Online Stream Learning for Smartphone-based Human Activity Recognition: An Overview

Ilham Amezzane[1], Youssef Fakhri[1], Mohamed El Aroussi[1], Mohamed Bakhouya[2]

[1] Ibn Tofail University, Kenitra, Morocco
[2] International University of Rabat, Sala Aljadida, Morocco
Ilham.amezzane@uit.ac.ma

Abstract— Smartphone-based Human Activity Recognition (SHAR) applications better performs if implemented in an online manner on continuous data streams. The selection of learning approaches highly impacts the performance of SHAR applications regarding key cost criteria such as: energy, memory and time efficiency. Stream learning is performed to achieve multiple objectives which include enhancements of the internal structures of learning models and their processing behavior. SHAR application enhancements include concept drift detection, tackling class imbalance from uncertain data streams, model personalization and optimization. In this paper, our goal is to present an overview of data stream characteristics; baseline learning algorithms, as well as evaluation approaches and metrics that can be used for online stream learning in the context of SHAR. We also present existing applications that fulfill the stream learning requirements as well as the most popular open source frameworks.

Index Terms—Human activity recognition, smartphone, stream learning, evolving streams.

1 Introduction

Smartphone-based Human Activity Recognition (SHAR) involves the use of different mobile embedded sensors available on modern mobile phones. It also involves Machine Learning (ML) techniques to automatically collect and infer user activities for different domains such as healthcare monitoring, assisted living for elderly, sports and wellbeing applications. In the supervised learning approach, the classifier usually need to be trained offline first (in desktop PCs, servers, or cloud systems) using labeled data, before online implementation where the trained model classifies each block of incoming data as one of the targeted activities within a small time window. This is made possible because data collected from sensors are provided in a time-series manner (data streams).

Typical SHAR approaches are based on static data scenario and assume that data arrives in form of batches and must be processed after such blocks become available. Batch mode offers better robustness to local fluctuations of the stream and a broader outlook on the incoming data [1]. However, increasing attention is being paid to tackling SHAR problem from online streaming perspective. In online streaming mode, incoming samples are processed one by one and the classification system is modified and adapted accordingly to the current state of the stream. This is made possible by making the recent samples more important during the update process. This approach is therefore more vulnerable to local fluctuations, but on the other hand allows for mining the stream on-the-fly. Moreover, it can employ additional mechanism for drift detection and adaption [1]. From the SHAR perspective, the online mode is more suitable than the batch one, as no delay in the decision making process is allowed, especially in sensitive cases such as assisted or elderly supervision [1]. Therefore, incoming data must be classified and the learning model adapted in real-time. Until recently, online stream learning inside mobile environments has been a challenging task for ML techniques considering the limited resources as well as time constraints. The benefit of the online streaming approach is that the training samples do not need to be stored on the phone nor scanned more than one time as they arrive [2]. The main challenges are as follows:

1. Limited computational and memory resources, as well as tight needs to make predictions in reasonable time.
2. Evolving nature of data streams, i.e., the distribution of data and target concepts can change over time. This could dramatically deteriorate performance of the used model.
3. One scan of incoming objects, i.e., the fact that data elements cannot be accessed multiple times as the new example is immediately discarded, which allows to process high speed data streams.
4. The training procedure can be stopped at any time and the prediction performance should not be worse than the one obtained in batch mode.

These challenges pose the need for different algorithms than those used for classical batch learning where data are stored in finite and persistent data repositories, or those used for incremental learning, as they do not focus on computational restrictions and do not consider dynamic changes [2]. To tackle these challenges, several algorithms have been introduced in the literature.

In the present paper, we present an overview of the main basic algorithms that can be used for online learning in the context of SHAR applications. The remaining of the paper is as follows: In section 2 we describe characteristics and issues of data streams. Section 3 presents the main baseline algorithms used in online supervised stream classification. In section 4, we list evaluation metrics commonly used in stream learning. Section 5 summarizes existing applications in the context of

SHAR. Popular open source frameworks are briefly introduced in section 6 before concluding in section 7.

