Journal of Machine Learning Research 12 (2011) 2411-2414 Submitted 8/09; Revised 7/10; Published 7/11

MULAN: A Java Library for Multi-Label Learning

Grigorios Tsoumakas GREG@CSD.AUTH.GR
Eleftherios Spyromitros-Xioufis ESPYROMI@CSD.AUTH.GR
Jozef Vilcek JOJOVILCO@GMAIL.COM
Ioannis Vlahavas VLAHAVAS@CSD.AUTH.GR
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki 54124, Greece

Editor: Cheng Soon Ong

Abstract

MULAN is a Java library for learning from multi-label data. It offers a variety of classification, ranking, thresholding and dimensionality reduction algorithms, as well as algorithms for learning from hierarchically structured labels. In addition, it contains an evaluation framework that calculates a rich variety of performance measures.

Keywords: multi-label data, classification, ranking, thresholding, dimensionality reduction, hierarchical classification, evaluation

1. Multi-Label Learning

A multi-label data set consists of training examples that are associated with a subset of a finite set of labels. Nowadays, multi-label data are becoming ubiquitous. They arise in an increasing number and diversity of applications, such as semantic annotation of images and video, web page categorization, direct marketing, functional genomics and music categorization into genres and emotions.

There exist two major multi-label learning tasks (Tsoumakas et al., 2010): *multi-label classification* and *label ranking*. The former is concerned with learning a model that outputs a bipartition of the set of labels into relevant and irrelevant with respect to a query instance. The latter is concerned with learning a model that outputs a ranking of the labels according to their relevance to a query instance. Some algorithms learn models that serve both tasks. Several algorithms learn models that primarily output a vector of numerical scores, one for each label. This vector is then converted to a ranking after solving ties, or to a bipartition, after *thresholding* (Ioannou et al., 2010).

Multi-label learning methods addressing these tasks can be grouped into two categories (Tsoumakas et al., 2010): *problem transformation* and *algorithm adaptation*. The first group of methods are algorithm independent. They transform the learning task into one or more single-label classification tasks, for which a large body of learning algorithms exists. The second group of methods extend specific learning algorithms in order to handle multi-label data directly. There exist extensions of decision tree learners, nearest neighbor classifiers, neural networks, ensemble methods, support vector machines, kernel methods, genetic algorithms and others.

Multi-label learning stretches across several other tasks. When labels are structured as a tree-shaped hierarchy or a directed acyclic graph, then we have the interesting task of *hierarchical multi-label learning*. *Dimensionality reduction* is another important task for multi-label data, as it is for

any kind of data. When bags of instances are used to represent a training object, then *multi-instance multi-label* learning algorithms are required. There also exist *semi-supervised learning* and *active learning* algorithms for multi-label data.

2. The MULAN Library

The main goal of MULAN is to bring the benefits of machine learning open source software (MLOSS) (Sonnenburg et al., 2007) to people working with multi-label data. The availability of MLOSS is especially important in emerging areas like multi-label learning, because it removes the burden of implementing related work and speeds up the scientific progress. In multi-label learning, an extra burden is implementing appropriate evaluation measures, since these are different compared to traditional supervised learning tasks. Evaluating multi-label algorithms with a variety of measures, is considered important by the community, due to the different types of output (bipartition, ranking) and diverse applications.

Towards this goal, MULAN offers a plethora of state-of-the-art algorithms for multi-label classification and label ranking and an evaluation framework that computes a large variety of multi-label evaluation measures through hold-out evaluation and cross-validation. In addition, the library offers a number of thresholding strategies that produce bipartitions from score vectors, simple baseline methods for multi-label dimensionality reduction and support for hierarchical multi-label classification, including an implemented algorithm.

MULAN is a library. As such, it offers only programmatic API to the library users. There is no graphical user interface (GUI) available. The possibility to use the library via command line, is also currently not supported. Another drawback of MULAN is that it runs everything in main memory so there exist limitations with very large data sets.

MULAN is written in Java and is built on top of Weka (Witten and Frank, 2005). This choice was made in order to take advantage of the vast resources of Weka on supervised learning algorithms, since many state-of-the-art multi-label learning algorithms are based on problem transformation. The fact that several machine learning researchers and practitioners are familiar with Weka was another reason for this choice. However, many aspects of the library are independent of Weka and there are interfaces for most of the core classes.

MULAN is an advocate of open science in general. One of the unique features of the library is a recently introduced experiments package, whose goal is to host code that reproduces experimental results reported on published papers on multi-label learning.

To the best of our knowledge, most of the general learning platforms, like Weka, don't support multi-label data. There are currently only a number of implementations of specific multi-label learning algorithms, but not a general library like MULAN.

