel stream processing engines, and the corresponding ML library implementations provide a support for the sliding window sampling [Lu et al., 2010] [Cormode et al., 2010].

In the context of network analytics a framework for developing algorithms that can adaptively learn from data streams that change overtime (i.e. evolving data streams) is presented in [Bifet et al., 2010]. These methods are based on using change detectors and estimator modules at the right places. It presents an adaptive sliding window algorithm ADWIN for detecting change and keeping updated statistics from a data stream, and use it as a black box in place or counters or accumulators in algorithms initially not designed for drifting data. More details on sliding windows can be found in [Gama et al., 2014].

Nevertheless, none of the latter general-purpose techniques show a clear answer to scalability with respect to an exponential growth of data sources.

2.4. Multi-resolution models

A multiresolution model consists of a collection of mesh fragments, usually describing small portions of an object with different level of details, so after building them, there is a mechanism to select a subset of fragments - defined quality criteria - and combining them into a reconstructed mesh that represents the object.

Manu multi-resolution techniques have been developed and they include Wavelets, Micro-Clusters and binary balanced trees. The key element in these techniques is the ability of capturing different resolutions at different time scales in the data stream. For a good review of all of these techniques see [Cormode et al., 2011][Han et al., 2005].

Microclusters maintain statistical information about the data locality. They are defined as a temporal extension of the cluster feature vector, and they are commonly used for stream clustering and other stream classification algorithms. By definition *microclusters* have the additivity property that makes them a natural choice for the massive data stream problem, and for the corresponding parallel processing implementations. More details on clustering will follow.

Wavelets have traditionally been used in a variety of image and query processing applications. The wavelet approach to build data summaries is conceptually close to histograms. The central difference is that, whereas histograms primarily produce buckets that are subsets of the original data attribute domain, wavelet representations transform the data and seek to represent the most significant features in a wavelet (i.e., "frequency") domain, and can capture combinations of high and low frequency information.

The most widely discussed wavelet algorithm approach is the *Haar-wavelet Transform* (HWT), which can be constructed in time linear in the size of the underlying data array. Several one-pass implementations for streaming data are available [Gilbert et al., 2003][Gilbert et al., 2001].

Most of the common distributed stream processing engines, such as SPARK have proven implementations of wavelet distributed algorithms.

Alternatives to wavelets and microclusters are **balanced binary trees (BBT).** A BBT is a tree where every leaf is "not more than a certain distance" away from the root than any other leaf. These structures provide efficient implementations for mutable ordered lists, and can be used for other abstract data structures such as associative arrays, priority queues and sets. Several parallel implementations are available [Chatterjee, 2002].

2.5. Sketches

The idea of sketches is to replace the sample approach – where the sample only takes information of those items which were selected to be in the sample – and use method able to see the entire input. Sketches are techniques used to compute different types of frequency statistics [Matías et al., 1999]. Such statistics are designed to provide approximate answer to specific queries. As a consequence sketches are restricted to retain only a small summary of it. There are queries that can be approximated well by sketches that are provably impossible to compute from a sample.

There are many types of sketches and generally speaking sketches are linear transformations that are widely used to generate approximate answers in stream mining. Some sketch-based methods derive their inspiration from wavelet techniques [Aggarwal et al. 2007].

Linear sketches are given as a linear transformation of the input vector v into s: $P_{nxm} * v_m = s_n$ where $n \ll m$, that corresponds to dimensionality reduction. In fact, sketch based methods can be considered a randomized version of wavelet techniques, and are among the most space efficient of all methods. The most known linear sketches are: frequent items, norms, quantiles, histograms, random subset sums, different counting sketches, Bloom filters, etc.

There are several implementations of sketches for distributed environments, such as Algebird (used with Scalding or Summingbird) that facilitates popular stream mining computations like the Count-Min sketch and HyperLogLog. The performance of sketches algorithms in distributed scenarios, and in particular based on the usage of hashing as technique to manage them has been reviewed in [Weinberg et al., 2010]. The main challenge for parallel computation is the size of the data, that is,

when it is large and of the same order of magnitude as the time series, may lead the computation may be quadratic in the size of the series.

2.6. Randomized algorithms

A randomized algorithm [Ynag, 2015] is one that receives, in addition to its input data, a stream of random that it can use for the purpose of making random choices. With this approach, even for a fixed input, from different runs of a randomized algorithm we should expect different results. It is then inevitable that a description of properties of a randomized algorithm will involve probabilistic statements. For example, even when the input is fixed, the execution time of a randomized algorithm is a random variable. The goals of these algorithms are simplicity, speed and sometimes other objectives depending on the problem (e.g., for Nash Equilibrium, the problem inherently requires randomization) [Karp, 1991] [Motwani et al., 1995].

Randomized algorithms have been commonly used for sampling and sketching large data set, in particular for infinite data streams. Different implementations have been tested on distributed platforms such as Spark [Strata, 2015].

3. Frequent pattern mining

One of the applications of sketching discussed earlier is the counting statistics. Frequent patterns mining is one of the most standards in the area of data mining. The goal is to mine frequent itemsets from transactional data. Such itemsets are then used for computing association rules between items. There has been an extensive work on rule association mining (ARM) using both centralized (single machine) and distributed approaches. For an overview, readers are kindly referred to [Zaki, 1999] [Li et al., 2008] [Ozkural et al., 2002] [Park et al., 2002] [Zeng et al., 2012] [Moens et al., 2013]. In the following we present a sample of frequent pattern mining algorithms dedicated to parallel and distributed platforms which have been already implemented in big data platforms.

3.1. Parallel FP-growth

The FP-growth algorithm is for frequent pattern mining algorithm described in the paper [Han et al., 2008][Han et al., 2000][Giannella et al., 2003]. Given a dataset of transactions, the first step of FP-growth is to calculate item frequencies and identify frequent items, then the second step of FP-growth uses a suffix tree (FP-tree) structure to encode transactions without generating candidate sets explicitly, which are usually expensive to generate. After the second step, the frequent itemsets can be extracted from the FP-tree.

The most common parallel implementation such that of MLIB is called PFP, as described in [Li et al., 2008]. PFP distributes the work of growing FP-trees based on the suffices of transactions, and hence more scalable than a single-machine implementation. More details on other parallel implementations may be found in [Li et al., 2008] [Zhou et al., 2010][Zhou et al., 2014][Ontic, 2015].

3.2. Parallel Randomized Approximate Association Rule Mining

PARMA, standing for Parallel randomized approximate association rule mining [Riondato et al., 2012], finds approximate collections of frequent itemsets. The authors guarantee the quality of the frequent itemsets that are being found, through analytical results. It achieves near-linear speedup while avoiding costly replication of data. PARMA does this by creating multiple small random

samples of the transactional dataset and running a mining algorithm on the samples independently and in parallel. The resulting collections of frequent itemsets or association rules from each sample are aggregated and filtered to provide a single collection in output. Because PARMA mines random subsets of the dataset, the final result is an approximation of the exact solution.

On the parallel implementation see [Cheng et al., 2008], and on general overview on the distributed version challenge see above references on FIM on big data.