2 HAR Data Stream Characteristics

2.1 Supervised Learning Framework

In SHAR applications, massive volumes of data are continuously generated in the form of data streams. These streams are potentially unbounded, ordered sequences of items which arrive at constant or variable sampling rates.

SHAR applications use different learning models based on supervised, unsupervised, semi-supervised and deep learning approaches. We focus in the present paper on the supervised learning models where the training data stream needs to be labeled so that learning model is able to predict the future similar data streams accurately. A data stream is the sequence of data instances (X^t, y^t) as presented in Fig. 1, arriving one at a time drawn from an unknown probability distribution pt (X, y); for time t= 1,2,3,...,T. where X is the set of attributes and y is the corresponding class label. We assume that an online classifier F receives the new input X^t at time step t and then predicts its class label. After some time, actual class label y^t is available and is used by F to evaluate the predictive performance and to provide additional information for training update. This whole process will be repeated at following time steps. This technique is used by most of supervised classification algorithms.

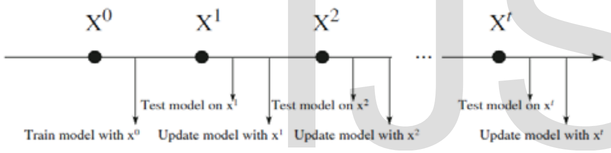

Fig. 1 Online stream processing [4]

Three different ways of annotating data streams exist: manual, automatic, and observational. The manual labeling process of each chunk of the data stream is quite laborious and time-consuming. Automatic labeling is performed through application configuration at the time of data collection. Therefore, automatic labeling is more comfortable compared to manual approach. In the observational method, the learning models are initially trained in automatic settings, however in the case of discrepancies users are allowed to intervene by manually labeling the data streams [5].

2.2 Concept Drifts

As data evolves over time, the concept about which data is collected may change. This phenomenon is called Concept Drift [6]. In the presence of drifts, the fundamental paradigm of traditional data mining does not hold anymore since it assumes a static and unknown distribution of training and testing samples. Therefore, it is crucial to monitor the appearing concept drifts in order to adapt the model to the changes accordingly. Moreover, an important characteristic of concept drift relates to the rate at which it happens. The rate of a drift can be sudden or gradual. The first type of drift occurs when a source distribution of data stream is suddenly replaced by another source distribution. The later type of drift is associated with slower rate of changes in data streams [4]. Multiple algorithms have been proposed for dealing with concept drifts in data streams as we will see in section 3.

2.3 Class Imbalance

Class imbalance occurs when some target classes are not equally represented. The difficulty in learning from imbalanced data is that the under-represented class cannot draw equal attention to the learning algorithm. This can often cause learning bias towards the highly-represented class, inducing poor generalization for future prediction [3]. Since human activities are generally imbalanced, learning difficulty in the context of SHAR is certainly increased. On the other side, class imbalance has attracted growing attention in data stream learning in recent years and different algorithms have been introduced to tackle this problem [3].

3. Stream Learning Algorithms

Two basic models of data streams exist: stationary, where examples are drawn from a fixed although unknown probability distribution, and non-stationary, where data can evolve over time. Nonetheless, two distinct approaches exist in research works: Active approaches which trigger changes in classifiers when drifts are detected; and passive approaches which continuously update the classifier regardless of appearing drifts occurring in the data stream [4].

3.1 Stationary Stream Learning

3.1.1 Hoeffding Tree

Hulten and Domingos [7] proposed the Hoeffding Tree (HT) algorithm presented in Fig. 2. They also refer to their implementation as *VFDT*, a Very Fast Decision Tree learner. In that paper, the HT is the basic theoretical algorithm, while VFDT introduces several enhancements for practical implementation. The HT novelty consisted of waiting for new instances to arrive instead of reusing instances as it is the case for batch learning approaches. The crucial decision needed to construct a decision tree is when to split a node, and with which example-discriminating test.

