Journal of Physics: Conference Series

PAPER • OPEN ACCESS

The Waikato Open Source Frameworks (WEKA and MOA) for Machine Learning Techniques

To cite this article: Mahmood Shakir Hammoodi *et al* 2021 *J. Phys.: Conf. Ser.* **1804** 012133

IOP Publishing
1804 (2021) 012133 doi:10.1088/1742-6596/1804/1/012133

The Waikato Open Source Frameworks (WEKA and MOA) for Machine Learning Techniques

Mahmood Shakir Hammoodi, Hasanain Ali Al Essa, Wial Abbas Hanon

University of Babylon, Computer Center.

pre.mahmood.shakier@uobabylon.edu.iq
hasanain@uobabylon.edu.iq
wial@uobabylon.edu.iq

Abstract. WEKA and MOA are a free open-source software project specific for data mining and data stream mining, respectively. They are written in Java and developed at the University of Waikato, New Zealand. This research paper presents a comprehensive study of both consists of algorithms, evaluation, visualization, correlation between WEKA and MOA, workflow of implementation, and the classification accuracy.

Keywords. WEKA, MOA, Machine Learning Techniques.

1. Introduction

WEKA and MOA are the state-of-the-art facilities for developing machine learning techniques and their application to real-world data mining problems. WEKA and MOA are an open source developed by the University of Waikato in New Zealand that implement a collection of learning algorithms for data mining tasks and data stream mining tasks using JAVA language. Where, WEKA stands for Waikato Environment for Knowledge Analysis, and MOA stands for Massive Online Analysis. The classifiers available in WEKA can be used in MOA such as *moa.classifiers.meta.WEKA.Classifier*.

Whereas the data streams and the classifiers available in MOA can also be used in WEKA such as *weka.data.generators.classifiers.classification.MOA*. However, the main difference between them is the evaluation method applied. A rotation estimation method is used for the purpose of evaluation in WEKA which is also known as Cross Validation [8]. The main aim of this method is to partition a dataset into batches (i.e., train and test). Whereas in MOA, the test task is applied firstly and then training a data instance. Where a limited time is required. Hence, huge data can be manipulated and prediction can be achieved quickly at any time. This is also known as Prequential [1].

WEKA and MOA use the data in ARFF file format as a dataset comprises of attribute names and types with their data instances [3]. A data instance consists of attributes values with/without a class label which is the discrete attribute whose value needs to be predicted based on the values of other attributes.

The rest of the paper is organized as follows. In Section II, the main characteristics of WEKA is introduced. In Section III, the main characteristics of MOA is also introduced. Workflow of WEKA and MOA is discussed in Section IV. Whereas Section V presents the evaluation of the performance of both WEKA and MOA with respect to the classification accuracy achieved.

2. WEKA

In WEKA, the algorithms are applied for data pre-processing, classification, clustering, regression, association rules, and visualization. The classification and clustering algorithms can be applied with batch and incremental learning such as NaiveBayes and Hoeffding Tree. Where batch learning means that an algorithm uses part of training data instances. Multi scan over one or more data instances are required for making predictions. There are at least three types of batch learning which are batch, stochastic, and mini-batch. In batch, one batch is generated using all training data instances. In stochastic, the batch represents the size of one data instance. While in mini-batch, the batch size is more than one sample and less than the size of the training dataset.

In WEKA, Java classes (i.e., Attributes, Instance, Instances) are embedded in package. An attribute is represented as an object of class attribute consists of its name and type. An Instance is represented as an object of class Instance consists of the attribute values [9].

A learning algorithm is evaluated using Holdout strategy. A single holdout set is used to measure performance when batch learning reaches a threshold. The division between train and test sets needs to be identified in advance. Hence, results from different experiments can be evaluated and compared [2]. In addition, Cross-Validation is used as an evaluation technique (i.e., applying percentage splits repeatedly). A dataset is divided into n Folds (i.e., 10 Folds as default). Where each Fold is hold in turn for the purpose of training and train on n1. Number of evaluations will be n which is averaged (i.e., accuracy).

There is a collection of filters to process the data instances and attributes such as (Add Filter, Delete Filter, Make Indicator Filter, Merge Attribute Values Filter, Nominal To Binary Filter, Select Filter, Replace Missing Values Filter, Swap Attribute Values Filter, Discretise Filter, and Numeric Transform Filter). Feature selection technique is also included to select relevant features such as Wrapper and Relief. Figures 1 and 2 show the WEKA's framework and explorer.

Figure 1. The WEKA's Framework.