3.3. Moment

Maintaining Closed Frequent Itemsets over a Stream Sliding Window (Moment) is an algorithm designed to solve the problem of mining closed frequent itemsets over a sliding window [Chi et al., 2004] using limited memory space. It make use of a synoptic data structure that allow being able to monitor transactions in the sliding window. With this, the algorithm can output the current closed frequent itemsets at any time. The limitations come with memory constraints. The synopsis data structure cannot monitor all possible itemsets, but monitoring only frequent itemsets will make it impossible to detect new itemsets when they become frequent. The algorithm solves this introducing a compact data structure, the closed enumeration tree (CET), to maintain a dynamically selected set of itemsets over a sliding-window. Moment is available in MOA.

3.4. FP-Stream

It is the streaming version of the FP-Growth algorithm [Giannella et al., 2003]; Whereas FP-Growth can analyze static data sets for patterns, FP-Stream is capable of finding patterns over data streams. FP-Stream relies on the FP-Growth for significant parts, but it's considerably more advanced. It maintains pattern tree tilted-time window, and mines frequent itemsets using a FP-Growth approach at every batch – the tilted time-windows. A technique for building the tree and allow to answer time-sensitive queries are defined. For a review of implementations see [Xiao, 2007].

4. Clustering of data streams

Clustering aims at grouping data to uncover the hidden structure that underlies the data. However, clustering of data streams can be seen from two perspectives: (1) summarization like the techniques shown in Sec. 2 above, (2) structure discovery as in standard offline clustering. Indeed most of the data stream clustering algorithms use (1) as part of (2) to accommodate the incremental nature of processing. We can classify clustering algorithms mostly into two categories: point-based clustering and granule-based clustering. The first class processes individual data points sequentially as they become available, while the second class uses windows/chunks and local clusters are computed for each chunk before these are re-clustered at a higher level. Usually stream-based clustering algorithms retain some (hyper-) structures that are computed recursively and that summarize the incoming data at the semantic level.

For the sake of illustration, in the following we provide a short description of some of the streaming clustering algorithms.

4.1. DBSCAN

DBSCAN [Ester et al., 96] is a density-based clustering algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby

neighbors), marking as outliers' points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.

DBSCAN does not require one to specify the number of clusters in the data a priori, as opposed to k-means. DBSCAN can find arbitrarily shaped clusters and it can even find a cluster completely surrounded by (but not connected to) a different cluster. DBSCAN has a notion of noise, and is robust to outliers.

DBSCAN has been used for streaming purposes in several approaches. For example, it has been used in DenStream for the offline clustering phase. OPTICS is an adapted DBSCAN version that has been implemented in distributed stream processing engines such as Spark Streaming.

4.2. CluStream

CluStream is a clustering algorithm proposed in [C. Aggarwal, 2003]. It is based on the concept of microcluster. The algorithms consist of two phase: online and offline. The *online phase* computes micro-clusters that summarize some per-feature statistics (sum and squared sum of values of the data). These micro-clusters represent the current snapshot of clusters and change over time as new data points become available. Their number is usually higher than the optimal number of clusters. Each instance coming from the input stream can then be added to the most similar micro-cluster or it leads to the creation of a new micro-cluster. New micro-clusters can also result from merging existing micro-clusters. They can also be deleted using their age. The *offline phase* is straight forward and consists of clustering the available micro-clusters at any time. Specifically, a weighted version of k-means is applied to produce high-level clusters.

CluStream has been introduced to respond to the one-pass natural requirement of stream processing and the consequent complexity of flexibly computing clusters over different kinds of time horizons using conventional algorithms. A direct extension of the stream based k-means algorithm in to such a case would require a simultaneous maintenance of the intermediate results of clustering algorithms over all possible time horizons that will irremediably create a bottleneck for online implementation.

Therefore the CluStream approach is natural design to stream clustering algorithms. Several implementation and extension of CluStream have been made available (MOA, SPARK-MLIB, SAMOA, CLUSANDRA), both in for distributed and standalone scenarios, and have been used for classification and network intrusion detection purposes, just to mention some. In particular, a distributed version of CluStream was developed in SAMOA.

4.3. DenStream

DenStream [Cao et al., 2006] is the density-based equivalent to Clustream. Instead of the weighted k-means, DBSCAN is applied (See below more details on DBSCAN). A notion of "dense" micro-cluster (named core-micro-cluster) is introduced to summarize the clusters with arbitrary shape, while the potential core-micro-cluster and outlier micro-cluster structures are proposed to maintain and distinguish the potential clusters and outliers. Standard implementations are available in MOA and parallel implementations on GPUs have been already experimented in G-Stream.

4.4. ClusTree

ClusTree [Kranen et al., 2009] [Kranen, 2011] is an anytime and parameter-free incremental on-line stream hierarchical clustering algorithm. It incorporates the age of the data points to reflect on the importance of more recent data. The algorithm makes use of the micro-cluster approach and

maintains a balanced hierarchical data structure for the clusters. ClusTree have been implemented in MOA and parallel implementation using [Zhinoos et al., 2015] have been recently explored. The algorithm is capable of detecting concept drift, novelty and outliers, and for efficient handling it introduces a self-adaptive index structure for maintaining stream summaries.

4.5. BIRCH

BIRCH (balanced iterative reducing and clustering using hierarchies) [Zang et al., 1996] is used to perform hierarchical clustering over large data-sets. The algorithm runs in four phases, where the 2nd is optional. The first phase builds cluster features and structures them in a hierarchical balanced tree. Each leaf contains a limited number of entries and then a child node is created based on threshold parameter. The tree is simplified by phase 2, to remove outliers and merge crowded sub-clusters. Then at each leaf a clustering process is conducted. After this step a set of clusters is obtained that captures the structure of the data. However, there might be minor and localized inaccuracies which can be handled by an optional step 4 by realigning the clustering with previous centroids.

An advantage of BIRCH is its ability to incrementally and dynamically cluster incoming, multi-dimensional metric data points in an attempt to produce the best quality clustering for a given set of resources (memory and time constraints). In most cases, BIRCH only requires a single scan of the database. Implementations are available at JBIRCH, SCIKIT_Learn and a discussion on its parallel accelerated implementation using CUDA environments may be found at [Dong et al. 2013].

5. Regression of data streams

Regression models seek to capture the dependencies between variables (features) in a dataset, possibly in the presence of noise. There are different classes of regression techniques, mostly categorized into linear and non-linear models and rely on various least deviation methods to fit the models to the data at hand.

To compute regression models from streaming data, the same philosophy as with clustering is adopted. The regression function parameters, $b_j, j = 1, \dots, m$, are recomputed on the fly as new data arrive using some synopsis collected from the previous data along with recursion mechanisms. For instance, the solution of the linear regression model, using the least squares, is given by: $B = (X^T X)^{-1} X^T Y$, where X is the data and Y are the corresponding responses. It can be easily shown that we can adapt this model to accommodate new batches of data [Nadungodage et al., 2011].

$$B = (M + X_i^T X_i)^{-1}(V + X_i^T Y_i)$$

where i is the index of the new batch and M and V are obtained from the previous batches recursively and correspond to:

$$\begin{cases} M = X^T X \\ V = X^T Y \end{cases}$$

In the following, we show some of the non-linear regression models.

5.1. Gaussian Process Regression

Formally, a Gaussian process generates data located throughout some domain such that any finite subset of the range follows a multivariate Gaussian distribution. The algorithm propose in [Seeger, 2004][Rasmussen et al., 2006][Petelin, 2013] is an example of a probabilistic, non-parametric

model with uncertainty predictions. The output of the GP model is a normal distribution, expressed in terms of the mean and the variance.

