Journal of Machine Learning Research 21 (2020) 1-5 Submitted 5/19; Revised 12/19; Published 1/20

MFE: Towards reproducible meta-feature extraction

Edesio Alcobaça EDESIO@USP.BR
Felipe Siqueira FELIPE.SIQUEIRA@USP.BR
Adriano Rivolli RIVOLLI@UTFPR.EDU.BR
Luís P. F. Garcia LUIS.GARCIA@UNB.BR
Jefferson T. Oliva JEFFERSONOLIVA@UTFPR.EDU.BR
André C. P. L. F. de Carvalho ANDRE@ICMC.USP.BR
Institute of Mathematical and Computer Sciences
University of São Paulo
Av. Trabalhador São-carlense, 400, São Carlos, São Paulo 13560-970, Brazil

Editor: Alexandre Gramfort

Abstract

Automated recommendation of machine learning algorithms is receiving a large deal of attention, not only because they can recommend the most suitable algorithms for a new task, but also because they can support efficient hyper-parameter tuning, leading to better machine learning solutions. The automated recommendation can be implemented using meta-learning, learning from previous learning experiences, to create a meta-model able to associate a data set to the predictive performance of machine learning algorithms. Although a large number of publications report the use of meta-learning, reproduction and comparison of meta-learning experiments is a difficult task. The literature lacks extensive and comprehensive public tools that enable the reproducible investigation of the different meta-learning approaches. An alternative to deal with this difficulty is to develop a meta-feature extractor package with the main characterization measures, following uniform guidelines that facilitate the use and inclusion of new meta-features. In this paper, we propose two Meta-Feature Extractor (MFE) packages, written in both Python and R, to fill this lack. The packages follow recent frameworks for meta-feature extraction, aiming to facilitate the reproducibility of meta-learning experiments.

Keywords: Machine Learning, AutoML, Meta-Learning, Meta-Features

1. Introduction

Machine learning (ML) algorithms have been successfully used to analyze and solve a large number of real-world problems. Since each ML algorithm has an inductive bias, the choice of the most suitable one is not a trivial task. Meta-learning (MtL) can be used to recommend suitable ML algorithms for a data set. For such, MtL looks for a function able to map the characteristics of a data set to the expected performance of a selected set of algorithms (Smith-Miles, 2009). This process is commonly referred to as an algorithm recommendation problem (Brazdil et al., 2008). The MtL literature presents proposals for the recommendation of ML algorithms (Reif et al., 2014), preprocessing techniques (Garcia et al., 2016), and the need for hyperparameter tuning (Mantovani et al., 2019). MtL can also

be used to recommend hyperparameter settings (Brazdil et al., 2008) and as a component of automated machine learning (AutoML) systems (Feurer et al., 2015).

A typical MtL task uses a meta-data set, where each meta-instance has, as predictive attributes (meta-features), features extracted from a data set. Furthermore, a meta-data set has as a target attribute (meta-target), the algorithm that in previous experiments reached the best predictive performance for the analyzed data set (Brazdil et al., 2008). The standard meta-features can be organized into five groups (Rivolli et al., 2018): simple, statistical, information-theoretic, model-based and landmarking (Smith-Miles, 2009; Reif et al., 2014).

MtL experiments are considered difficult to be replicated because essential details about the experiments are missing in many studies. These details are usually the data characterization process and the meta-feature hyperparameters used. Despite some attempts to overcome this problem, such as Data Characteristic Tools (DCT) (Lindner and Studer, 1999), OpenML (OML) (Vanschoren et al., 2013) and BYUMetalearn (BYUM)[1], they are not enough. DCT includes a few simple, statistical, and information-theoretic meta-features. OML does automatic data characterization, however limiting the user choice (e.g., it is not possible to combine measures with different summary functions). BYUM, a package under development, presents only some standard meta-features. Furthermore, these tools do not follow recent frameworks and formalization for meta-features (Pinto et al., 2016; Rivolli et al., 2018).

The framework presented in Pinto et al. (2016) organizes the automatic meta-feature extraction process, but it does not address the reproducibility problem. Rivolli et al. (2018) extend that work proposing a framework to systematize meta-feature extraction. This extended framework addressed the standard implementation and use of meta-features.

The Meta Feature Extractor (MFE) packages complement this survey by providing tools for meta-feature extraction that enables the MtL research reproducibility. They agree with the formalization presented in Rivolli et al. (2018) and implement the main MtL meta-features. These packages, available in Python (`pymfe`[2]) and R (`mfe`[3]), are detailed in this work.