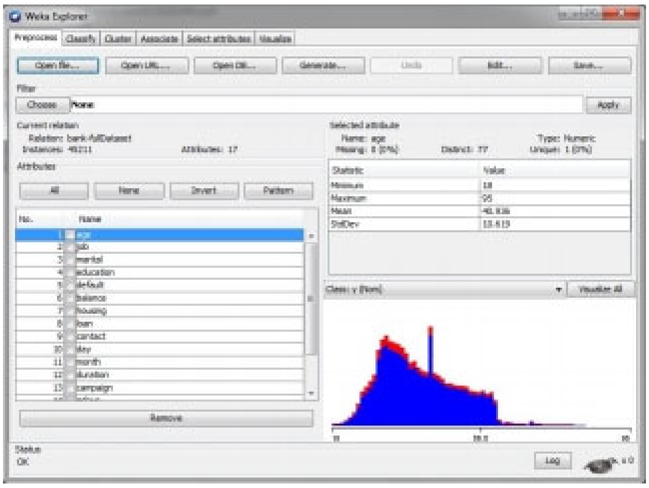

Figure 2. The WEKA's Explorer.

3. MOA

In MOA, the algorithms are applied for classification, regression, clustering, outlier detection, concept drift detection, recommender systems, and visualization. MOA is a software environment and an open source framework which is also written in Java for implementing algorithms and running experiments for online learning from evolving data streams. The stream arrives instance by instance. Once an instance from a data stream has been processed, it cannot be retrieved as it is ignored [4]. Figure 3 shows the MOA's framework.

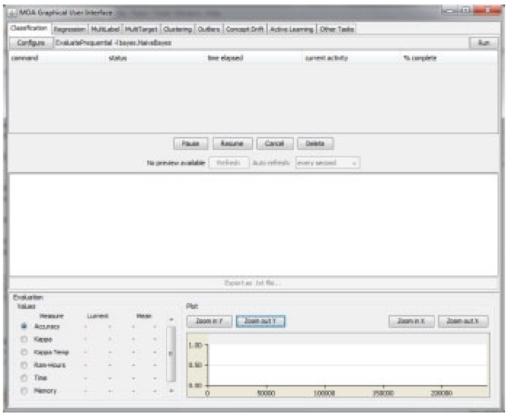

Figure 3. The MOA's Framework

A learning algorithm is evaluated using test and train (Prequential). In the current time (i.e., time stamp) of a data instance, test a model is applied before using the data instances for training the model. Where a set of time stamps refer to a time window. If a dataset consists 1000 data instances with window size of 100. This means that there are 10 windows. 100 data instances with each window. Only recent data is examined instead of the whole data stream using windowing approaches [7] which consist of three models which are sliding, damped, and landmark [11], [10]. Sliding window is the most common model which is frequently applied in data with streaming setting [10]. It is also known as first-in-first-out.

The performance of two algorithms can be evaluated in MOA using visualization method for clustering and outliers, as shown in Figures 4, 5, 6, and 7. Another evaluation is moa.classifiers.drift.SingleClassifierDrift which is used to evaluate drift detection methods (i.e., unexpected in data over time).

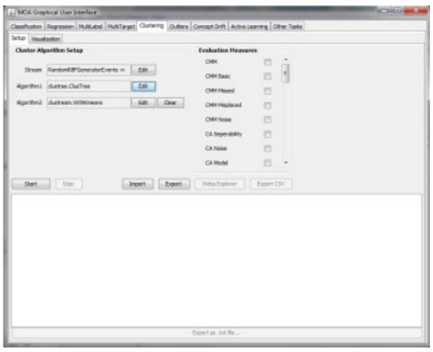

Figure 4. Setup of Clutering Algorithms Visualization in MOA.

Figure 5. Evaluation of the Performance of the Clustering Algorithms in MOA

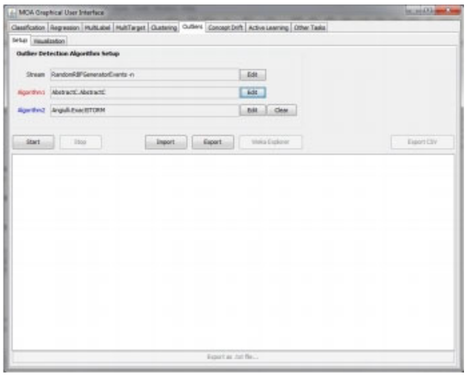

Figure 6. Setup of Outliers Detection Visualization in MOA.