The algorithm infers continuous values with a Gaussian process prior is known as Gaussian process regression, or kriging. The algorithm allows its extension to Gaussian process regression with multiple target variables, known as known as cokriging. Gaussian processes are useful as a powerful non-linear multivariate interpolation tool. Additionally, Gaussian process regression can be extended to address learning tasks in both supervised (e.g. probabilistic classification) and unsupervised (e.g. manifold learning) learning frameworks. Standard implementations may be found in GPLP [Park et al 20012] or in GP-NBC [Conlin 2014] for streaming implementations.

5.2. Kernel Regression

Kernel regression is a well-known non-parametric regression technique in statistics to estimate the conditional expectation of a random variable. The objective is to find a non-linear relation between a pair of random variables X and Y. The scalability of kernel machines is a big challenge when facing millions of samples due to storage and computation issues for large kernel matrices that are usually dense. As reviewed in [S. Si et al 2014], many papers have suggested tackling this problem by using a low-rank approximation of the kernel matrix.

6. Classification of data streams

One of the areas that has attracted a lot attention from different communities, especially from the machine learning community is classification. Some very early classification algorithms dedicated to data streams are based on decision trees. In the following, we show some examples thereof, but other classes of classifiers will be discussed in the general context of online machine learning, that will drive the PROTEUS investigations in the future.

6.1. Very Fast Decision Trees

Hoeffding Tree or VFDT [Hulten et al., 2001][Domingo et al., 2000] is the standard decision tree algorithm for data stream classification. The Hoeffding tree induction algorithm induces a decision tree from a data stream incrementally, inspecting each example in the stream only once, without the need for storing examples after they have been used to update the tree. The only information needed in memory is the tree itself, which stores sufficient information in its leaves in order to grow, and can be employed to form predictions at any point in time between processing training examples.

VFDT uses the Hoeffding bound to decide the minimum number of arriving instances to achieve certain level of confidence in splitting the node. This confidence level determines how close the statistics between the attribute chosen by VFDT and the attribute chosen by decision tree for batch learning.

VDFT has been implemented in different libraries on top of different processing frameworks, being the most known the implementation in SAMOA. Different variations considering changes and concept drift have been proposed and implemented. [Hulten et., 2001][Gama et al., 2006].

6.2. Vertical Hoeffding Trees

Based on the work of the SAMOA team, The Vertical Hoeffding Tree (VHT) classifier is a distributed classifier that utilizes vertical parallelism on top of the Very Fast Decision Tree (VFDT) or Hoeffding Tree classifier proposed [Murdopo et al., 2014] and currently implemented in SAMOA.

The implementation is based on the standard Vertical Parallelism approach in which the partitions are built in term of attributes for parallel processing. In this implementation approach, the VHT decision tree induction processes the partitioned instances to calculate the information-theoretic criteria in parallel. For example, if we have instances with 100 attributes and we partition the instances into 5 portions, we will have 20 attributes per portion. The algorithm processes the 20 attributes in parallel to determine the "local" best attribute to split and combine the parallel computation results to determine the "global" best attribute to split and grow the tree.

Alternative parallel implementations have been proposed [Ben-Haim, 2010] using horizontal parallelism approaches, and they have been also made available in SAMOA.

7. Scalable online ML algorithms for data streams: PROTEUS scope

The sections 2-7 describe the standard algorithms and techniques that can be found in the literature related to streaming. Examples of these will be implemented in SOLMA, the PROTEUS library. However, the research that will be developed in PROTEUS will focus more on online machine learning algorithms, their scalability and implementation on distributed big data platforms.

7.1. Relevant notions

Scalable machine learning algorithms has recently gained an increasing interest due to the advent of Big Data. The meaning of scalability is understood in the traditional sense; so an algorithm is scalable if it can cope efficiently with large amount of data. The big data can be static or dynamic. In the former case, the data is split and the ML algorithm is designed to run in parallel but offline. In the latter case, the data arrives as a stream, possibly generated by a non-stationary process partitioned into chunks of length 1 or more and the ML algorithm is designed to run in parallel, in an online mode. Hence “big” data streams require scalable online learning algorithms that operate sequentially and fast, in linear or sub-linear time.

While there has been a lot of effort to develop online algorithms (see Sections 2-7), little has been devoted to their development for distributed platforms. In the following we will explain the main paradigms used to devise online ML algorithms and distributed ML algorithms, before we highlight some of the ML classes of algorithms that require further development in the context of PROTEUS.

7.2. Online learning

Transforming offline algorithms into online ones often require approximation and recursion. Online learning algorithms can be analysed using stochastic approximation as shown in [Bottou, 2010]. The most known scheme for doing this is to use online gradient descent (OGD) instead of batch gradient to minimise the empirical risk which is given as:

$$J(w) \triangleq \frac{1}{N}\sum_{i=1}^{N} Q(z_i, w) \qquad (1)$$

where *w* indicates the learned system's parameters, z_i the i[th] input and $Q(z_i, w)$ is the loss function that measures the performance of the learning system given its parameters and the input z_i. In other terms, the learning rule is given as:

$$w_{t+1} = w_t - \alpha_t \frac{1}{N} \sum_{i=1}^{N} \nabla_w Q(z_i, w_t) \quad (2)$$

where α_t is the learning rate. An iteration of the batch gradient descent algorithm requires the computation of the average of the gradients of the loss function $\nabla_w Q(z_i, w_t)$ over the whole training set (Eq. 2). The online gradient descent is a more simplified form of Eq. (2) for updating the system's parameters sequentially without the recourse to the average of the loss function:

$$w_{t+1} = w_t - \alpha_t \nabla_w Q(z_t, w_t) \quad (3)$$

The factor 1*/N is* absorbed into the step-size.

While online gradient descent is popular, it is just a special case of online mirror descent (OMD). This latter assumes the application of any convex regularization function to handle online convex optimisation, a class of online learning algorithms. This class is more general because, the prediction domain is a convex set and the loss function is a convex function. Hence, OGD is obtained by reducing convex functions to linear (and affine) functions which are convex too as discussed in [Shalev-Shwartz, 2011] [Orabona et al. 2015]. OMD has been used for both regression and classification.

Therefore to transform an offline ML algorithm into an online one, we need to rewrite the batch (offline) learning rule. Ideally the online version should be lossless in the way that it behaves equivalently, achieving the same or close performance as the offline version. Most of the existing online approaches pertaining to various ML learning paradigms (e.g., statistical learning, neural networks, graphical models, etc.) rely on the principal of stochastic and approximation; see for instance [Arabona et al. 2015][Bouchachia and Vanaret, 2015][Liang et al., 2006][Bordes et al., 2005][Laskov et al., 2006].

It must however be noted that a large body of work was driven by the regret minimisation in sequential predictions [Cesa-Bianchi and Lugosi, 2006][Shalev-Shwartz, 2011] based on online convex optimization. The idea is to abandon the basic assumption that the outcomes are generated by an underlying stochastic process and view the sequence as the product of some unknown and unspecified phenomena which could be deterministic, stochastic, or even adversarially adaptive to the learner. Many online learning algorithms have been inspired by online convex optimization [Kivinen and Warmuth, 2001][Kakade et al., 2009] [Shalev-Shwartz and Singer, 2007].

