Journal of Machine Learning Research 17 (2016) 1-5 Submitted 7/15; Revised 5/16; Published 5/16

JCLAL: A Java Framework for Active Learning

Oscar Reyes OGREYESP@GMAIL.COM
Eduardo Pérez EPEREZP@FACINF.UHO.EDU.CU
Department of Computer Science
University of Holguín
Holguín, Cuba

María del Carmen Rodríguez-Hernández 692383@UNIZAR.ES
Department of Computer Science and Systems Engineering
University of Zaragoza
Zaragoza, Spain

Habib M. Fardoun HFARDOUN@KAU.EDU.SA
Department of Information Systems
King Abdulaziz University
Jeddah, Saudi Arabia

Sebastián Ventura SVENTURA@UCO.ES
Department of Computer Science and Numerical Analysis
University of Córdoba
Córdoba, Spain
Department of Information Systems
King Abdulaziz University
Jeddah, Saudi Arabia

Editor: Geoff Holmes

Abstract

Active Learning has become an important area of research owing to the increasing number of real-world problems which contain labelled and unlabelled examples at the same time. JCLAL is a Java Class Library for Active Learning which has an architecture that follows strong principles of object-oriented design. It is easy to use, and it allows the developers to adapt, modify and extend the framework according to their needs. The library offers a variety of active learning methods that have been proposed in the literature. The software is available under the GPL license.

Keywords: active learning, framework, java language, object-oriented design

1. Introduction

In the last decade, the study of problems which contain a small number of labelled examples and a large number of unlabelled examples at the same time have received special attention. Currently, there are two main areas that research the learning of models from labelled and unlabelled data, namely Semi-Supervised Learning and Active Learning (AL). AL is concerned with learning accurate classifiers by choosing which instances will be labelled, reducing the labelling effort and the cost of training an accurate model (Settles, 2012).

Currently, there are several software tools which assist the experimentation process and development of new algorithms in the data mining and machine learning areas, such as Rapid Miner, WEKA, Scikit-learn, Orange and KEEL. However, these tools are focused to Supervised and Unsupervised Learning problems.

Some libraries and independent code that implement AL methods can be found on the Internet, such as Vowpal Wabbit, DUALIST, Active-Learning-Scala, TexNLP and LibAct. The Active-Learning-Scala and LibAct libraries are mainly focused to AL, they implement several AL strategies that have been proposed in the literature. On the other hand, Vowpal Wabbit, DUALIST and TexNLP have been designed for a different purpose, but they also include some AL methods.

To date, and in our opinion, there has been insufficient effort towards the creation of a computational tool mainly focused to AL. In our view, a good computational tool is not only a tool which includes the most relevant AL strategies, but also one that is extensible, user-friendly, interoperable, portable, etc.

The above situation motivated the development of the JCLAL framework. JCLAL is an open source software for researchers and end-users to develop AL methods. It includes the most relevant strategies that have been proposed in single-label and multi-label learning paradigms. It provides the necessary interfaces, classes and methods to develop any AL method.

This paper is arranged as follows: Section 2 provides a general description of the JCLAL framework. The Section 3 presents an example for using the software. Finally, the documentation and the requirements of this software are outlined in Section 4.

2. The JCLAL Framework

JCLAL is inspired by the architecture of JCLEC (Ventura et al., 2007; Cano et al., 2014) which is a framework for evolutionary computation. JCLAL provides a high-level software environment to perform any kind of AL method. It has an architecture that follows strong principles of object-oriented programming, where it is common and easy to reuse code. The main features of the library are the following:

- Generic. Through a flexible class structure, the library provides the possibility of including new AL methods, as well as the ability to adapt, modify or extend the framework according to developer's needs.

- User friendly. The library has several mechanisms that offer a user friendly programming interface. It allows users to execute an experiment through an XML configuration file.

- Portable. The library has been coded in the Java programming language. This ensures its portability between all platforms that implement a Java Virtual Machine.

- Elegant. The use of the XML file format provides a common ground for tools development and to integrate the framework with other systems.