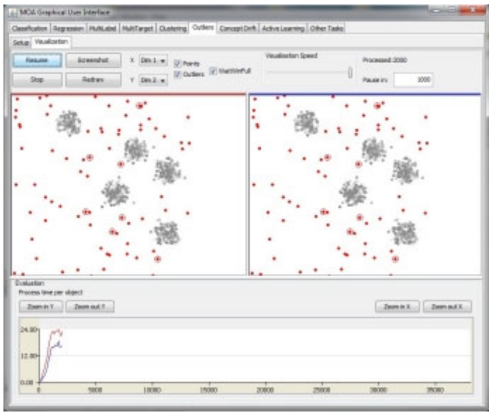

Figure 7. Evaluation of the Performance of the Outliers Detection Algorithms in MOA.

4. WORKFLOW OF WEKA AND MOA

This section presents the workflow of WEKA and MOA in terms of evaluation technique used. Figure 8 shows the typical workflow of WEKA. A dataset needs to be partitioned in dvance (i.e., offline) for training and test. A classifier is then introduced to build a model using the training data. Whereas test data is used to evaluate the generated model. This is very common technique used in WEKA to evaluate the models. However, huge data (i.e., big data) cannot be processed using this technique as it is considered time consuming [1].

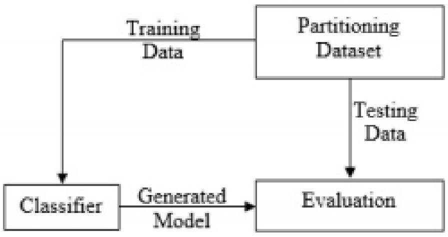

Figure 8. Worflow of WEKA.

Thus, analysing data streams in real-time needs an adaptive and computationally efficient algorithm which does not need to process the entire data as shown in Figure 9. Where a single scan over data stream is required. However a dataset which does not generated in a streaming setting mentioned in Section III can be used in MOA as well. Whereas WEKA unable to process the dataset with the streaming setting as a large amount of memory space is required.

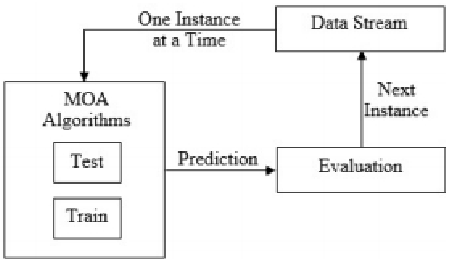

Figure 9. Worflow of MOA.

Where Test and Train refer to Prequential. Examining data (i.e., data instance) is applied over the windowing approaches in real-time (i.e., online). Each examined data instance (i.e., test task) is then sent into a classifier for the purpose of training. Hence data with any size (i.e., even if size is unlimited) can be used as Prequential is considered as memory less [5]. However, once a data instance is arrived it cannot be retrieved again as it is discarded. Therefore, MOA algorithms are designed to handle data streams in terms of accumulating statistics over time. This is known as statistical windowing approaches. For example, a counter can be kept for counting number of true positive prediction over time stamp, for each data instance with its class label. Where the counter is incremented by 1 when predicted class label equals to actual class label. Accuracy is then calculated by dividing number WEKA Generator MOA (Test-Then-Train) WEKA (Cross-Validation) Agrawal 95.10 94.50 BayesNet 72.90 69.90 RandomRBF 89.10 88.95 RDG1 94.50 92.14 LED24 74.60 73.91 of true positive predictions over total attempts at the end of stream [6].

5. EVALUATION

This section presents an evaluation of the performance of WEKA and MOA in terms of the accuracy achieved using 5 datasets generated by WEKA data generators which are available in (weka.datagenerators.classifiers.classification) such as Agrawal, BayesNet, LED24, RandomRBF, and RDG1. Each generated dataset with 100000 data instances. A default setting of each data generator is applied. Table I shows description of the generated datasets. In MOA, window size of 10000 instances is used. Whereas Cross-Validation of 10 Folds is used in WEKA.

Table 1. DESCRIPTION OF DATASETS

WEKA Generator	Features	Class Label
Agrawal	9	2
BayesNet	9	2
RandomRBF	10	2
RDG1	9	2

LED24	24	10

For the purpose of evaluation, 2 incremental classifiers are chosen which are NaiveBayes and Hoeffding Tree. They can be applied in WEKA and MOA. Table II and Table III show the accuracy reported. Whereas Figure 10 and Figure 11 show the difference in percentage of the accuracy. Although the datasets are generated using WEKA generators, MOA achieved a higher accuracy compared with WEKA with respect to the classifiers and datasets. The accuracy would be affected due to partitioning a dataset into parts or Folds for training and testing. This also is considered not applicable technique for big data. Where size of data is not known as it is generated and labeled dynamically in real-time. Therefore, each data instance is used for both tasks test and train overtime. Accuracy is then calculated incrementally, and can be increased gradually over time.