The remainder of this paper is organized as follows. Section 2 describes the meta-feature definition adopted. Section 3 describes the meta-features framework, the groups of meta-features implemented and the proposed packages, `pymfe` for Python and `mfe` for R. Finally, Section 4 concludes this text and points out future work directions.

2. Meta-feature Formal Definition

In the MtL literature, meta-features are measures used to characterize data sets and/or their relations with algorithm bias (Brazdil et al., 2008). Rivolli et al. (2018) formalize the meta-feature definition as a set of k values, extracted from a data set $\mathcal{D}$, by a function $f\colon \mathcal{D} \to \mathbb{R}^k$, described by Equation 1. In this equation, $m\colon \mathcal{D} \to \mathbb{R}^{k'}$ is a characterization measure applied to the data set $\mathcal{D}$, $\sigma\colon \mathbb{R}^{k'} \to \mathbb{R}^k$ is a summarization function, and h_m and h_σ are hyperparameters for m and σ, respectively.

1. `https://github.com/byu-dml/metalearn`
2. `https://pypi.org/project/pymfe/`
3. `https://cran.r-project.org/package=mfe`

$$f\big(\mathcal{D}\big) = \sigma\big(m(\mathcal{D}, h_m), h_\sigma\big) \tag{1}$$

The measure m can extract more than one value from each data set, *i.e.*, k' can vary according to $\mathcal{D}$, which can be mapped to a vector of fixed length k using a summarization function σ. In MtL, where a fixed cardinality is needed, the summarization functions can be, e.g., *mean*, *minimum*, *maximum*, *skewness* and *kurtosis*. Thus, a meta-feature is a combination of a measure and a summarization function.

The MFE implementation packages follow the standardization proposed in Rivolli et al. (2018) and implement meta-features from the previously mentioned groups. They implement cutting edge meta-features, including more than 90 measures and 12 summarization functions. Both packages have the same measures and produce similar output for each measure and summarization function, allowing cross-platform executions. The implemented meta-features and their taxonomy classification and mathematical definitions are presented in Rivolli et al. (2018).

3. Project Overview

The MFE packages are an initiative to increase reproducibility in MtL research, bring formalization, categorization, and implementation of cutting edge meta-features. The proposed packages offer a broad set of meta-features and are also open to anyone who wants to contribute. Some facilities are adopted to manage the MFE updates and continuous expansion:

Software Quality: To ensure code quality, numerous unit tests are performed with high code coverage (more than 90% on `pymfe` and `mfe`). Moreover, `pymfe` assures code consistency by following `PEP8` Python code convention, and `mfe`, `testthat` package convention. Inclusion of new code is automatically checked, and quality metrics are provided to validate and enforce continuous software quality.

Continuous Integration: Travis CI is used to allow continuous code integration, building, and software test. Thus, both users and developers can easily use and contribute to the packages.

Community-based development: External contributions are welcome and encouraged. They can be included via Git and GitHub. These tools allow collaborative programming, issue tracking, code integration, and discussion about new ideas.

Documentation: Reliable and consistent documentation are provided using `sphinx` (Python version), `vignettes` and `roxygen2` (the last two in R version). Both show how to extract the meta-features for different configurations.

The Python version has widespread and robust open source libraries, such as `numpy`, `sklearn`, and `scipy`. The `pymfe` implementation, which is sklearn-inspired, provides two methods: *(i)* `fit` computes all data transformations and pre-computations required for the selected meta-features; *(ii)* `extract` applies the characterization measures followed by summarization returning the meta-features. Similarly, the R version uses robust open sources libraries, such as the `stats` and `utils` but also more advanced libraries like `e1071`, `rpart`, `infotheo` and `rrcov`. The meta-fetures are extracted by the function `metafeatures`.

The user has complete control of the meta-feature extraction in both packages. For a given data set, meta-features can be selected individually, in groups, or all of them. It

is also possible to choose none or as many summary functions as wished. The packages compute meta-features faster, performing several initial pre-computations of routines that are common to various measures, avoiding code re-execution.

