ARFF Data Source Library for Distributed Single/Multiple Instance, Single/Multiple Output Learning on Apache Spark

Jorge Gonzalez-Lopez[1], Sebastian Ventura[2], and Alberto Cano[1(✉)]

[1] Department of Computer Science, Virginia Commonwealth University, Richmond, USA
{gonzalezlopej,acano}@vcu.edu
[2] Department of Computer Science and Numerical Analysis, University of Cordoba, Córdoba, Spain
sventura@uco.es

Abstract. Apache Spark has become a popular framework for distributed machine learning and data mining. However, it lacks support for operating with Attribute-Relation File Format (ARFF) files in a native, convenient, transparent, efficient, and distributed way. Moreover, Spark does not support advanced learning paradigms represented in the ARFF definition including learning from data comprising single/multiple instances and/or single/multiple outputs. This paper presents an ARFF data source library to provide native support for ARFF files, single/multiple instance, and/or single/multiple output learning on Apache Spark. This data source extends seamlessly the Apache Spark machine learning library allowing to load all the ARFF file varieties, attribute types, and learning paradigms. The ARFF data source allows researchers to incorporate a large number of diverse datasets, and develop scalable solutions for learning problems with increased complexity. The data source is implemented on Scala, just like the Apache Spark source code, however, it can be used from Java, Scala, and Python. The ARFF data source is free and open source, available on GitHub under the Apache License 2.0.

Keywords: ARFF · Apache Spark · Multi-instance · Multi-output

1 Introduction

The exponential growth of data, both in size and complexity, has resulted in a pressing need to develop scalable solutions to learn models from large-scale data. As ever-increasing data sizes have outpaced the capabilities of single machines, distributing computations to multiple nodes has gained more relevance [5]. Several frameworks have been developed based on the MapReduce programming model [7]. This model offers a simple and robust paradigm to handle large-scale datasets in a cluster. One of the first implementations of the MapReduce model

J. M. F. Rodrigues et al. (Eds.): ICCS 2019, LNCS 11538, pp. 173–179, 2019.
https://doi.org/10.1007/978-3-030-22744-9_13

was Hadoop [18], yet one of its critical disadvantages is that it processes the data from the distributed file system, which introduces a high latency. On the contrary, Spark [19] provides in-memory computation which results in a big performance improvement, especially on iterative jobs. Using in-memory data has been proved to be of the utmost importance for speeding up machine learning algorithms [2,8].

The WEKA [10] machine learning suite unifies access to well-known and state-of-the-art techniques for traditional classification, regression, clustering, and feature selection. This tool has been extended to accommodate numerous types of advanced learning paradigms including MULAN [17] and MEKA [14] for multi-output learning, MILK [10] for multi-instance learning, MOA [3] for data stream mining, JCLEC [6] for all previous, among others. These frameworks are built on top of WEKA or provide wrappers to its methods, and despite considering different learning paradigms all of them are based on the Attribute-Relation File Format (ARFF). Most of the benchmark datasets used in supervised learning are provided in this relational format, forcing the researchers to use one of the mentioned frameworks or to implement parsers to load the data in their codes. There are wrapper approaches to facilitate the use of Weka methods on Spark through the Weka interface [12] but the Spark machine learning library still lacks native support for ARFF files and the advanced learning paradigms.

This paper presents a native data source library to support ARFF files, single/multiple instance, and single/multiple output learning on Apache Spark[1]. The functionality of the data source extends the ones included in Apache Spark machine learning library but incorporating all the advanced learning paradigms considered in the definition of the ARFF. Therefore, researchers in these areas are provided with a powerful tool to facilitate the implementation and usage of their methods on Apache Spark, taking advantage of the distributed performance capabilities. This is expected to attract even more attention and users to the Apache Spark framework and help in its future development.

The rest of the manuscript is organized as follows: Sect. 2 introduces the learning paradigms supported by the data source, Sect. 3 presents the library framework, Sect. 4 shows some illustrative examples, and finally Sect. 6 presents some conclusions and future work.

2 Learning Paradigms

Learning paradigms can be categorized according to the task goal into supervised learning (classification for predicting a discrete label, regression for predicting a continuous output) and unsupervised learning (clustering, pattern mining, among others). However, these paradigms can be also broken down according to the induction process from a single/multiple instances and according to the prediction of a single/multiple outputs. This way, new paradigms such as multi-instance learning, multi-label classification, multi-target regression, multi-instance multi-label learning have become popular in recent years.