Table 3. THE ACCURACY ACHIEVED WITH NAIVEBAYES CLASSIFIER

WEKA Generator	MOA (Test-Then-Train)	WEKA (Cross-Validation)
Agrawal	88.40	88.30
BayesNet	66.90	65.86
RandomRBF	69.10	68.88
RDG1	87.30	85.22
LED24	74.80	74.01

Table 3. THE ACCURACY ACHIEVED WITH HOEFFDING TREE CLASSIFIER

WEKA Generator	MOA (Test-Then-Train)	WEKA (Cross-Validation)
Agrawal	95.10	94.50
BayesNet	72.90	69.90
RandomRBF	89.10	88.95
RDG1	94.50	92.14
LED24	74.60	73.91

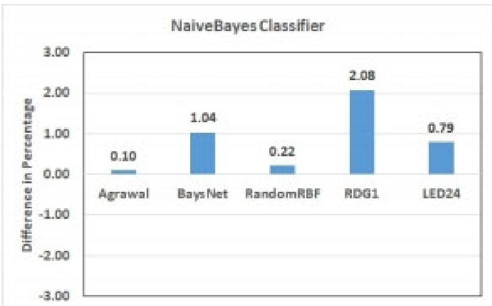

Figure 10. Difference in Percentage with NaiveBayes Classifier

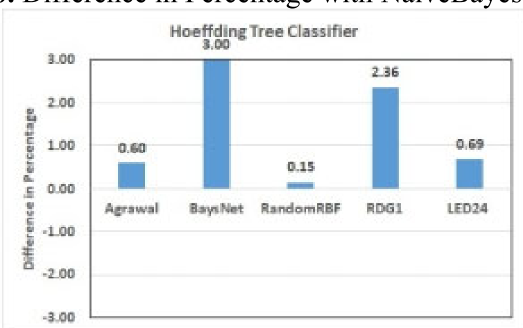

Figure 11. Difference in Percentage with Hoeffding Tree Classifier

6. CONCLUSION

WEKA can only handle datasets with limited size. Whereas MOA is designed to deal with big data, which is generated and labeled dynamically in real-time with ultra high speed. Where a limited amount of memory is used in a limited amount of time. A class label can be predicted at any time as well as one data instance is processed at a time. Based of these environments and requirements, a robust evaluation method is used with MOA to evaluate the performance of the algorithms as discussed in this research paper. The results show that the Test-Then-Train can achieved higher accuracy compared with Cross-Validation.

References

[1] Albert Bifet, Gianmarco de Francisci Morales, Jesse Read, Geoff Holmes, and Bernhard Pfahringer. Efficient online evaluation of big data stream classifiers. In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pages 59–68. ACM, 2015.

[2] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. Journal of Machine Learning Research, 11(May):1601–1604, 2010.

[3] Mohd Fauzi bin Othman and Thomas Moh Shan Yau. Comparison of different classification techniques using weka for breast cancer. In 3rd Kuala Lumpur International Conference on Biomedical Engineering 2006, pages 520–523. Springer, 2007.

[4] Alberto Cano. A survey on graphic processing unit computing for large-scale data mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(1):e1232, 2018.

[5] [5] Joa˜o Gama, Raquel Sebastia˜o, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 329–338. ACM, 2009.

[6] Mahmood Shakir Hammoodi, Frederic Stahl, and Atta Badii. Real-time feature selection technique with concept drift detection using adaptive micro-clusters for data stream mining. Knowledge-Based Systems, 161:205–239, 2018.

[7] Imen Khamassi, Moamar Sayed-Mouchaweh, Moez Hammami, and Khaled Gh´edira. Discussion and review on evolving data streams and concept drift adapting. Evolving systems, 9(1):1–23, 2018.

[8] Marianthi Markatou, Georgios Afendras, and Claudio Agostinelli. Weighted cross validation in model selection. Wiley Interdisciplinary Reviews: Computational Statistics, 10(6):e1439, 2018.

[9] Ian H Witten, Eibe Frank, Leonard E Trigg, Mark A Hall, Geoffrey Holmes, and Sally Jo Cunningham. Weka: Practical machine learning tools and techniques with java implementations. 1999.

[10] Jonghem Youn, Junho Shim, and Sang-Goo Lee. Efficient data stream clustering with sliding windows based on locality-sensitive hashing. IEEE Access, 6:63757–63776, 2018.

[11] Unil Yun, Donggyu Kim, Eunchul Yoon, and Hamido Fujita. Damped window based high average utility pattern mining over data streams. Knowledge-Based Systems, 144:188–205, 2018.