SparkBOOST, an Apache Spark-based boosting library

Tiziano Fagni (tiziano.fagni@isti.cnr.it)
Andrea Esuli (andrea.esuli@isti.cnr.it)

Istituto di Scienze e Tecnologie dell'Informazione (ISTI)
Italian National Research Council (CNR)
Pisa, Italy

March 3, 2016

Abstract

SparkBOOST is a Java library built over Apache Spark that provides a distributed implementation of AdaBoost.MH and MP-Boost machine learning algorithms. These boosting algorithms are known to be very effective and robust to overfitting in many application domains, e.g. in natural language processing contexts. SparkBOOST offers to developers a fast way to scale these algorithms to large scale problems, where one needs to build classifiers from very large training datasets or simply needs to quickly classify huge stream of documents. The library can be integrated into custom programs by using a simple API. The SparkBOOST implementation also provides some command line tools to perform learning and classification on data sources available in LibSVM format.

1 Introduction

Boosting[2] is a powerful machine learning (ML) technique used to build very effective models in supervised learning problems. The main idea of boosting is to build a strong learner by using contributions coming from a committee of weak learners. In the last years boosting techniques have been used with success over various applications domains, in particular the algorithms AdaBoost.MH[3] and MP-Boost[1] have shown very good accuracy as automatic text classifiers on several benchmarks. One of the main advantages that these boosting algorithms have over other well-known ML methods, such as Support Vector Machines (SVMs), is the fact that they actually have no parameters that require fine-tuning to obtain good performances. This makes boosting algorithms a first-choice tool in all the cases in which the cost of parameter tuning is not acceptable because of time or resources limitations.

The exponential growth of many social network platforms, and models of social sharing, of the last years have produced an unprecedented explosion of data availability (also known as the BIG DATA era). Most of this data could find application into ML process. Apache Spark[4] is currently one of the leading open source distributed data processing platform available on the market able to manage huge quantities of data in this scenario. Spark offers to developers MLlib, a ML library, but this library is relatively limited in terms of available algorithms types. In particular, no boosting algorithms are offered. The main aim of SparkBOOST[1] is to fill this gap by providing a Java implementation of the two boosting algorithms mentioned above on the Apache Spark platform.

2 Library usage

SparkBOOST is provided in the form of a Java library which can be imported in your own source code. The library also gives to developers a set of command line tools to quickly train and apply classifiers. The command line tools work with data sources in LibSVM format.

2.1 Compile the source code

Apache Maven and a Java 8 compiler are required to build the software from its source code. The following commands will download and build SparkBOOST:

```
git clone https://github.com/tizfa/sparkboost.git
cd sparkboost
mvn clean
mvn -P shading package
```

This set of commands will build a software bundle containing all the necessary Spark libraries. You can find the software bundle in the `target` subdirectory of the library root directory.

2.2 Using command line tools

2.2.1 How to build classifiers

SparkBOOST offers to developers or end-users a set of command line tools to quickly build boosting classifiers and apply them to specific problems. All command line tools assume that the input data source is available in LibSVM format[2]. You can build both multilabel multiclass classifiers or binary classifiers (use the flag `-b` to specify this behaviour in the tools).

To build an MP-Boost classifier for a specific dataset file `path/to/datasetFile`, use the following commands:

[1]https://github.com/tizfa/sparkboost
[2]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

```
java -cp ./target/sparkboost-0.6-bundle.jar
    it.tizianofagni.sparkboost.MPBoostLearnerExe path/to/datasetfile
    path/to/modelOutput numIterations sparkMasterName parallelismDegree
```

where `path/to/modelOutput` is the output file where the generated classifier will be saved, `numIterations` is the number of iterations run by the algorithm, `sparkMasterName` is the name of Spark master host (or `local[*]` to execute the process using Spark in local mode) and `parallelismDegree` is the number of processing units to be used when executing the algorithm.

The command to build an AdaBoost.MH classifier is exactly the same as before, just using the class `AdaBoostMHLearnerExe` class instead of the `MPBoostLearnerExe` class.