It is however worthwhile noticing that these algorithms assume always the existence of an oracle (feedback) to adjust the learner's model. The question is therefore how can they cope with streaming data, where potentially no continuous feedback or none is available? Moreover they have not been studied in the context of distributed systems to check their scalability in presence of high-velocity streams that may require many processors. Nevertheless, the algorithms based on Expert Advice look amenable to a distributed architecture in a straightforward.

7.3. Distributed online learning

There has been over the recent years an ever growing attention to distributed learning and a number of contributions and tools have been introduced [Zhang et al. 2012][Zhang et al. 2013][Balcan et al., 2012][Boyd et al., 2011][Duchi et al., 2012]. In terms of distributed online learning, there exist

very few studies. For instance the authors in [Dekel et al., 2012] introduce a distributed mini-batch algorithm that illustrates how serial gradient-based online prediction can be implemented on a distributed platform. Similar work was presented in [Recht et al., 2011] to parallelise stochastic gradient descent with a focus on lock-free update approach. Much of the work in this area is based on ideas in [Bertsekas and Tsitsiklis, 1997] for parallelizing stochastic gradient descent.

In terms of parallel and distributed computing strategies, often we have two main strategies: model-based parallelism and data-based parallelism. The model-based parallelism assumes that the model (e.g., deep neural network) is built on different workers. Each worker is dedicated to learn part of the model. The most popular strategy is however the data-based parallelism where data is split on independent workers with no order and let each worker computes its own update on the local data before the partial results are aggregated (averaged) at the central worker. This approach has been adopted in many studies like in [Zinkevich et al., 2010][Mann et al., 2009]. In [Zinkevich et al.,2010], data is partitioned into k blocks of size c = O(m/k) of the dataset of randomly drawn. Each block is assigned to a worker which runs the stochastic gradient descent separately with constant learning rate as in Eq. (5). Finally the average of $w's$ obtained from all workers is computed. The simple algorithm was proven to converge.

In [Mann et al., 2009] three distributed methods are used to train the conditional maxnet classification model: a distributed gradient computation, majority voting, and a mixture weight method. In the first, the model parameter vector is computed in parallel by m workers and then aggregated. In the second, the workers are trained on different data sample and predictions are made locally before a majority voting is used to compute the overall output. In the last approach the parameter vectors of the individual trained models are weighted and summed up. The result is used to do prediction.

Theoretical work showed that the convergence (error bounds) of the various update schemes [Zinkevich et al., 2010][Mann et al., 2009][Zhang et al. 2012][Zhang et al. 2013][Balcan et al., 2012][Boyd et al., 2011][Duchi et al., 2012] [Dekel et al., 2012][Recht et al., 2011].

For instance, [Shamir, 2014] describes the limits of online (with partial information and stochastic optimisation) and distributed algorithms for statistical learning and estimation. [Tekin and van de Shaar, 2013] presents a distributed framework along with its theoretical convergence for online big data classification using contextual bandit. [Langford et al., 2009] show that online learning with delayed updates converges well, thereby facilitating parallel online learning.

Following the data-based parallelism, [Sayed, 2014] describes three paradigms: incremental, consensus and diffusion. *Incremental strategies* assume that the distributed computing system with predefined bandwidth consists of a set of independent but communicating computing nodes. Each node receives an incoming stream of input instances from an external source (e.g., generating source or load balancer/splitter). In the incremental strategy, the learning rule is updated through the succession of M workers that are pre-ordered (as a sequence of neighbours) as shown in Eq. (4).

$$w_{N,t+1} = w_{M-1,t+1} - \alpha_t \nabla_w Q(X_{M,t}, w_t) \qquad (4)$$

Here $X_{i,t}$ indicates the batch of data (of any length) at worker i and w_t is the global model parameters shared by all workers. This however is not fully distributed because of w_t. A better solution would be to replace w_t by $w_{i-1,t}$ to eliminate the communication overhead.
A worker i will then update the parameters as follows:

$$w_{i,t+1} = w_{i-1,t+1} - \alpha_t \nabla_w Q(X_{i,t}, w_{i-1,t+1}) \quad (5)$$

The disadvantage of the incremental strategy is that the order of execution is predefined in away a worker receives the update from one neighbour at a time and passes it to only one other neighbour and more seriously is the workers do not work in parallel.

The consensus strategy partly overcomes these shortcomings by allowing the workers to collect the update from their neighbours and combine with their local gradient:

$$w_{i,t+1} = \sum_{l \in N_i} a_{li} w_{i,t} - \alpha_t \nabla_w Q(X_{i,t}, w_{i,t}) \quad (6)$$

Here a_{li} are user-defined non-negative combination weights such that $\sum_{l=1}^{M} a_{li} = 1$ and $a_{li} = 0, for\ all\ l \notin N_i$. At time t+1, the workers throughout the network run the consensus update simultaneously by using the update amounts from iteration t.

The last strategy, *the diffusion strategy*, is basically a combination of both previous strategies. There are however two ways of implementing the diffusion strategy. The first is called Combine-then-Adapt and is given as follows:

$$w_{i,t+1} = \sum_{l \in N_i} a_{li} w_{i,t} - \alpha_t \nabla_w Q\left(X_{i,t}, \sum_{l \in N_i} a_{li} w_{i,t}\right) \quad (7)$$

The second one is called Adapt-the-Combine which changes the order of the operation in Eq. (7):

$$w_{i,t+1} = \sum_{l \in N_i} a_{li} \left(w_{i,t} - \alpha_t \nabla_w Q\left(X_{i,t}, w_{l,t}\right)\right) \quad (8)$$

In PROTEUS, we will investigate and evaluate a number of scenarios to identify the best variant based on the characteristics of streams and eventually the hardware infrastructure. **¡Error! No se encuentra el origen de la referencia.** below illustrates a generic architecture that can be instantiated in different ways. For instance, input can be processed over many workers according to three schemes: one sample by one sample, batch by batch, or in the case of multivariate streams, a vertical distribution can be adopted (each data stream will be sequentially processed independently). Model building can be local at the individual workers, global by merging the sub-models built by the individual workers, or both, local followed by global. The global model can be replicated or not. For all these alternatives, appropriate and efficient algorithmic approximations will be devised.

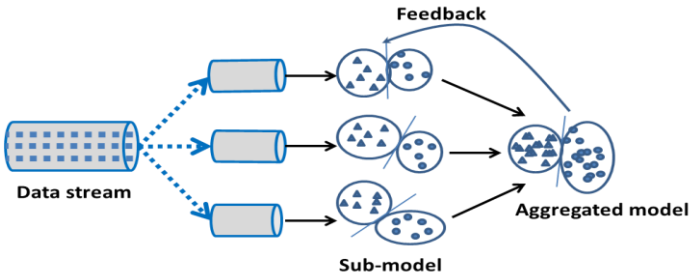

Figure 2: Generic computational model

8. Samples of candidate ML algorithms

In the following we highlight some of the potential ML techniques and algorithms that we will investigate in the context of PROTEUS. These are however not exhaustive and a priority may be set later during the development stage.

8.1. Deep learning

Deep learning (DL) is becoming very popular. Restricted Boltzmann Machines (RBMs) and Deep Belief Networks (DBNs) are trained using batch-based approach and can be trained online using stochastic gradient descent. However, to the best of our knowledge, there is no attempt to use online learning for training DL algorithms, except the work by [Chen et al., 2015].