- Open Source. The source code is free and available under the GNU General Public License (GPL). It is hosted at SourceForge, GitHub, OSSRH repository provided by Sonatype, and Maven Central Repository.

JCLAL aims to bring the benefits of machine learning open source software (Sönnenburg et al., 2007) to people working in the area of AL. The library offers several state-of-the-art AL strategies for single-label and multi-label learning paradigms. It uses the WEKA (Hall et al., 2009) and MULAN (Tsoumakas et al., 2011) libraries. WEKA is one of the most popular libraries which has several resources on supervised learning algorithms. On the other hand, MULAN is a Java library which includes several multi-label learning algorithms. For future versions, we hope to provide AL strategies related with multi-instance and multi-label-multi-instance learning paradigms.

Currently, the library provides the following single-label AL strategies: Entropy Sampling, Least Confident and Margin Sampling which belong to the Uncertainty Sampling category. Together with the Vote Entropy and Kullback Leibler Divergence strategies which belong to the Query By Committee category. In the Expected Error Reduction category, the Expected 0/1-loss and Expected Log-Loss strategies are included. One AL strategy belongs to the Variance Reduction family. The Information Density framework is also provided. More information about all of these single-label strategies can be found in (Settles, 2012). On the other hand, the following multi-label AL strategies are provided: Binary Minimum (Brinker, 2006), Max Loss (Li et al., 2004), Mean Max Loss (Li et al., 2004), Maximal Loss Reduction with Maximal Confidence (Yang et al., 2009), Confidence-Minimun-NonWeighted (Esuli and Sebastiani, 2009), Confidence-Average-NonWeighted (Esuli and Sebastiani, 2009), Max-Margin Prediction Uncertainty (Li and Guo, 2013) and Label Cardinality Inconsistency (Li and Guo, 2013).

The Stream-Based Selective Sampling and Pool-Based Sampling scenarios are supported. JCLAL provides the interfaces and abstract classes for implementing batch-mode AL methods and other types of oracle. Furthermore, the library has a simple manner of defining new stopping criteria which may change according to the problem. The library contains a structure which allows a set of listeners to simply define the events of an algorithm. The AL methods can be tested using the following evaluation methods: Hold-Out, *k*-fold cross validation, 5X2 cross validation and Leave-One Out. A method for actual deployment is also provided.

The library contains a set of utilities, e.g. algorithms for random number generation, sort algorithms, sampling methods and methods to compute, for example, AUC. A plug-in which permits the integration of the library with WEKA's explorer is also provided.

3. Using JCLAL

The library allows the users to execute an experiment through an XML configuration file as well as directly from Java code. A configuration file comprises a series of parameters required to run an algorithm. Below, an example of a configuration file is shown, which we call `MarginSampling.cfg`.

In this example, a 10-fold cross validation evaluation method is used on the data set `ecoli` located in the folder `datasets`. For each fold, 5% of the training set is selected to construct the labelled set and the rest of the instances form the unlabelled set. A pool-based sampling scenario with the Margin Sampling strategy is used. The Naive Bayes algorithm is used as a base classifier.

```
<experiment>
 <process evaluation-method-type="net.sf.jclal.evaluation.method.kFoldCrossValidation">
  <rand-gen-factory seed="9871234" type="net.sf.jclal.util.random.RanecuFactory"/>
  <file-dataset>datasets/ecoli.arff</file-dataset>
  <stratify>true</stratify>
  <num-folds>10</num-folds>
  <sampling-method type="net.sf.jclal.sampling.unsupervised.Resample">
   <percentage-to-select>5.0</percentage-to-select>
  </sampling-method>
  <algorithm type="net.sf.jclal.activelearning.algorithm.ClassicalALAlgorithm">
   <stop-criterion type="net.sf.jclal.activelearning.stopcriteria.MaxIteration">
      <max-iteration>50</max-iteration>
   </stop-criterion>
   <stop-criterion type="net.sf.jclal.activelearning.stopcriteria.UnlabeledSetEmpty"/>
   <listener type="net.sf.jclal.listener.ClassicalReporterListener">
    <report-title>Margin-Sampling</report-title>
    <report-frequency>1</report-frequency>
    <report-directory>reports/ecoli</report-directory>
    <report-on-file>true</report-on-file>
   </listener>
   <scenario type="net.sf.jclal.activelearning.scenario.PoolBasedSamplingScenario">
    <batch-mode type="net.sf.jclal.activelearning.batchmode.QBestBatchMode">
     <batch-size>1</batch-size>
    </batch-mode>
    <oracle type="net.sf.jclal.activelearning.oracle.SimulatedOracle"/>
    <query-strategy type="net.sf.jclal.activelearning.singlelabel.querystrategy.
    MarginSamplingQueryStrategy">
     <wrapper-classifier type="net.sf.jclal.classifier.WekaClassifier">
       <classifier type="weka.classifiers.bayes.NaiveBayes"/>
     </wrapper-classifier>
    </query-strategy>
   </scenario>
  </algorithm>
 </process>
</experiment>
```