The most popular criteria for selecting decision tree split tests is 'information gain' (*G*), which measures the average amount of 'purity' that is gained in each subset of a split [8]. The strength of the HT method is that it has theoretical proofs guaranteeing that HT algorithm can build trees of the same quality as batch learned trees, if the number of instances needed at a node when selecting a splitting attribute, is sufficient enough. The name of this algorithm is derived from the hoeffding bound, which states that with probability *1−δ*, the true mean of a random variable of range *R* will not differ from the estimated mean after *n* independent observations by more than *ε*, as defined in (1) [8]:

$$\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}} \tag{1}$$

This bound justifies that a small sample can often be enough to choose an optimal splitting attribute. For any given potenti al split, HT checks whether the difference of averaged information gains of the top two attributes is likely to have a positive mean—if so, the winning attribute may be picked with a certain degree of confidence. Thus the HT algorithm can deter-

```
1:  Let $HT$ be a tree with a single leaf (the root)
2:  for all training examples do
3:    Sort example into leaf $l$ using $HT$
4:    Update sufficient statistics in $l$
5:    Increment $n_l$, the number of examples seen at $l$
6:    if $n_l \; mod \; n_{min} = 0$ and examples seen at $l$ not all of same class then
7:      Compute $\overline{G}_l(X_i)$ for each attribute
8:      Let $X_a$ be attribute with highest $\overline{G}_l$
9:      Let $X_b$ be attribute with second-highest $\overline{G}_l$
10:     Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n_l}}$
11:     if $X_a \neq X_\emptyset$ and $(\overline{G}_l(X_a) - \overline{G}_l(X_b)) > \epsilon$ or $\epsilon < \tau$ then
12:       Replace $l$ with an internal node that splits on $X_a$
13:       for all branches of the split do
14:         Add a new leaf with initialized sufficient statistics
15:       end for
16:     end if
17:   end if
18: end for
```

Fig. 2. Hoeffding Tree algorithm

-mine, with high probability, the smallest number n of examples needed. Moreover, HT contributes in solving the uncertainty in learning time since it consumes constant time per instance.

3.1.2 Very Fast Decision Trees (VFDT)

Domingos et al. [7] modified the HT algorithm with an improved method called VFDT, with the following characteristics [9]:

1. Ties: when two attributes have similar split gain, the improved method splits if the Hoeffding bound computed is lower than a certain threshold parameter τ.
2. Speed: instead of computing the best attributes to split every time a new instance arrives, it computes them every time a number n_{min} of instances has arrived.
3. Memory utilization: it deactivates the least promising nodes at the time of low memory and drops the poor splitting attributes.

HT can grow slowly and performance can be poor initially, so this extension provides an immediate boost to the learning curve [9]. Although not suitable to handle drifts, HT or VFDT are used as a base for many state-of-the-art drift learners. Enhancements to the basic VFDT algorithm include methods of limiting memory usage, the use of alternative bounds which requires less examples for each split node, approaches to dealing with numerical attributes, pruning mechanisms, and the use of sliding windows or drift detectors to adapt the algorithm to non-stationary settings [4].

3.2 Evolving Stream Learning

Most stream classification algorithms are capable of predicting, detecting, and adapting to concept drifts. Drift adaptation algorithms generally require data management and forgetting mechanisms in order to deal with time-changing streams, so that they are relevant to the most recent data. This usually takes the form of a sliding window that forgets older examples or a fading factor that decays the weight of older examples [4].

