MULTIBOOST.PY

A PYTHONIC, flexible, extensible, multiclass implementation of ensemble learning algorithms.

Djalel Benbouzid
Kégl Balázs

Center for Data Science, Université Paris-Saclay

1 Prelude

ADABOOST (Freund and Schapire, 1997) is one of the best off-the-shelf learning methods developed in the last fifteen years. It constructs a classifier in an incremental fashion by adding simple classifiers to a pool, and uses their weighted "vote" to determine the final classification. ADABOOST was later extended to multi-class classification problems (Schapire and Singer, 1999). Although various other attempts have been made handle the multi-class setting, ADABOOST.MH remains the gold standard of multi-class and multi-label[1] boosting due to its simplicity and versatility (Kégl, 2014).

2 Motivation

Despite the simplicity and the practical success of the ADABOOST, there are relatively few off-the-shelf implementations available in the free software market. Whereas binary ADABOOST with decision stumps is easy to code, multi-class ADABOOST.MH and complex base learners are not straightforward to implement efficiently.

In the landscape of the available implementations, MULTIBOOST (Benbouzid et al., 2012) offers a modular implementation of ADABOOST.MH, it is developed in C++ and it includes a various palette of weak learners as well as other strong learners. The modular architecture of MULTIBOOST showed a great advantage for quickly implementing new boosting algorithms or new weak learners without having to modify the rest of the code. In particular, the base learner decomposition proposed by Kégl (2014) and already implemented in MULTIBOOST allows to turn any binary classifier into a multi-class/multi-label classifier. In addition to its practical aspect, this setup also showed outstanding performance in comparison with other boosting algorithms and even with the standard implementation of ADABOOST.MH.

During the recent past years, the data science community largely adopted PYTHON as a common programming language for machine learning and data analysis, which naturally led to the flourishing of rich and versatile machine learning toolboxes such as SCIKIT-LEARN

[1] In the multi-label setting, one observation can belong to more than one class.

(Pedregosa et al., 2011) and MDP[2]. Furthermore, data analysis often requires the combination of many learning algorithms into a workflow that tend to be automatized. In order to make MULTIBOOST part of this workflow, we would like MULTIBOOST to be "interfaceable" with the PYTHON software ecosystem, while keeping its efficiency asset.

3 Specifications

3.1 Language and building blocks

MULTIBOOST.PY will be mainly implemented in the PYTHON, however, some critical parts (the hot spots) will need to be implemented in CYTHON, after careful code profiling.

In terms of dependencies, MULTIBOOST.PY will rely on the standard data analysis tools in PYTHON, ie. NUMPY, SCIPY and PANDAS. Furthermore, all the classifiers (weak and strong) will follow the interfaces defined in SCIKIT-LEARN so to make their usage totally accessible and compliant with the other tools provided by this library.

3.2 Library / Standalone

MULTIBOOST.PY is meant to be an general boosting library as well as a turn-key, standalone software. As a library, it should allow the quick implementation of new algorithms. At the same time, the package will provide a command-line interface, allowing to read datasets in some specific formats.

3.3 Modularity

The main asset of MULTIBOOST.PY over other implementations of boosting algorithms lies in a *multi-level modularity*. The first level of modularity is inherent to ensemble methods as they decompose the ensemble learner (or strong learner in the case of boosting) from the base (or weak) learner. Furthermore, inspired by the C++ implementation of MULTIBOOST, a specific class of base learners allows to extend any binary classifier to multi-class classification. For a K-class problem, the base classifier is decomposed like so

$$\mathbf{h}(\cdot) = \alpha \times \phi(\cdot) \times \mathbf{v}$$

where α is the base learner coefficient, $\phi \in \{\pm 1\}$ is a binary classifier and $\mathbf{v} \in \{\pm 1\}^K$ is a vote vector that is optimized separately.

This decomposition, which orthogonally learns a binary classifier, the ϕ function, and sets the vales of the $\mathbf{v}$ vector, greatly eases the implementation of new multi-class base learners, by only focusing on the ϕ function.

As a third and last level of modularity, MULTIBOOST.PY will also implement the ANYBOOST framework (Mason et al., 2000) in order to easily derive boosting algorithms from proper loss functions.

[2] http://mdp-toolkit.sourceforge.net/

4 Features

4.1 Strong learners

- AdaBoost.MH.
- AnyBoost.

4.2 Base learners

- Decision stump learner (for continuous features)
- Indicator learner (for nominal features)
- The Hamming Trees (MULTIBOOST) meta-algorithm.
- Mixed-types features (continuous, nominal)

4.3 I/O modes

- Input
  - NUMPY arrays
  - Command-line arguments
  - Configuration file
- Output
  - Learning curve (iteration-wise metrics)
  - Classification scores
  - Flexible output system for the easy implementation of new metrics to output.

4.4 Documentation and tests

The classes and functions will be cautiously documented with `doctrings`. The development will be fully test-driven.

5 Schedule

Project starting June 1, 2014. The first tasks would be to:

- implement the AdaBoost outer loop in python,
- port the Hamming trees of MULTIBOOST in python,
- test the combination of the two aforementioned and compare the results with MULTIBOOST,
- profile the code and potentially rewrite some portions in CYTHON

References

Benbouzid, D., Busa-Fekete, R., Casagrande, N., Collin, F.-D., and Kégl, B. (2012). MultiBoost: a multi-purpose boosting package. *Journal of Machine Learning Research*, 13:549–553.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139.

Kégl, B. (2014). The return of AdaBoost.MH: multi-class Hamming trees. In *International Conference on Learning Representations*.

Mason, L., Bartlett, P., Baxter, J., and Frean, M. (2000). Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems*, volume 12, pages 512–518. The MIT Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336.