[1] https://github.com/jorgeglezlopez/spark-arff-data-source.

Let $\mathcal{X}$ denote the domain of instances in a dataset, a single instance $\boldsymbol{x} \in \mathcal{X}$ is represented as a set of features $\boldsymbol{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, x_d\}$, where $\boldsymbol{d}$ is the number of features. Traditional supervised learning, single-instance single-output, finds a mapping function that associates a single instance $\boldsymbol{x}_i$ to a single output $\boldsymbol{y}_i$, where classification considers $y_i \in \mathbb{Z}$ and regression $y_i \in \mathbb{R}$.

Multi-instance learning associates a set of instances $\boldsymbol{b}_i = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, x_{|b_i|}\}$, known as a bag, to a single output $\boldsymbol{y}_i$. Multi-instance classification [1] considers $y_i \in \mathbb{Z}$, while multi-instance regression [11] defines $y_i \in \mathbb{R}$.

Multi-output learning associates a single instance $\boldsymbol{x}_i$ to a finite set of outputs $\boldsymbol{y}_i = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_t\}$. In multi-output classification [9,16], also known as multi-label classification, the output is represented as a binary vector $\boldsymbol{y}_j \in \{0, 1\}^t$ where each value can be 1 if the label is present and 0 otherwise. In multi-output regression [4,13,15], also known as multi-target regression, each output is a vector of real values $\boldsymbol{y}_j \in \mathbb{R}^t$.

Multi-instance multi-output learning combines both paradigms to represent an example as a bag (set of instances) associated with multiple outputs. Most of the learning paradigms presented are recent problems, therefore the availability of algorithms and datasets is fairly restricted. Our data source aims to bring these new paradigms to the distributed environment on Apache Spark.

3 Library Framework

The data source has been implemented in Scala, just like the original Apache Spark source code. This source implements the *FileFormat* trait in the original *datasources* package. This trait allows Spark to read data from external storage systems, like HDFS or a local file system, through *DataFrameReader* interface. Figure 1 shows a class diagram with the structure of the data source.

- *ARFFFileFormat*: It represents the entry point of the data source from the *DataFrameReader* interface. This class inherits from a series of interfaces in order to ensure the correct communication from the *DataFrameReader*. The first interface is the *DataSourceRegister* which can register the data source under an alias. The second set of interfaces is the *TextBasedFileFormat* and *FileFormat*, which define the methods that will be called from the *DataFrameReader* in order to create the proper DataFrame. The creation of the DataFrame is split into creating a suitable *schema* for the attributes and parsing all the instances. Both processes are isolated to each other because of the *FileFormat* interface.
- *ARFFInferSchema*: This class receives the header and the options defined by the user. The header comes either from the beginning of the file or from an independent file. The class uses the *ARFFAttributeParser* class to extract the information of each attribute using regular expressions. This information is used to create the required *schema* that matches the learning paradigm, as well as to store the information of the attributes in the *metadata*. Each column uses the *ARFFAttributeParser* and the *ExtendedAttributeGroup* to

transform the corresponding information of the header into *metadata*. The *ExtendedAttributeGroup* adds support for new types of attributes, such as *String* and *Date*.

– *ARFFInstanceParser*: It parses each of the lines of data in the file into *Rows* for the DataFrame. It reconstructs the *ARFFAttributeParser* of each attribute from the *metadata* received in the *schema*. Once all the parsers have been constructed, it reads each line of data allowing to read both *dense* and *sparse* instances. In every instance, the original values are transformed into a *numeric* format and stored in the corresponding fields of a *Row* following the same order they present in the *header*.
– *ARFFOptions*: This class handles the options set by the user, which can only be set from there at the beginning and will be final during the execution of the data source. The supported options are explained in Sect. 4.