On the other hand, there have been some investigations on how DBNs can be implemented on a distributed architecture [Dean et al., 2012] where a distributed version of Deep Learning is implemented on Spark. However, there have been no attempts to develop an online distributed learning architecture for DBNs in general and in particular for DBNs for non-stationary setting with concept drift. Many of the offline serial algorithms described in [Bengio, 2009] and the references cited therein can be adapted to the scalable online distributed setting.

8.2. Bayesian non-parametric models

Bayesian non-parametric models have gained a lot of interest due to their flexibility and their versatility in various domains such as density estimation (e.g., clustering), classification, regression, and sequential modelling. Bayesian non-parametric models are interesting because of their capability to find the appropriate learning model. Due to their computational requirements and their relevance for many domains that could involve online learning, scalable inference algorithms for Bayesian non-parametric models are very welcome.

There have been recently some attempts to develop online learning for Dirichlet process mixture models which are non-parametric models [Lin, 2011][Nott et al., 2013][Sato, 2005] and other attempts to implement them in parallel [Doshi-Velez et al., 2009]. It would be a great contribution to develop online scalable Bayesian non-parametric models that can cope with streaming data. In fact one way of doing this is to adopt the distributed stochastic gradient descent approach already discussed earlier.

8.3. Support vector machines and kernel methods

One of the most successful ML techniques is support vector machines (SVMs). There is an extensive body of research on SVMs. Some of it is dedicated to incremental and online learning [Diehl and Cauwenberghs, 2003][Kivinen et al. 2010][Tax and Laskov, 2003][Bordes et al, 2005]. Various approaches have been proposed, but the simplest is to retrain SVM whenever new data item arrives and is poorly classified using the existing hypothesis as a starting point. Different strategies can be deployed to update the support vectors used for the re-training. [Bordes et al, 2005] in particular developed an online SVM based on the sequential minimal optimisation algorithm after reviewing some of the kernel perceptron algorithms.

There exist also some of the parallel and distributed implementations of SVMs [Chang et al., 2007][Zhu et al., 2009]. For instance, in [Chang et al. 2007], a primal-dual interior-point method (IPM) was adopted. The parallelisation usually happens at the level of matrices involved in IPM involving a parallel row-based Incomplete Cholesky Factorization (ICF). [Zhu et al., 2009], on the other hand, use the popular stochastic gradient descent to parallelise the primal SVM objective. Still there are not many parallel implementations of SVMs.

Interestingly enough, to the best of our knowledge, no distributed online SVM exists yet. It would be worth developing SVMs (and kernelised methods in general) that operate online and scale well.

8.4. Semi-supervised and active learning

Semi-supervised learning and active learning are usually used to learn from few and/or actively labelled data, combined with large amount of unlabelled data. While much the semi-supervised learning research has been devoted to the batch learning case [Bouchachia and Pedrycz, 2006] [Ben-David et al. 2008][Yuanyuan et al. 2010][Krijthe and Loog, 2016], only few investigations have been devoted to the online case [Bouchachia and Vanaret, 2014][Zhang et al., 2013]. More importantly, the scalability of such algorithms has not been studied.

Similar situation is encountered with active learning. Recently few studies have been focussing on active learning in the context of online and streaming [Mohamad et al., 2016][Zliobaite et al., 2014][Chu et al., 2011][Slonim et al. 2011][Yang et al., 2014]. Again, to the best of our knowledge no study has investigated distributed implementation of active online learning. Moreover a new concept has been introduced aiming at generalising active learning is called machine teaching [Zhu, 2015] and it is worthwhile to investigate this concept in the context of the online setting.

8.5. Change detection

The relevance of change detection in streaming applications is quite straightforward as it aims at identifying the any change that occurs. In general, change refers to many concepts such as concept drift [Gama et al., 2014], novelty detection [Pimentel et al., 2014] [Marsland, 2003], and anomaly detection [Chandola et al., 2009]. Each of these has had great attention by different research communities. Generally speaking, a change corresponds to change in the underlying probability distribution of the data. The goal is therefore to identify the deviation of the model at hand by monitoring its behavior in real-time. Thus, to deal with changes in the context of streaming, there should be a need to mechanisms of detection and verification that allow distinguishing noise from real change.

While concept drift has been the focus of many studies in the context of streaming data, please a dedicated recent survey in [Gama et al., 2014] for that, where detection methods are categorised into: sequential analysis, control charts, monitoring of distributions, contexts. These are however can be applied to all types of change.

From the novelty perspective, [Pimentel et al., 2014] provided a different classification: probabilistic (parametric and non-parametric such as mixture models, space-based models, kernel densitiy, etc.), distance-based (k-NN and clustering), reconstruction-based approaches (neural networks and sub-space techniques), domain-based models (one-class SVM) and information-theoretic methods. On the other hand, [Chambola et al., 2009] discussed similar categorisation: classification-based methods, clustering-based methods, non-parametric models, spectral methods and information-theoretic methods.

Many of the change detection techniques operate online, but very little has been done to understand the scalability of such techniques in terms of coping with high dimensional data streams and these techniques can be implemented on big data platforms.

9. SOLMA design

The set of scalable online machine learning algorithms that will be developed will be collected in a library, called SOLMA standing for "Scalable Online Learning Massive Analysis" that will be sitting on top of Apache Flink. However, we will take a progressive approach to do that. In the first stage, SOLMA will consist of two main components: the algorithms and the utilities. At a later stage, a graphical interface will be developed along with a workflow scheduler. We aim at ensuring that SOLMA is modular and easy to use with transparent signature for each algorithm. A complete decoupling from the data will be ensured. Note that SOLMA will be published on GitHub[2] for public access.

In terms of content, as mentioned earlier, PROTEUS will concentrate in the first place on algorithms pertaining to Section 8. However, enhanced versions of the ones from Section 2-7, especially those related to sampling, sketching, mining and clustering, will be implemented as well.

10. Conclusions

The present report presents an overview of machine learning and data mining techniques that have been proposed in the literature. Despite the broad coverage it is far from being exhaustive. It is meant to draw the main research being done in the area of data streams analytics. The report also provides some of the potential research directions that PROTEUS will take as it is mainly guided by the novelty and the scientific impact. The design of the library is also briefly discussed.

[2] https://github.com/open-source

References

C. Aggarwal, P. Yu. “A survey of synopsis construction in data streams”. In Data Stream: Models and Algorithms, 2007.

S. Jeffrey "Random sampling with a reservoir" in ACM Transactions on Mathematical Software 11 (1): 37–57. 1985.

D. E. Knuth. “Seminumerical Algorithms”, in volume 2 of The Art of Computer Programming. Addison Wesley, Reading, MA, 2nd edition, 1981.

F. Olken. “Random Sampling from Databases” PhD thesis, Berkeley, 1997.

G. Frahling, P. Indyk, and C. Sohler. “Sampling in dynamic data streams and applications”. In Symposium on Computational Geometry, June 2005.

P. Gibbons. “Distinct sampling for highly-accurate answers to distinct values queries and event reports”. In International Conference on Very Large Data Bases, 2001.

P. B. Gibbons and Y. Matias. “New sampling-based summary statistics for improving approximate query answers”. In SIGMOD, pages 331–342, 1998.

N. Duffield, C. Lund, and M. Thorup. “Estimating flow distributions from sampled flow statistics”. In Proceedings of ACM SIGCOMM, 2003.

