Journal of Machine Learning Research 20 (2019) 1-5 Submitted 12/17; Revised 11/18; Published 2/19

spark-crowd: A Spark Package for Learning from Crowdsourced Big Data

Enrique G. Rodrigo ENRIQUE.GRODRIGO@UCLM.ES

Juan A. Aledo JUANANGEL.ALEDO@UCLM.ES

José A. Gámez JOSE.GAMEZ@UCLM.ES
University of Castilla-La Mancha
Escuela Superior de Ingeniería Informática de Albacete
Avenida de España s/n
Albacete (02071), Spain

Editor: Geoff Holmes

Abstract

As the data sets increase in size, the process of manually labeling data becomes unfeasible by small groups of experts. Thus, it is common to rely on crowdsourcing platforms which provide inexpensive, but noisy, labels. Although implementations of algorithms to tackle this problem exist, none of them focus on scalability, limiting the area of application to relatively small data sets. In this paper, we present `spark-crowd`, an Apache Spark package for learning from crowdsourced data with scalability in mind.

Keywords: crowdsourcing, crowdsourced data, learning from crowds, big data, multiple noisy labeling, spark

1. Introduction

With the recent appearance of crowdsourcing platforms such as Amazon Mechanical Turk, several researchers have expressed their interest in taking advantage of them to increase the efficiency and scope of their research (Snow et al. 2008). Thanks to these platforms, problems that would be too expensive to tackle using traditional methods have now become easier and, at the same time, tasks which were not feasible become tractable.

However, the use of crowdsourcing in the field of Machine Learning is not without problems. The most important of them is the considerable uncertainty associated with members of the group providing the annotations in the process of data acquisition. Some of them may be better qualified for the task than others; some may give useless information or even try to ruin the results. After this process, the practitioner receives a data set of tuples $(example, annotator, annotation)$. With these, the problem usually consists in estimating the ground truth for each instance using the annotations obtained. As the annotator's quality is unknown, it is also common to want an estimation of the quality of each annotator. Thus, to obtain good results with crowdsourced data, the machine learning process needs to take into account all these difficulties (Zhang et al. 2016).

These problems are exacerbated with large data sets as the cost constraints for the project compels to use as few annotations as possible so that the majority of examples can be labeled. For this reason, the practitioner usually receives a reduced number of annotations per instance, invalidating the use of simple aggregation methods and making the estimation of which annotator to trust crucial. Moreover, as the data increases, some other problems for data practitioners also appear, as processing the data sets in a limited amount of time becomes much more difficult (Ouyang et al. 2016). Recently, platforms such as Apache Hadoop and Apache Spark can tackle these problems, but they are not used in the available implementations of the algorithms capable of learning accurately from crowdsourced data.

In this paper, we present `spark-crowd`[1], an *Apache Spark* package for tackling the problem of learning from crowds in domains where scalability is essential. The following sections will give an overview of the project information and methods, compare it with other packages implementing similar techniques, and comment the future plans of the developers of this package.

2. Project Management

This project makes heavy use of `git` and `Github` to make the collaboration easier, as well as a means for issue tracking, code integration, and idea discussions. To assure the quality of the package, we use a continuous integration system (`CircleCI`) so that the contributions of any developer can be checked automatically to facilitate the integration of new code. Apart from this, we also ensure code quality through a set of unit tests (testing the functionality of all the methods), which provides a coverage of 99% of the package in its version 0.2.1.

The documentation for the project is provided using `sphinx` through Github Pages[2]. This documentation includes installation instructions as well as examples to start using the package. The users can also find a description of the methods as well as a comparison with similar packages. We also offer a Docker Image with the package preinstalled, so that testing the algorithms is as comfortable as possible in the platforms Docker is supported (Windows, Mac and Linux based systems, among others). Moreover, all the dependencies, as well as the package itself, are available under open source licenses so that they can be used in a wide variety of scenarios.

The package is also published in `Maven Central` and `Spark Packages` which makes adding the library as a dependency easier (both as a project dependency or directly in the Spark Shell). It is also published in the `mloss` site, to facilitate dissemination.