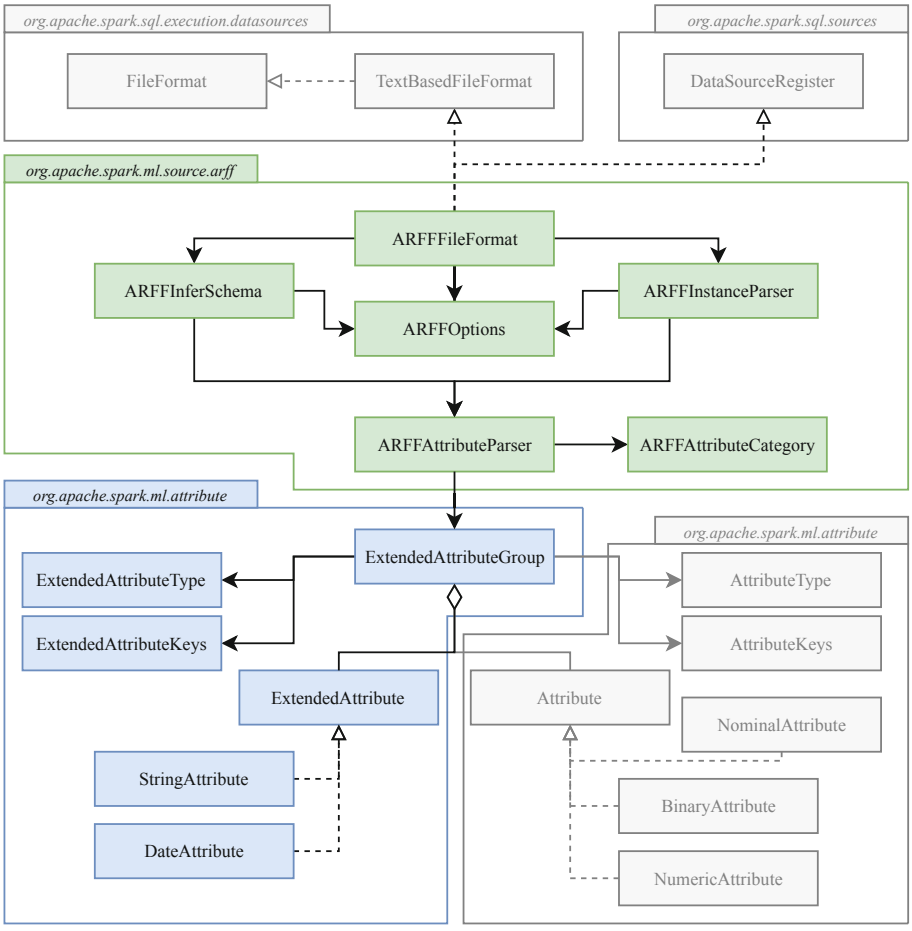

Fig. 1. ARFF data source class diagram for Apache Spark.

This data source has a series of advantages over the *libSVM* data source, which is the data source used by the machine learning library. The main functionalities and advantages are:

- Support for different types of learning paradigms, each with a different *schema*.
- Automatic conversion of all the features to numeric types. It transforms types such as *Date*, *String*, or *Nominal* to a Double. This allows using the Dataframe directly with the machine learning methods.
- Storage of the information of each attribute in the metadata of the *schema*. This information can be used in different algorithms such as finding the best splits in decision trees over *nominal* data.
- Dynamic and automatic conversion to either *dense* or *sparse* instances, whichever uses less storage space.

4 Illustrative Examples

The process to create a DataFrame out of an ARFF file is the same as for the built-in sources. Algorithm 1.1 illustrates the simplest form to call the data source. It also supports to manually specify extra options (through *ARFFOptions*) to define the learning paradigm, formed by a key and a value. The available options are:

- *("schemaFile", "path")*: Specifies a file with the definition of the ARFF header. By default, it uses the header at the top for the files being loaded.
- *("xmlMultilabelFile", "path")*: Specifies the XML file that defines the attributes that are considered outputs in a multi-label paradigm. By default, it is empty.
- *("numOutputs", number)*: Specifies the number of attributes at the end of the header that are considered outputs. However, if "xmlMultilabelFile" is defined there is no need to specify a number of outputs. By default, only the last attribute is an output.
- *("multiInstance, boolean)*: Indicates if the file defines a multi-instance paradigm. By default it is *false*.

```
val dataFrame = sparkSession
  .read
  .format(''org.apache.spark.ml.source.arff'')
  .option(key, value) // Add the options required
  .load(path)
```

Algorithm 1.1. DataFrame creation from an ARFF file

5 Current Software Version

Table 1 presents the information about the current version of the software. This table indicates the address where the source code can be found, as well as the documentation where the details of the implementation can be reviewed.