R. Gemulla and W. Lehner. “Sampling time-based sliding windows in bounded space”. In SIGMOD, pages 379–392, 2008.

G. Cormode et S. Muthukrishnan, K.Yi, Q. Zhang. “Continuous Sampling from Distributed Streams”. In PODS, pages 77–86, 2010.

B. Babcock, M. Datar, R. Motwani. “Sampling from a moving window over streaming data”. SODA '02 Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, 2002.

V. Braverman, R. Ostrovsky, and C. Zaniolo. “Optimal sampling for sliding windows”. In PODS, 2009.

R. Gemulla and W. Lehner. „Sampling time-based sliding windows in bounded space” In SIGMOD, pages 379–392, 2008.

M. Datar, A. Gionis, P. Indyk, and R. Motwani. “Maintaining stream statistics over sliding windows”. In Proc. of SODA, 2002 .

X. Lu. W. Tok, C. Raissi, S. Bressan, “A Simple yet Effective and Efficient, Sliding Window Sampling Algorithm”, Database Systems for Advanced Applications. Volume 5981 of the series Lecture Notes in Computer Science pp 337-35, 2010.

S. Singh, S. Tirthapaura, “An evaluation of streaming algorithms for distinct counting over a sliding window”, In ICT, 201.

A. Bifet,. "Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams." Proceedings of the 2010 conference on adaptive stream mining: Pattern learning and mining from evolving data streams. IOS Press, 2010.

J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. "A survey on concept drift adaptation." In ACM Comput. Surv. 46, 4, 2014.

N. Alon, Y. Matias. M. Szegedy "The Space Complexity of Approximating the Frequency Moments" in ACM Symposium on Theory of Computing, pp. 20–29, 1996.

K. Weinberger, A. Dasgupta J. Attenberg, J. Langford, A. Smola, "Feature Hashing for Large Scale Multitask Learning", ICML '09 Proceedings of the 26th Annual International Conference on Machine Learning, 2009.

S. Guha, N. Koudas, K. Shim, "Approximation and streaming algorithms for histogram construction problems". In Journal ACM Transactions on Database Systems TODS 2006.

W.Jung, J. Park, J Lee, "Versatile and Scalable Parallel Histogram Construction". In PACT '14 Proceedings of the 23rd international conference on Parallel architectures and compilation, 2014.

U. Milic, I. Gelado, N. Puzovic, A. Ramirez, M. Tomasevic. "Parallelizing General Histogram Application for CUDA Architectures". In Proceedings of the IEEE International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation. IEEE. June, 2013.

C. Demetrescua, B. Escoffierb G. Moruz, "Adapting parallel algorithms to the W-Stream model, with applications to graph problems", In Theoretical Computer Science, 2011.

P.B. Gibbons, Y. Yossi Matias, and V. Poosala. "Fast incremental maintenance of approximate histograms". In VLDB. Vol. 97 1997.

M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining stream statistics over sliding windows". In Proceedings of Annual ACMSIAM Symposium on Discrete Algorithms, San Francisco, USA, pp. 635–644. Society for Industrial and Applied Mathematics. 2002.

G.J Ross "High Performance Histogramming on Massive Parallel Processers" In Ms. Thesis, University of Illinois, 2014.

G. Cormode, M. Garofalakis, P.J. Haas. C. Jermaine, "Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches", In Foundations and Trends in Database. 2011.

S. Chatterje. "Parallel and Distributed computing PRAM Algorithms", in lecture notes, Yale University 2002.

S. Muthukrishnan, M. Strauss, "Approximate Histogram and Wavelet Summaries of Streaming Data" In DIMACS Tech Report 2004.

A. C. Gilbert, Y. Kotidis, S. Muthukrishnan and M. J. Strauss, "One-pass wavelet decomposition of data streams," IEEE transactions on knowledge and data engineering, Vol. 15, No. 3, May/June 2003.

A. C. Gilbert, Y. Kotidis, S. Muthukrishnan and M. J. Strauss, "Surfing wavelets on streams: one-pass summaries for approximate aggregate queries," In Proceedings of the 27th VLDB Conference, Roma, Italy 2001

A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan and M. J. Strauss, "Fast, small-space algorithms for approximate histogram maintenance," In STOC ’02, May 19- 21, 2002, Montreal, Quebec, Canada

R.M. Karp “An introduction to randomized algorithms” In Discrete Applied Mathematics, 34:165-201, 1991.

R. Motwani and P Raghavan “Randomized Algorithms” In Cambridge University Press 1995.

J. Yang, X. Meng, M. Mahoney. “Implementing Randomized Matrix Algorithms in Parallel and Distributed Environments”, In PIEEE, 2016.

R. Xin, “Sketching Big Data with Spark: randomized algorithms for large-scale data analytics”, In Strata NYC 2015.

P. Domingos and G. Hulten. “Mining high-speed data streams” In KDD ’00, 2000.

G. Hulten, L. Spencer, and P. Domingos. “Mining time-changing data streams” in KDD ’01, 2001.

Y. Ben-Haim, E. Tom-Tov: “A Streaming Parallel Decision Tree Algorithm”. In JMLR 2010.

A. Murdopo, A. Bifet, G. De Francisci Morales, N. Kourtellis: “VHT: Vertical Hoeffding Tree”. Working paper, 2014.

C. Cortes, and V. Vapnik, "Support-vector networks" In Machine Learning 20 (3): 273, 1995.

P. Rai, h. Daumé III, S. Venkatasubramanian, “Streamed Learning: One-Pass SVMs”, In IJCAI 2009

P. Kranen., I. Assent, C. Badaluf, T. Seidl, “Self-Adaptive Anytime Stream Clustering (ClusTree) IEEE International Conference Data Mining, 2009.

C. Aggarwal, J. Han, J. Wang, and P. Yu. “A Framework for Clustering Evolving Data Streams”. In VLDB Conference, 2003.

F. Cao, M. Ester, W. Qian, A. Zhou: “Density-Based Clustering over an Evolving Data Stream with Noise”. In SDM ’06, 2006.

P. Kranen, “Anytime Algorithms for Stream Data Mining”, PhD Thesis, 2011

H: Zhinoos, T. Sellis, X. Zhang, “Anytime Concurrent Clustering of Multiple Streams with an Indexing Tree” In Proceedings of the 4th International Workshop on Big Data, 2015

T. Zhang, R. Ramakrishnan, M. Livny. “BIRCH: An Efficient Data Clustering Method for Very Large Databases”. In ACM SIGMOD Conference, 1996.

C. Nadungodage, Y. Xia, F. Li, J. Lee, J. Ge. “StreamFitter: a real time linear regression analysis system for continuous data streams”. In Proceedings of the 16th international conference on Database systems for advanced applications: Part II, pp: 458-461, Springer-Verlag. 2011

J. Dong, F. Wang, B. Yuan. “Accelerating BIRCH for Clustering Large Scale Streaming Data Using CUDA Dynamic Parallelism”, Intelligent Data Engineering and Automated Learning – IDEAL 2013

M. Ester, H Kriegel, J. Sander, X. Xu: “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In KDD ’96, 1996.

T. Sun, C. Shu, F. Li, H. Yu, L. Ma, Y. Fang. “An Efficient Hierarchical Clustering Method for Large Datasets with Map-Reduce”. In PDCAT 2009.

H. Gao, J. Jiang, L. She, Y. Fu. “A New Agglomerative Hierarchical Clustering Algorithm Implementation based on the Map Reduce Framework”. In International Journal of Digital Content Technology and its Applications, 2010.