3. Using MULAN

This section presents an example of how to setup an experiment for empirically evaluating two multi-label algorithms on a multi-label data set using cross-validation. We create a new Java class for this experiment, which we call `MulanExp1.java`.

The first thing to do is load the multi-label data set that will be used for the empirical evaluation. MULAN requires two text files for the specification of a data set. The first one is in the ARFF format of Weka. The labels should be specified as nominal attributes with values "0" and "1" indicating

absence and presence of the label respectively. The second file is in XML format. It specifies the labels and any hierarchical relationships among them. Hierarchies of labels can be expressed in the XML file by nesting the `label` tag.

In our example, the two filenames are given to the experiment class through command-line parameters.

```
String arffFile = Utils.getOption("arff", args);
String xmlFile = Utils.getOption("xml", args);
```

Loading the data can then be done using the following code.

```
MultiLabelInstances data = new MultiLabelInstances(arffFile, xmlFile);
```

The next step is to create an instance from each of the two learners that we want to evaluate. We will create an instance of the RA*k*EL and ML*k*NN algorithms. RA*k*EL is actually a meta algorithm and can accept any multi-label learner as a parameter, but is typically used in conjunction with the Label Powerset (LP) algorithm. In turn LP is a transformation-based algorithm and it accepts a single-label classifier as a parameter. We will use Weka's J48 algorithm for this purpose. ML*k*NN is an algorithm adaptation method that is based on *k*NN.

```
RAkEL learner1 = new RAkEL(new LabelPowerset(new J48()));
MLkNN learner2 = new MLkNN();
```

We then declare an `Evaluator` object that handles empirical evaluations and an object of the `MultipleEvaluation` class that stores cross-validation results.

```
Evaluator eval = new Evaluator();
MultipleEvaluation results;
```

To actually perform the evaluations we use the `crossValidate` method of the `Evaluator` class. This returns a `MultipleEvaluation` object, which we can print to see the results in terms of all applicable evaluation measures available in MULAN.

```
int numFolds = 10;
results = eval.crossValidate(learner1, data, numFolds);
System.out.println(results);
results = eval.crossValidate(learner2, data, numFolds);
System.out.println(results);
```

For running the experiment, we can use the emotions data (emotions.xml and emotions.arff) that are available together with the MULAN distribution. Other open access multi-label data sets can be found at `http://mulan.sourceforge.net/datasets.html`. Assuming the experiment's source file is in the same directory with emotions.arff, emotions.xml, weka.jar and mulan.jar from the distribution package, then to run this experiment we type the following commands (under Linux use : instead of ; as path separator).

```
javac -cp mulan.jar;weka.jar MulanExp1.java
java -cp mulan.jar;weka.jar;. MulanExp1 -arff emotions.arff -xml emotions.xml
```

The `mulan.examples` package includes additional examples of usage of the MULAN API, such as how to do hold-out and cross-validation learning experiments, how to store/load learned models, perform dimensionality reduction, estimate data set statistics and obtain predictions on test sets with unknown label values.

4. Documentation, Requirements and Availability

MULAN's online documentation[1] contains user oriented sections, such as *getting started with* MULAN and *the data set format of* MULAN, as well as developer-oriented sections, such as *extending* MULAN, *API reference* and *running tests*. There is also a mailing list for requesting support on using or extending MULAN.

MULAN is available under the GNU GPL licence. The current version of the library[2] is 1.3.0. It requires Java version 1.6, Weka version 3.7.3 and JUnit version 4.5 (only for running tests).

Acknowledgments

We would like to thank several people that have contributed pieces of code to the library. First and most importantly Robert Friberg for his help on the first steps towards MULAN. Then, the following people in alphabetical order: S. Bakirtzoglou, W. Cheng, M. Ioannou, I. Katakis, S.-H. Park, E. Rairat, G. Sakkas, G. Saridis, K. Sechidis, E. Stachtiari and G. Traianos.

References

Marios Ioannou, George Sakkas, Grigorios Tsoumakas, and Ioannis Vlahavas. Obtaining bipartitions from score vectors for multi-label classification. *IEEE International Conference on Tools with Artificial Intelligence*, 1:409–416, 2010. ISSN 1082-3409.

Sören Sonnenburg, Mikio L. Braun, Cheng Soon Ong, Samy Bengio, Leon Bottou, Geoffrey Holmes, Yann LeCun, Klaus-Robert Müller, Fernando Pereira, Carl Edward Rasmussen, Gunnar Rätsch, Bernhard Schölkopf, Alexander Smola, Pascal Vincent, Jason Weston, and Robert Williamson. The need for open source software in machine learning. *JMLR*, 8:2443–2466, 2007.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, chapter 34, pages 667–685. Springer, 2nd edition, 2010.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

1. Available at `http://mulan.sourceforge.net/documentation.html`.
2. Available for download at `http://sourceforge.net/projects/mulan/`.