2.2.2 How to classify examples

After learning a classification model, we can classify a test dataset[3] using the command:

```
java -cp ./target/sparkboost-0.6-bundle.jar
    it.tizianofagni.sparkboost.BoostClassifierExe datasetfile
    classifierModel outputResultsFile sparkMasterName parallelismDegree
```

where `datasetFile` is the input file containing the dataset test examples, `classifierModel` is the file containing the previously learned classifier model, `outputResultsFile` is the ouput file containing classification results, `sparkMasterName` is the name of Spark master host (or `local[*]` to execute the process using Spark in local mode) and `parallelismDegree` is the number of processing units to be used when executing the algorithm.

2.3 Java API

The following in an example of how to use the SparkBOOST API to train and apply a classifier to data.

A learner is prepared by setting its Spark context and the iteration and parallelism parameters:

```
JavaSparkContext sc = ... // Spark context to use;

// Create and configure AdaBoost.MH learner. For MP-Boost, just use the
    class
// MPBoostLearner.
AdaBoostMHLearner learner = new AdaBoostMHLearner(sc);
learner.setNumIterations(numIterations);
learner.setParallelismDegree(parallelismDegree);
```

[3]It does not matter if the model has been built with MP-Boost or AdaBoost.MH learner, they share the same format for classification models.

If the training data is saved in LibSvm format in a file, the file can be passed directly to the learner:

```
// Build a new classifier. Here we assume that the training data is
    available in
// the input file which is written in LibSvm format.
BoostClassifier classifier = learner.buildModel(inputFile, labels,
    binaryProblem);
```

Other sources of data can be provided in input as an RDD with MultilabelPoint items:

```
// Or you can prepare yourself the training data by generating an RDD
    with items
// of type MultilabelPoint...
JavaRDD<MultilabelPoint> trainingData = ...
// and then train the classifier.
BoostClassifier classifier = learner.buildModel(trainingData);
```

The learned models are saved and loaded using Spark `DataUtils` methods:

```
// Save classifier in outputModelPath using any valid syntax
// allowed by Spark/Hadoop.
DataUtils.saveModel(sc, classifier, outputModelPath);

// Load boosting classifier from disk.
classifier = DataUtils.loadModel(sc, outputModelPath);
```

A model can be applied to test data from a file in LibSvm format or from an RDD with MultilabelPoint items:

```
// Classify documents contained in "testInputFile", a file in libsvm
    format.
ClassificationResults results = classifier.classify(sc, testInputFile,
    parallelismDegree, labels, binaryProblem);

// or classify documents available in an RDD.
JavaRDD<MultilabelPoint> rdd = ...
results = classifier.classify(sc, rdd, parallelismDegree);

// Print results in a StringBuilder.
StringBuilder sb = new StringBuilder();
sb.append("**** Effectiveness\n");
sb.append(results.getCt().toString() + "\n");
sb.append("********\n");
for (int i = 0; i < results.getNumDocs(); i++) {
   int docID = results.getDocuments()[i];
   int[] labels = results.getLabels()[i];
   int[] goldLabels = results.getGoldLabels()[i];
   sb.append("DocID: " + docID + ", Labels assigned: " +
       Arrays.toString(labels) + ", Labels scores: " +
       Arrays.toString(results.getScores()[i]) + ", Gold labels: " +
       Arrays.toString(goldLabels) + "\n");
}
```

3 Conclusion

We have presented SparkBOOST, an open source Java library, built over Apache Spark, that provides a distributed implementation of two popular boosting ML algorithms. Using Apache Spark under the hood, the library allows to easily scale the algorithms to big data contexts. Programmers can integrate SparkBOOST into their workflow by using the simple Java API or by using the provided command line tools.

References

[1] A. Esuli, T. Fagni, and F. Sebastiani. MP-Boost: A Multiple-Pivot Boosting Algorithm and Its Application to Text Categorization. In F. Crestani, P. Ferragina, and M. Sanderson, editors, *String Processing and Information Retrieval*, number 4209 in Lecture Notes in Computer Science, pages 1–12. Springer Berlin Heidelberg, Oct. 2006.

[2] R. E. Schapire. *Nonlinear Estimation and Classification*, chapter The Boosting Approach to Machine Learning: An Overview, pages 149–171. Springer New York, New York, NY, 2003.

[3] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2):135–168.

[4] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.