F. Troyano, J. Aguilar-Ruiz, and J. Riquelme. “Discovering decision rules from numerical data streams”. In ACM Symposium on Applied computing, pages 649–653, 2004.

F. Troyano, J. Aguilar-Ruiz, and J. Riquelme “Data streams classification by incremental rule learning with parameterized generalization”. In ACM Symposium on Applied Computing, pages 657–661, 2006.

E. Almeida, C. Ferreira, J. Gama. "Adaptive Model Rules from Data Streams." In ECML-PKDD ’13, 2013.

A. T. Vu, G. De Francisci Morales, J. Gama, A. Bifet: “Distributed Adaptive Model Rules for Mining Big Data Streams”. In BigData ’14, 2014.

R. Agrawal, T. Imieliński, and A. Swami “Mining association rules between sets of items in large databases" SIGMOD Rec. 22, 2 1993.

M. Baldi, E. Baralis, F. Risso, “Data mining techniques for effective and scalable traffic analysis” in: Integrated Network Management, Ninth IFIP/IEEE International Symposium, 2005.

M. Zaki. “Parallel and distributed association mining: A survey”. In IEEE Concurrency, pages 14–25, 1999.

J. Li, Y. Liu, W.-k. Liao, and A. Choudhary. “Parallel data mining algorithms for association rules and clustering”. In Intl. Conf. on Management of Data, 2008.

E. Ozkural, B. Ucar, and C. Aykanat. “Parallel frequent item set mining with selective item replication” In IEEE Trans. Parallel Distrib. Syst., pages 1632–1640, 2011.

B. Park and H. Kargupta. “Distributed data mining: Algorithms, systems, and applications”. In Data Mining Handbook, 2002.

L. Zeng, L. Li, L. Duan, K. Lu, Z. Shi, M. Wang, W. Wu, and P. Luo. “Distributed data mining: a survey” in Information Technology and Management, pages 403–409, 2012.

D. Pott and C. Sammut, “Fast Incremental Learning of Linear Model Trees” in Machine Learning 61, 2005.

E. Ikonomovska, J. Gama, S. Džeroski: “Learning model trees from evolving data streams”. Data Mining and Knowledge Discovery 23(1), 128–168, 2011.

C. Xu, Y. Zhang, R. Li, “On the Feasibility of Distributed Kernel Regression for Big Data”. In arXiv:1505.00869v1, 2015.

A. Shaker and E Hullermeier. “IBLStreams: A System for Instance-Based Classification and Regression on Data Streams”. In Evolving Systems 3, 2015.

C. E. Rasmussen & C. K. I. Williams, “Gaussian Processes for Machine Learning”, in MIT Press, 2006.

D. Tuong, J. Seeger, “Local Gaussian Process Regression, for Real Time Online Model Learning and Control” in NIPS 2009.

C. Park, J. Huang, Y. Ding, “GPLP: A Local and Parallel Toolbox for Gaussian Process Regression” in Journal of Machine Learning Research, 13, 2011.

R. Conlin, “Computationally Efficient Gaussian Process Changepoint Detection and Regression” PhD Thesis MIT, 2014.

S. Si, C. Hsieh and I. Dhillon “Memory Efficient Kernel Approximation” in Proceedings of The 31st International Conference on Machine Learning, pp. 701–709, 2014.

J Han, J, Pei, P. Yu, “Mining frequent pattern without candidates”, SIGMPD 2000.

C. Giannella, J. Han, J. Pei, X. Yan, P. S. Yu: “Mining frequent patterns in data streams at multiple time granularities”. In NGDM 2003.

H Li, Y. Wang, D Zhang, M.Zhang, E. Chang, “Pfp: Parallel fp-growth for query recommendation” in RecSys 08, 2008.

L. Zhou, Z. Zhong, J. Chang, J. Li, J. Huang, and S. Feng. “Balanced parallel FP-Growth with MapReduce”. In YC-ICT, 2010.

L. Zhou, X. Wang. “Research of the FP-Growth Algorithm Based on Cloud Environments” In .Journal of Software, 2014.

“Available algorithms identification” ONTIC Project (GA number 619633), Jan 2015

R.Agrawal and R. Srikant “Fast algorithms for mining association rules in large databases” In Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile, September 1994.

R.Bayardo, "Efficiently mining long patterns from databases" (PDF). In ACM Sigmod Record. 27, 1998.

S. Moens, E. Aksehirli, and B. Goethals. “Frequent itemset mining for big data” In BigData 2013 Workshop on Scalable Machine Learning. IEEE, 2013.

M.-Y. Lin, P.Y. Lee, and S.C. Hsueh. “Apriori-based frequent itemset mining algorithms on MapReduce” In ICUIMC, 2012.

M. Riondato, Justin A. De Brabant, Rodrigo Fonseca,: “PARMA: a parallel randomized algorithm for approximate association rules mining in MapReduce”. In ACM, 2012.

J. Cheng, Y. Ke, W. Ng: “A Survey on Algorithms for Mining Frequent Itemsets over Data Streams”. In Knowledge and Information Systems, Volume 16, Number 1, July, 2008.

Y. Chi , H. Wang, P. Yu , R. Muntz: “Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window”. In ICDM ’04, 2004.

N. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression". In The American Statistician 46 (3): 175–185, 1992.

R. Klinkenberg, T. Joachims. “Detecting concept drift with support vector machines”. In ICML Conference, pages 487–494, 2000.

N. Syed, H. Liu, and K. Sung. “Handling concept drifts in incremental learning with support vector machines”. In ACM KDD Conference, pages 317–321, 1999

L. Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In Proc. of COMPSTAT, page 177-188, 2010.

S. Shalev-Shwartz. “Online Learning and Online Convex Optimization” In Foundations and Trends in Machine Learning: Vol. 4: No. 2, pp 107-194, 2012.

F. Orabona, K. Crammer, N. Cesa-Bianchi. “A generalized online mirror descent with applications to classification and regression” In Machine Learning 99(3): 411-435 2015.

A. Bouchachia, C. Vanaret. “GT2FC: An Online Growing Interval Type-2 Self-Learning Fuzzy Classifier” In IEEE Transactions on Fuzzy Systems, (22(4): 999-1018, 2014.

N. Liang, G. Huang, P. Saratchandran, N. Sundararajan. “A fast and accurate on-line sequential learning algorithm for feedforward networks”. In IEEE Trans Neural Networks, 17(6):1411–1423, 2006.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. “Fast kernel classifiers with online and active learning” In The Journal of Machine Learning Research, Volume 6, Pages 1579-1619, 2005.

P. Laskov, C. Gehl, S. Krueger, K.-R. Mueller. “Incremental support vector learning: Analysis, implementation and applications”. In Journal of Machine Learning Research, 7:1909–1936, 2006.

N. Cesa-Bianchi and G. Lugosi. “Prediction, Learning, and Games”. In Cambridge University Press, 2006.

J. Kivinen, M.K. Warmuth. “Relative loss bounds for multidimensional regression problems”. Machine Learning, 45(3):301-329, 2001.

S.M. Kakade, S. Shalev-Shwartz, and A. Tewari. “Regularization techniques for learning with matrices”. In The Journal of Machine Learning Research, Volume 13, Issue 1, January 2012.

S. Shalev-Shwartz and Y. Singer. "A primal-dual perspective of online learning algorithms". Machine Learning Journal, Volume 69, Issue 2, pp 115-142, 2007.