3.2.1 Concept-adapting Very Fast Decision Trees (CVFDT)

Hulten, Spencer, and Domingos [10] tackled the problem of drift adaptation with an algorithm called Concept-adapting Very Fast Decision Tree (CVFDT). CVFDT uses a sliding window system over VFDT. Unlike traditional batch learning, it does not construct a new model each time from the beginning. After a fixed number of new instances arrive, the relevant statistics at every node are updated; and the Hoeffding bounds are recomputed [11]. If the concept is changing, a better splitting attribute is found as it has a higher gain and then a new subtree is learned. The algorithm then waits for more instances in order to confirm that the new learned subtree is of better quality than the original one, which is then replaced [11]. This technique is dependent on the window size. The size should be neither too small, in order to store enough examples to construct an accurate model, nor too large, in order to represent the concept accurately and continuously.

3.2.2 Hoeffding Adaptive Tree (HAT)

The Hoeffding Adaptive Tree (HAT) is an adaptive extension to the HT that has theoretical guarantees and uses the ADWIN algorithm [12] as a change detector and error estimator. ADWIN is parameter- and assumption-free in the sense that it automatically detects and adapts to the current rate of change. Especially, ADWIN keeps a variable length window of length W, but does not maintain the window explicitly; instead it compresses it using only O(logW) memory and O(logW) processing time per item, rather than the O(W) expected from a naive implementation [12]. It is therefore interesting to know the main contributions of HAT with respect to CVFDT such as: 1) The alternate trees are created as soon as change is detected, without having to wait after the change, for a fixed number of new examples; 2) HAT replaces the old trees by the new alternate trees as soon as there is evidence that they are more accurate [12].
It can be said that the HAT adapts to the scale of time change in the data, rather than having to rely on the a priori guesses made by the user. In the case of noisy distributions with outliers, the HAT and CVFDT will not fluctuate abruptly, since they compute mean values that helps to smooth the computation [12].

3.2.3 Incremental Algorithms

Several traditional incremental classifiers were also adapted to concept drift requirements. Sliding windows are usually employed as a forgetting mechanism. For example, the k Nearest Neighbors (kNN) algorithms are naturally transformed to incremental versions for use in the streaming case with different techniques for selecting the limited subset of the most useful examples for accurate predictions. The kNN classifier is updated when concept drift is detected by discarding obsolete examples from the knowledge base [2]. Neural Networks can also be adapted to evolving data streams by dropping the epoch protocol and presenting examples in one pass [2]. Other algorithms use a structure similar to a decision tree to create rule-specific drift detectors [2].

3.2.4 Tackling Class Imbalance and Concept Drift Simultaneously

Few works handling concept drift in class-imbalanced data streams exist in the literature. Actually, most existing concept drift detection methods are only designed for balanced data streams. Since data streams in SHAR application scenario naturally exhibit both phenomenons, we mention here a recent interesting systematic study that tackles the combined challenge of concept drift in class-imbalanced data streams [3]. Authors assume that most existing papers that proposed new concept drift detection methods for imbalanced data did not consider the effect of class imbalance techniques on final prediction and concept drift detection. In their paper, they provided a thorough review and an experimental insight into this problem. Based on the analysis, general guidelines are proposed for the development of an effective algorithm such as:

1. Is the class imbalance technique effective in predicting minority-class as well as adaptive to class imbalance changes?
2. Is the detection performance of the concept drift technique affected by the class imbalance technique?
3. How to make the class imbalance technique and concept drift technique work together to achieve better online prediction.

4 Evaluation in Data Stream

The data stream classification of SHAR performance is usually assessed using evaluation measures identical to static supervised classification such as accuracy and F1-score. However, unlike batch learning mode, repeated runs to estimate these measures over the streaming data are not possible because of the time and resource constraints. During stream classification, the time required to process individual instances and the average memory usage should be kept constant. For this reason, training and testing time along with model memory size have to be periodically monitored.

4.1 Holdout Evaluation

With no concept drift assumption, a single static held out set should be sufficient for performance estimation as long as it is independent and sufficiently large relative to the complexity of the target concept. Test set sizes on the order of tens of thousands of examples have previously been considered sufficient [8]. To track model performance over time, the model can be evaluated periodically, for example, after every one million training examples. Therefore, periodic holdout method gives a more accurate estimation of the accuracy on more recent data. However, testing the model too often may significantly slow the evaluation process, depending on the size of the test set [8].