3. Implementation Description

This package implements 12 methods for dealing with crowdsourced data, as shown in Table 1. We provide algorithms for both discrete class (binary and multiclass) and continuous target variables. Except for the `MajorityVoting` methods (that includes the mode for discrete classes and the mean for continuous target variables), all of them return an estimation of the annotator's quality. Only the Raykar's algorithms (divided into the methods

1. The package is available in `https://github.com/enriquegrodrigo/spark-crowd`.
2. You can find the documentation of the package at https://enriquegrodrigo.github.io/spark-crowd/.

Method	Binary	Multiclass	Continuous	Reference
Majority Voting	X	X	X	Sheng (2011)
DawidSkene	X	X		Dawid and Skene (1979)
IBCC	X	X		Kim and Ghahramani (2012)
GLAD	X			Whitehill et al. (2009)
CGLAD	X			Rodrigo et al. (2018)
Raykar	X	X	X	Raykar et al. (2010)
CATD			X	Li et al. (2014a)
PM			X	Li et al. (2014b)
PMTI			X	Zheng et al. (2017)

Table 1: Methods implemented in `spark-crowd`

RaykarBinary, RaykarMulti, and RaykarCont) are able to take advantage of feature vectors while the others just use information about the annotations. For a more detailed description of the implemented algorithms, the reader may refer to the references in Table 1. In addition to the algorithms, the package also provides classes to manage annotation types as well as result objects.

An example of the library usage can be found in Listing 1. Although this code seems simple, it can be executed both locally or in a distributed environment (using the Apache Spark platform) without any modifications. The user may refer to the documentation in order to find more involved examples.

```
import com.enriquegrodrigo.spark.crowd.methods.DawidSkene
import com.enriquegrodrigo.spark.crowd.types._
//Loading file (any spark compatible format)
val data = spark.read.parquet("datafile.parquet").as[MulticlassAnnotation]
//Applying algorithm (data with columns [example, annotator, value])
val mode = DawidSkene(data.as[MulticlassAnnotation])
//Get MulticlassLabel with the class predictions
val pred = mode.getMu().as[MulticlassLabel]
//Annotator precision matrices
val annprec = mode.getAnnotatorPrecision()
```

Listing 1: Example of spark-crowd usage

4. Related Software

The most relevant software packages related to the one presented in this document are `Ceka` (Zhang et al. 2015) and the methods implemented in Zheng et al. 2017 (referred here as `Truth-Inf`). Both of them contain several algorithms for learning from crowdsourced data. The reader can find a full comparison study between these packages in the `spark-crowd` documentation, at the *Comparison with other packages* section. Here, however, we include a comparison for those methods considered in all three packages, i.e. MajorityVoting and DawidSkene[3]. We used the same environment for $\mathbf{t_1}$ (execution using only one core). For

3. The packages also implement GLAD and Raykar's algorithms. However, in `Ceka` these algorithms are implemented using wrappers to other libraries (the library for the GLAD algorithm is not available in our platform as it is given as an EXE Windows file, and the wrapper for Raykar's algorithms does not admit any configuration parameters).

	Majority Voting							DawidSkene						
	Ceka		Truth-Inf		spark-crowd			Ceka		Truth-Inf		spark-crowd		
Data set	**Acc**	$\mathbf{t_1}$	**Acc**	$\mathbf{t_1}$	**Acc**	$\mathbf{t_1}$	$\mathbf{t_c}$	**Acc**	$\mathbf{t_1}$	**Acc**	$\mathbf{t_1}$	**Acc**	$\mathbf{t_1}$	$\mathbf{t_c}$
binary1	0.931	21	0.931	1	0.931	11	7	0.994	57	0.994	12	0.994	31	32
binary2	0.936	15973	0.936	8	0.936	11	7	0.994	49259	0.994	161	0.994	60	51
binary3	X	X	0.936	112	0.936	21	8	X	X	0.994	1705	0.994	111	69
binary4	X	X	0.936	2908	0.936	57	37	X	X	M	M	0.904	703	426