There are several ways to execute an experiment. One way is using the JAR file. For running the experiment just type:

```
java -jar jclal-1.0.jar -cfg "examples/MarginSampling.cfg"
```

After the experiment is run, a summary report which comprises information about the induced classifier and several performance measures is created.

4. Documentation, Requirements and Availability

The library is available under the GNU GPL license. A user manual and developer documentation which describes the software packages, examples, information to include new methods, API reference and running tests, is provided.

The software requires Java 1.7, Apache commons logging 1.1, Apache commons collections 3.2, Apache commons configuration 1.5, Apache commons lang 2.4, JFreeChart 1.0, WEKA 3.7, MULAN 1.4 and JUnit 4.10 (for running tests). There is also a mailing list and a discussion forum for requesting support on using or extending the framework.

Acknowledgments

This research was supported by the Spanish Ministry of Economy and Competitiveness, project TIN-2014-55252-P, and by FEDER funds.

References

K. Brinker. *From Data and Information Analysis to Knowledge Engineering: Proceedings of the 29th Annual Conference of the Gesellschaft für Klassifikation e.V. University of Magdeburg*, chapter On Active Learning in Multi-label Classification, pages 206–213. Springer Berlin Heidelberg, 2006.

A. Cano, J. M. Luna, A. Zafra, and S. Ventura. A classification module for genetic programming algorithms in JCLEC. *Journal of Machine Learning Research*, 1:1–4, 2014.

A. Esuli and F. Sebastiani. *Advances in Information Retrieval: Proceedings of the 31th European Conference on IR Research (ECIR)*, chapter Active Learning Strategies for Multi-Label Text Classification, pages 102–113. Springer Berlin Heidelberg, 2009.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An Update. *SIGKDD explorations*, 11(1):10–18, 2009.

X. Li and Y. Guo. Active learning with multi-label SVM classification. In *Proceedings of the 23th International Joint Conference on Artificial Intelligence*, pages 1479–1485, 2013.

X. Li, L. Wang, and E. Sung. Multi-label SVM active learning for image classification. In *Proceedings of the International Conference on Image Processing (ICIP)*, volume 4, pages 2207–2210. IEEE, 2004.

B. Settles. *Active Learning.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 1st edition, 2012.

S. Sönnenburg, M. L. Braun, C. S. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, K.-R. Müller, F. Pereira, C. E. Rasmussen, G. Rätsch, B. Schölkopf, A. Smola, P. Vincent, J. Weston, and R. C. Williamson. The need for open source software in machine learning. *Journal of Machine Learning Research*, 8:2443–2466, 2007.

G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas. MULAN: a java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

S. Ventura, C. Romero, A. Zafra, J. A. Delgado, and C. Hervás. JCLEC: a java framework for evolutionary computation. *Soft Computing*, 12:381–392, 2007.

B. Yang, J. T. Sun, T. Wang, and Z. Chen. Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 917–926. ACM, 2009.