4.2 Prequential Evaluation

An alternate scheme of estimating the performance of stream classifiers involves interleaving testing with training. Each individual example is first used to test the classifier before it is used for training (Fig. 1). Interleave train and test, also referred to as prequential (combination of words predictive and sequential), follows the online learning protocol, where data are evaluated as they are collected [8]. Whenever an example is observed, the current model makes a prediction; when the system receives feedback from the environment, the loss function can be computed. Such a procedure highlights the current rather than overall performance, because simply calculating a cumulative measure over the entire stream may lead to strongly biased results and thus, may not detect appearing drifts in the stream.

4.3 Evaluation Metrics

Below we list the most popular existing performance metrics. Most of them should be selected and monitored during evaluation of data stream classification depending on the data stream characteristics:

1. Accuracy: the proportion of all correct predictions to the total number of examples. Prequential accuracy is popularly used with supervised learning.
2. G-Mean: the geometric mean of sensitivity and specificity is often applied on class-imbalanced data streams to avoid the bias of the overall accuracy.
3. Kappa Statistic: $K = p_0 - p_c / 1 - p_c$, where p_0 is the classification accuracy and pc is the probability of a random classifier making a correct prediction.
4. Generalized Kappa Statistics: such as Kappa M for dealing with imbalanced data streams.
5. Prequential AUC: suitable for streams with skewed distributions.
6. Memory consumption: the average memory requirements of each algorithm, and also their change over time.
7. Update time: the amount of time that an algorithm requires to update its structure and accommodate new data from the stream. In an ideal situation, the update time should be lower than the arrival time of a new.
8. Decision time: the amount of time that a model needs to make a decision regarding new instances from the stream. This phase usually comes before the updating procedure takes place.
9. RAM hours: indicating processing time and memory measured in a single metric.

5 Online SHAR Applications

5.1 MARS

MARS was the first SHAR system where the classifier is built on the mobile device itself [13]. MARS processes the data stream generated from accelerometer sensor and provides personalization of learning models that are built based on individual annotated data for certain physical activities. Incremental Naive Bayes is used to update an anytime model in order to accommodate changes in the data stream.

5.2 Star

Star is an adaptive stream learning framework for SHAR [14]. It proposes an active learning technique well suited for choosing only a small amount of data to be labeled. Then the system is further refined with the selected true labeled data. Additionally, the framework uses incremental learning approach to handle concept drift over evolving streaming data in order to fit a particular user or context. Star performs activity recognition over sliding windows using modeling components. The experiments showed low computational cost and real time recognition.

5.3 A2EL

A novel online scheme, combining principles of Active and Adaptive Ensemble Learning (A2EL) for the context of SHAR is proposed in [1]. The system architecture uses a weighted Naïve Bayes classifier that can adapt to the incoming data of the stream without a need for an explicit concept drift detector. To tackle the multi-class nature of activity recognition problem, authors introduced a weighted combination for one-vs-one decomposition to reconstruct the original multi-class problem from two-class outputs. Moreover, the proposed ensemble is enhanced with an active learning module to reduce the labeling cost over real-time senor data streams.

6 Open Source Frameworks

Below we list the most two popular open source frameworks that provide tools for data stream analysis and learning:

1. Massive Online Analysis (MOA) [15]: Related to the WEKA project [16], MOA is implemented in Java. It provides a collection of ML algorithms (e.g., classification, clustering, outlier detection, and concept drift detection) and evaluation methods (e.g., periodic holdout, test-then-train, and prequential). MOA also provides a benchmark suite of artificial data generators for the MOA stream mining community which is growing actively. MOA can be found at https://github.com/Waikato/moa.
2. Scalable Advanced Massive Online Analysis (SAMOA) [17]: described as a framework as well as a library, it combines stream mining and distributed computing for big data streams. It features a pluggable architecture that allows it to run on several distributed stream processing engines. Like MOA, SAMOA is written in Java and supports the most common machine learning tasks such as classification and clustering. SAMOA can be found at http://www.samoa-project.net/.