Table 2: Results for accuracy and execution time on the test data sets

$\mathbf{t_c}$ we used an Apache Spark cluster with 3 executor nodes of 10 cores and 30Gb of memory each. The same code was executed on both platforms. This is, actually, one of the advantages of the library presented here. In Table 2, we show the accuracy and execution time (in seconds) obtained for four data sets of increasing size by the three libraries (the documentation of the package contains the details for these data sets). Regarding accuracy, all the packages achieved comparable results, as should be expected. Regarding the execution time, the results obtained by our package are significantly better, especially in the last two data sets. Against `Ceka`, our implementation obtained a slight speedup even in the smallest data set for both algorithms. For the second data set, `spark-crowd` obtained speedups of several orders of magnitude. It was not possible to obtain results for the last two instances in a reasonable amount of time with `Ceka`. However, `spark-crowd` solved these cases in few minutes. `Truth-Inf` obtained better results in terms of execution time in comparison to Ceka. As the algorithms chosen in this comparison are quite simple in terms of time complexity, the benefits of parallelization are less apparent, especially in the first two instances for the MajorityVoting algorithm, where the implementation in `Truth-Inf` obtained better results in terms of execution time. However, as the data and the complexity of the algorithm increases, the cost of parallelization is less noticeable. In the MajorityVoting algorithm one can see good results in the last two data sets. For the DawidSkene algorithm, the benefits of parallelization can be seen from the second data set, with a significant speedup especially in the third. `Truth-Inf` was not able to complete the execution for the last data set, due to a memory error.

5. Future Plans and Conclusions

As this is a field of recent interest, new algorithms may appear. Thus, it is in our plans to add the most interesting algorithms to the package. We also expect to develop new features and algorithms that scale even better for this problem using this library as the base of further developments. Finally, we hope that the community contributes with new implementations, which we will be delighted to integrate into the library.

Acknowledgments

This work has been partially funded by FEDER funds, the Spanish Research Agency (MINECO) and the Junta de Comunidades de Castilla-La Mancha (JCCM) through projects TIN2013-46638-C3-3-P, TIN2016-77902-C3-1-P and SBPLY/17/18050/000493. Enrique G. Rodrigo has also been funded by the FPU scholarship FPU15/02281 by MECD.

References

Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics*, pages 20–28, 1979.

Hyun-Chul Kim and Zoubin Ghahramani. Bayesian classifier combination. In *AISTATS*, volume 22, pages 619–627, La Palma, Canary Islands, 2012.

Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han. A confidence-aware approach for truth discovery on long-tail data. *Proceedings of the VLDB Endowment*, 8(4):425–436, 2014a.

Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *ACM SIGMOD*, pages 1187–1198, New York, NY, USA, 2014b.

Robin Wentao Ouyang, Mani Srivastava, Alice Toniolo, and Timothy J Norman. Truth discovery in crowdsourced detection of spatial events. *IEEE Transactions on Knowledge and Data Engineering*, 28(4):1047–1060, 2016.

Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322, 2010.

Enrique G. Rodrigo, Juan A. Aledo, and Jose A. Gamez. CGLAD: Using GLAD in crowdsourced large datasets. *In Lecture Notes in Computer Science, vol 11314 (IDEAL 2018)*, pages 783–791, 2018.

Victor S Sheng. Simple multiple noisy label utilization strategies. In *IEEE ICDM*, pages 635–644, 2011.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNLP*, pages 254–263, 2008.

Jacob Whitehill, Ting fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, pages 2035–2043. 2009.

Jing Zhang, Victor S Sheng, Bryce A Nicholson, and Xindong Wu. Ceka: a tool for mining the wisdom of crowds. *Journal of Machine Learning Research*, 16(1):2853–2858, 2015.

Jing Zhang, Xindong Wu, and Victor S Sheng. Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576, 2016.

Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: is the problem solved? *Proceedings of the VLDB Endowment*, 10(5): 541–552, 2017.