Y. Zhang, J. Duchi, M. Jordan, and M. Wainwright. "Information-theoretic lower bounds for distributed statistical estimation with communication constraints". In Advances in Neural Information Processing Systems 27, 2013.

Y. Zhang, J. Duchi, and M. Wainwright. "Communication-efficient algorithms for statistical optimization". In J. Mach. Learn. Res. 14, 3321-3363. 2013.

M. Balcan, A. Blum, S. Fine, Y. Mansour. "Distributed learning, communication complexity and privacy". In Journal of Machine Learning Research - Proceedings Track, 23:26.1–26.22, 2012.

S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein. "Distributed optimization and statistical learning via the alternating direction method of multipliers", In Foundations and Trends in Machine Learning, 3, 2011.

J. Duchi, A. Agarwal, M. Wainwright. "Dual averaging for distributed optimization: Convergence analysis and network scaling". In IEEE Transactions on Automatic Control, 57(3):592–606, 2012.

O. Dekel, R. Gilad-Bachrach, O. Shamir, L. Xiao. "Optimal distributed online prediction using mini-batches". Journal of Machine Learning Research, 13:165–202, 2012.

B. Recht, C. Re, S. Wright, F. Niu "Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent". In Advances in Neural Information Processing Systems, pages: 693—701, 2011.

D. Bertsekas, J. Tsitsiklis. "Parallel and Distributed Computation: Numerical Methods". In Athena Scientific, Belmont, MA, 1997.

O. Shamir. "Fundamental limits of online and distributed algorithms for statistical learning and estimation." In Advances in Neural Information Processing Systems (NIPS), 2014.

C. Tekin, M. van der Schaar. "Distributed Online Learning via Cooperative Contextual Bandits". In IEEE Transactions on Signal Processing 63(14): 3700-371. 2015.

J. Langford, A. J. Smola, , M. Zinkevich. "Slow learners are fast". In Advances in Neural Information Processing Systems, pages: 2331-2339, 2009.

A. H. Sayed, "Adaptation, learning, and optimization over networks," in Foundations and Trends in Machine Learning, vol. 7, pp. 311–801. NOW Publishers, Boston-Delft, Jul. 2014.

M. Zinkevich, M.Weimer, A. Smola, L. Li, "Parallelized stochastic gradient descent". In NIPS, pages 2595–2603, 2010.

G. Mann, R. McDonald, M. Mohri, N. Silberman, and D. Walker. "Efficient large-scale distributed training of conditional maximum entropy models". In Advances in Neural Information Processing Systems 22, pages 1231–1239. 2009.

G. Chen, R. Xu, and S. Srihari. "Sequential Labeling with online Deep Learning". In arXiv:1412.3397, 2015.

J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, and A. Ng. “Large scale distributed deep networks” In NIPS, pp. 1232–1240, 2012.

Y. Bengio. “Learning deep architectures for AI”, in Foundations and Trends in Machine Learning, 2(1), 1–127, 2009.

D. Lin. “Online learning of nonparametric mixture models via sequential variational approximation.” In Advances in Neural Information Processing Systems 26, 2013.

D. Nott, X. Zhang, C. Yau, and A. Jasra. “A sequential algorithm for fast fitting of Dirichlet process mixture models”. In Arxiv: 1301.2897, 2013.

F. Doshi-Velez, D. Knowles, S. Mohamed, and Z. Ghahramani. “Large scale non-parametric inference: Data parallelisation in the Indian buffet process.” In Advances in Neural Information Processing Systems 23, pages 1294-1302, 2009.

M. Sato. “Online model selection based on the variational Bayes”. In Neural Computation, 13(7):1649–1681, 2005.

C. Wang, J. Paisley, and D. Blei. “Online variational inference for the hierarchical Dirichlet process”. In Artificial Intelligence and Statistics, 2011.

D. Tax, P. Laskov, “Online SVM learning: from classification to data description and back”, in: Proceedings of IEEE International Workshop on Neural Networks for Signal Processing, 2003.

C. P. Diehl and G. Cauwenberghs, “SVM incremental learning, adaptation and optimization”, In Proceedings of the International Joint Conference on Neural Networks, 2003.

E. Chang, K. Zhu, H.Wang, H. Bai, J. Li, Z. Qiu, and H. Cui. “PSVM: Parallelizing support vector machines on distributed computers”. In Advances in Neural Information Processing Systems, 20:16, 2007.

Z. A. Zhu, W. Chen, G. Wang, C. Zhu, and Z. Chen, “P-packSVM: Parallel primal gradient descent kernel SVM,” in ICDM, 2009.

A. Bouchachia, W. Pedrycz: “Enhancement of fuzzy clustering by mechanisms of partial supervision”. Fuzzy Sets and Systems 157(13): 1733-1759, 2006.

S. Ben-David, T. Lu, and D. Pl. “Does unlabeled data provably help? Worst-case analysis of the sample complexity of semi-supervised learning”. In Proceedings of the 21st Annual Conference on Learning Theory (COLT), pages 33-44, 2008.

G. Yuanyuan, X. Niu, and H. Zhang. “An extensive empirical study on semi-supervised learning”. In Proceedings of the 10th IEEE International Conference on Data Mining (ICDM), pages 186-195, 2010.

J. Krijthe, M. Loog. “Projected Estimators for Robust Semi-supervised Classification”. arXiv:1602.07865 February 2016.

A.Bouchachia, C.Vanaret: “GT2FC: An Online Growing Interval Type-2 Self-Learning Fuzzy Classifier”. IEEE T. Fuzzy Systems 22(4): 999-1018, 2014.

G. Zhang, Z. Jiang, and L. S. Davis, “Online semi-supervised discriminative dictionary learning for sparse representation,” in Computer Vision ACCV 2012, 2013, pp. 259–273.

S. Mohamad, A. Bouchachia, M. Sayed-Mouchaweh. “Bi-criteria Active Learning for Dynamic Data Streams”, In IEEE transactions on neural networks and learning systems, submitted, 2016.

I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes, “Active learning with drifting streaming data”. In IEEE transactions on neural networks and learning systems, vol. 25, no. 1, pp. 27–39, 2014.

N. Slonim, E. Yom-Tov, and K. Crammer, “Active online classification via information maximization,” in IJCAI, pp. 1498–1504, 2011.

Z. Yang, J. Tang, and Y. Zhang, “Active learning for streaming networked data,” in CIKM, 2014.

W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. Tseng. “Unbiased online active learning in data streams”. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 195–203, 2011.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou, “Fast kernel classifiers with online and active learning,” J. Mach. Learn. Res., vol. 6, pp. 1579–1619, Sep. 2005.

X. Zhu. “Machine Teaching: an Inverse Problem to Machine Learning and an Approach Toward Optimal Education. In The Twenty-Ninth AAAI Conference on Artificial Intelligence (Senior Member Track, AAAI), 2015.

M. Pimentel, D. Clifton, L. Clifton, and L. Tarassenko. “Review: A review of novelty detection”. *Signal Process.* 99 (June 2014), 215-249, 2014.

S. Marsland. “Novelty detection in learning systems”, In Neural. Comput. Surv. 3, 2003.

V. Chandola, A. Banerjee, and V. Kumar. “Anomaly detection: A survey”. ACM Comput. Surv. 41, 3, 2009.