7 CONCLUSION

In essence, SHAR applications need to perform online learning and classification on continuous data streams. The selection of learning approaches highly impacts the performance of SHAR applications regarding key cost criteria such as: energy, memory and time efficiency. Online learning in the context of SHAR must be performed to achieve multiple objectives which include system level and application level performance enhancements. The system level performance objectives include battery life enhancements in mobile devices. The application level performance objectives include enhancements of internal structures of learning models and their processing behavior in terms of concept drift detection, tackling class imbalance from uncertain data streams, model personalization and optimization. The availability of open source frameworks dedicated to big data stream online analysis should help further developments in these challenging tasks.

References

[1] Krawczyk, B., 2017. Active and adaptive ensemble learning for online activity recognition from data streams. Knowledge-Based Systems, 138, pp.69-78.

[2] Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J. and Woźniak, M., 2017. Ensemble learning for data stream analysis: A survey. Information Fusion, 37, pp.132-156.

[3] S. Wang, L. L. Minku, and X. Yao, “A systematic study of online class imbalance learning with concept drift,” arXiv preprint arXiv:1703.06683, 2017.

[4] Stefanowski J., Brzezinski D. (2017) Stream Classification. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA

[5] ur Rehman, M.H., Liew, C.S., Wah, T.Y. and Khan, M.K., 2017. Towards next-generation heterogeneous mobile data stream mining applications: Opportunities, challenges, and future research directions. Journal of Network and Computer Applications, 79, pp.1-24.

[6] L. L. Minku, “Online ensemble learning in the presence of concept drift,” Ph.D. dissertation, School of Computer Science, The University of Birmingham, 2010.

[7] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In Knowledge Discovery and Data Mining, pages 71–80, 2000.

[8] Albert Bifet, Geoff Holmes and Richard Kirkby, DATA STREAM MINING: A Practical Approach, MAY 2011. http://jwijffels.github.io/RMOA/MOA_2014_04/doc/pdf/StreamMining.pdf

[9] [Online]:https://www.cms.waikato.ac.nz/~abifet/book/chapter_6.htm. Last accessed on June,10 2018.

[10] Hulten, G., Spencer, L., Domingos, P.: Mining time changing data streams. In: KDD, pp. 97–106. ACM (2001)

[11] T.R.Hoens, R. Polikar, N.V. Chawla, “Learning from streaming data with concept drift and imbalance: An overview”, Prog.Artif.Intell., vol. 1, pp. 89-101, Apr. 2012.

[12] Bifet, A., Zhang, J., Fan, W., He, C., Zhang, J., Qian, J., Holmes, G. and Pfahringer, B., 2017, August. Extremely fast decision tree mining for evolving data streams. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1733-1742). ACM.

[13] Gomes, J.B., Krishnaswamy, S., Gaber, M.M., Sousa, P.A., Menasalvas, E., 2012b. Mars: a personalised mobile activity recognition system. In: Mobile Data Management (MDM), 2012 IEEE Proceedings of the 13th International Conference on, July 23- 26, Balngluru, India. IEEE, 2012, pp. 316–319.

[14] Abdallah, Z.S., Gaber, M.M., Srinivasan, B., & Krishnaswamy, S. (2015). Adaptive mobile activity recognition system with evolving data streams. Neurocomputing, 150, 304-317.

[15] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. 2010b. MOA: Massive Online Analysis. The Journal of Machine Learning Research 11 (2010), 1601–1604.

[16] Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

[17] Gianmarco De Francisci Morales and Albert Bifet. 2015. SAMOA: Scalable Advanced Massive Online Analysis. Journal of Machine

Learning Research 16 (2015), 149–153.