Table 1. Software metadata

Nr.	Software metadata description	Information
S1	Current software version	1.1
S2	Permanent link to executables of this version	https://github.com/jorgeglezlopez/spark-arff-data-source/
S3	Legal Software License	Apache License 2.0
S4	Computing platform	Apache Spark
S5	Installation requirements & dependencies	Apache Spark 2.0 or newer, and Scala 2.10 or 2.11
S6	User manual	https://github.com/jorgeglezlopez/spark-arff-data-source/blob/master/README.md
S7	Support email for questions	`gonzalezlopej@vcu.edu` `acano@vcu.edu`

6 Conclusions

This paper presented a new data source library for Apache Spark, which extends seamlessly the built-in data sources to support the ARFF file format and advanced learning paradigms including single/multiple instances and single/-multiple outputs learning. This new data source aims to significantly increase the number of machine learning datasets and algorithms ready-to-use for the Spark community. It provides the tools to help researchers to develop new algorithms in a distributed environment to address the increasing complexity.

This software has defined the schema structure for different paradigms, besides incorporating the data format. As future work, it is expected that new frameworks with non-traditional learning paradigms will extend this work by building novel distributed learning algorithms.

Acknowledgments. This research was partially supported by the 2018 VCU Presidential Research Quest Fund and an Amazon AWS Machine Learning Research award.

References

1. Amores, J.: Multiple instance classification: review, taxonomy and comparative study. Artif. Intell. **201**, 81–105 (2013)
2. Bei, Z., Yu, Z., Luo, N., Jiang, C., Xu, C., Feng, S.: Configuring in-memory cluster computing using random forest. Future Gener. Comput. Syst. **79**, 1–15 (2018)
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
4. Borchani, H., Varando, G., Bielza, C., Larrañaga, P.: A survey on multi-output regression. Wiley Interdisc. Rev.: Data Min. Knowl. Discov. **5**(5), 216–233 (2015)
5. Cano, A.: A survey on graphic processing unit computing for large-scale data mining. Wiley Interdisc. Rev.: Data Min. Knowl. Discov. **8**(1), e1232 (2018)
6. Cano, A., Luna, J.M., Zafra, A., Ventura, S.: A classification module for genetic programming algorithms in JCLEC. J. Mach. Learn. Res. **16**, 491–494 (2015)
7. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
8. Gonzalez-Lopez, J., Cano, A., Ventura, S.: Large-scale multi-label ensemble learning on Spark. In: IEEE Trustcom/BigDataSE/ICESS, pp. 893–900 (2017)
9. Gonzalez-Lopez, J., Ventura, S., Cano, A.: Distributed nearest neighbor classification for large-scale multi-label data on Spark. Future Gener. Comput. Syst. **87**, 66–82 (2018)
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. ACM SIGKDD **11**(1), 10–18 (2009)
11. Herrera, F., et al.: Multi-instance regression. In: Multiple Instance Learning, pp. 127–140. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47759-6_6
12. Koliopoulos, A.K., Yiapanis, P., Tekiner, F., Nenadic, G., Keane, J.: A parallel distributed weka framework for big data mining using Spark. In: IEEE International Congress on Big Data, pp. 9–16 (2015)
13. Melki, G., Cano, A., Kecman, V., Ventura, S.: Multi-target support vector regression via correlation regressor chains. Inf. Sci. **415–416**, 53–69 (2017)
14. Read, J., Reutemann, P., Pfahringer, B., Holmes, G.: MEKA: a multi-label/multi-target extension to WEKA. J. Mach. Learn. Res. **17**(21), 1–5 (2016)
15. Reyes, O., Cano, A., Fardoun, H., Ventura, S.: A locally weighted learning method based on a data gravitation model for multi-target regression. Int. J. Comput. Intell. Syst. **11**(1), 282–295 (2018)
16. Tsoumakas, G., Katakis, I.: Multi-label classification: an overview. Int. J. Data Warehous. Min. **3**(3), 1–13 (2007)
17. Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., Vlahavas, I.: MULAN: a Java library for multi-label learning. J. Mach. Learn. Res. **12**, 2411–2414 (2011)
18. White, T.: Hadoop: The Definitive Guide. O'Reilly Media, Inc., Sebastopol (2012)
19. Zaharia, M., et al.: Apache Spark: a unified engine for big data processing. Commun. ACM **59**(11), 56–65 (2016)