Table 1 compares the main characteristics of MFE against existing alternatives. The comparison includes the meta-feature groups available, the number of extracted meta-features, and whether their extraction is systematic. As can be seen, MFE support six more meta-feature groups than these alternatives: relative landmarking (Rivolli et al., 2018), subsampling landmarking (Soares et al., 2001), clustering-based (Pimentel and de Carvalho, 2019), concept (Rivolli et al., 2018), itemset (Song et al., 2012) and complexity (Lorena et al., 2019). Moreover, the MFE packages offers the most extensive set and follow recent frameworks.

Name	Meta-feature Groups											# Meta features	Systematic
	Simple	Statistical	Info. Theo.	Model Based	Landmarking			Clustering	Concept	Itemset	Complexity		
					∘	Rel.	Sub.						
DCT	✓	✓	✓		✓							21	no
OML	✓	✓	✓	✓	✓							226	no
BYUM *v0.5.4*	✓	✓	✓	✓	✓							231	no
PyMFE *v0.2.0*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	1572	yes

Table 1: Comparison between meta-feature tools. The ∘ indicates simple landmarking. Rel and Sub are relative and subsample landmarking, respectively.

The packages are available on GitHub[4,5]. They have been visited/cloned many times and contain more than 900 commits at the edition time. Additional information, such as installation, documentation, and examples are provided and centralized on GitHub.

4. Conclusion

The MFE packages followed frameworks that allow the systematic meta-features extraction for MtL, ensuring experimental reproducibility. Moreover, as they are open-source software, it is possible to share and modify the code, stimulating collaboration among MtL researchers. Future work directions include updating the packages with new meta-features soon after their proposal. Also, address some limitations, such as support for missing values and extraction of meta-features for regression tasks.

Acknowledgments

This study was funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq, FAPESP (grants 2018/14819-5, 2016/18615-0 and 2013/07375-0) and Intel Inc., for providing hardware resource and DevCloud, used in part of the experiments. We also would like to thank Davi Pereira-Santos, Saulo Martiello Mastelini for their comments.

4. `https://github.com/ealcobaca/pymfe`
5. `https://github.com/rivolli/mfe`

References

P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. Springer Science & Business Media, 2008.

M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pages 2962–2970, 2015.

L. P. F. Garcia, A. C. P. L. F. de Carvalho, and A. C. Lorena. Noise detection in the meta-learning level. *Neurocomputing*, 176:14–25, 2016.

G. Lindner and R. Studer. Ast: Support for algorithm selection with a cbr approach. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 418–423, 1999.

A. C. Lorena, L. P. F. Garcia, J. Lehmann, M. C. P. Souto, and T. K. Ho. How complex is your classification problem?: A survey on measuring classification complexity. *Association for Computing Machinery Computing Surveys (CSUR)*, 52(5):107, 2019.

R. G. Mantovani, A. L. D. Rossi, E. Alcobaça, J. Vanschoren, and A. C. P. L. F. de Carvalho. A meta-learning recommender system for hyperparameter tuning: predicting when tuning improves svm classifiers. *Information Sciences*, 2019.

B. A. Pimentel and A. C. P. L. F. de Carvalho. A new data characterization for selecting clustering algorithms using meta-learning. *Information Sciences*, 477:203–219, 2019.

F. Pinto, C. Soares, and J. Mendes-Moreira. Towards automatic generation of metafeatures. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 215–226, 2016.

M. Reif, F. Shafait, M. Goldstein, T. Breuel, and A. Dengel. Automatic classifier selection for non-experts. *Pattern Analysis and Applications*, 17(1):83–96, 2014.

A. Rivolli, L. P. F. Garcia, C. Soares, J. Vanschoren, and A. C. P. L. F. de Carvalho. Towards reproducible empirical research in meta-learning. *Computing Research Repository*, arXiv:1808.10406v2, 2018. URL `http://arxiv.org/abs/1808.10406v2`.

K. A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *Association for Computing Machinery Computing Surveys (CSUR)*, 41(1):1–25, 2009.

C. Soares, J. Petrak, and P. Brazdil. Sampling-based relative landmarks: Systematically test-driving algorithms before choosing. In *Portuguese Conference on Artificial Intelligence*, pages 88–95. Springer, 2001.

Q. Song, G. Wang, and C. Wang. Automatic recommendation of classification algorithms based on data set characteristics. *Pattern Recognition*, 45(7):2672–2689, 2012.

J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked science in machine learning. *Special Interest Group on Knowledge Discovery in Data Explorations Newsletter*, 15(2):49 – 60, 2013.