6th ICML Workshop on Automated Machine Learning (2019)

Automating Multi-Label Classification Extending ML-Plan

Marcel Wever marcel.wever@upb.de
Felix Mohr felix.mohr@upb.de
Alexander Tornede alexander.tornede@upb.de
Eyke Hüllermeier eyke@upb.de
Heinz Nixdorf Institut, Paderborn University, Paderborn, Germany

Abstract

Existing tools for automated machine learning, such as Auto-WEKA, TPOT, auto-sklearn, and more recently ML-Plan, have shown impressive results for the tasks of single-label classification and regression. Yet, there is only little work on other types of machine learning problems so far. In particular, there is almost no work on automating the engineering of machine learning solutions for multi-label classification (MLC). We show how the scope of ML-Plan, an AutoML-tool for multi-class classification, can be extended towards MLC using MEKA, which is a multi-label extension of the well-known Java library WEKA. The resulting approach recursively refines MEKA's multi-label classifiers, nesting other multi-label classifiers for meta algorithms and single-label classifiers provided by WEKA as base learners. In our evaluation, we find that the proposed approach yields strong results and performs significantly better than a set of baselines we compare with.

1. Introduction

In recent years, the field of AutoML has made significant progress in developing techniques for automating the task of model selection and hyperparameter tuning. State-of-the-art AutoML tools (Thornton et al., 2013; Komer et al., 2014; Feurer et al., 2015; Mohr et al., 2018b) have shown impressive results for binary and multinomial classification problems. We refer to this type of problems as single-label classification (SLC) in the following.

However, other learning problems, including multi-label classification (MLC), have received much less attention so far. In MLC, instead of predicting only a single class label for an instance, an entire subset of "relevant" labels is predicted. Learning algorithms for MLC have been designed by either adapting the learning algorithm itself or by reducing the original MLC problem to (multiple instances of) the SLC setting. The latter can be considered as a meta-learning technique with a single-label classifier as a base learner.

From an AutoML perspective, automating the configuration of a multi-label classifier is especially challenging, as these reduction techniques introduce deeper hierarchical structures. More specifically, while the configuration of a multi-label classifier's base learner is equivalent to the previous AutoML task for SLC, the meta-strategies for the multi-label classifiers themselves create another level of the hierarchy. The effect on the complexity of the search space is especially strong, because the evaluations are even more expensive.

In this paper, we propose the AutoML tool ML^2-Plan (Multi-Label ML-Plan) to configure multi-label classifiers based on ML-Plan. The latter provides a suitable basis to start from, especially due to its ability to model hierarchical dependencies by means of techniques from hierarchical task network (HTN) planning (Georgievski and Aiello, 2015). Besides, ML-Plan has already been applied to deeper recursive structures in previous work (Wever et al., 2018b). Apart from the work by de Sá et al. (2017, 2018), which uses evolutionary algorithms, we are not aware of previous work on automated multi-label classification.

We compare ML^2-Plan to a random search, a genetic algorithm (de Sá et al., 2017), and a grammar-based genetic programming approach (de Sá et al., 2018). Empirically, we show that our approach performs particularly well and significantly outperforms the baselines.

2. Multi-Label Classification

In contrast to conventional (single-label) classification, the setting of *multi-label classification* (MLC) allows an instance to belong to several classes simultaneously, i.e., to be assigned several labels at the same time. For example, a single image could be tagged simultaneously with labels `Sun` and `Beach` and `Sea`.

More formally, let $\mathcal{X}$ denote an instance space, and let $\mathcal{L} = \{\lambda_1, \ldots, \lambda_m\}$ be a finite set of class labels. We assume that an instance $\boldsymbol{x} \in \mathcal{X}$ is (non-deterministically) associated with a subset of labels $L \in 2^{\mathcal{L}}$; this subset is often called the set of relevant labels, while the complement $\mathcal{L} \setminus L$ is considered as irrelevant for $\boldsymbol{x}$. We identify a set L of relevant labels with a binary vector $\boldsymbol{y} = (y_1, \ldots, y_m)$, in which $y_i = 1$ iff $\lambda_i \in L$. By $\mathcal{Y} = \{0,1\}^m$ we denote the set of possible labelings.

In general, a multi-label classifier $\boldsymbol{h}$ is a mapping $\mathcal{X} \to \mathcal{Y}$. For a given instance $\boldsymbol{x} \in \mathcal{X}$, it returns a prediction in the form of a vector $\boldsymbol{h}(\boldsymbol{x}) = \big(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x})\big)$. The problem of MLC can be stated as follows: Given training data in the form of a finite set of observations $\big\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\big\}_{i=1}^{N} \subset \mathcal{X} \times \mathcal{Y}$, the goal is to learn a classifier $\boldsymbol{h} : \mathcal{X} \to \mathcal{Y}$ that generalizes well beyond these observations in the sense of minimizing the risk with respect to a specific loss function. Various loss functions are commonly used in MLC. Let $\mathcal{D}_{\text{test}} = (\mathcal{X}_{\text{test}}, \mathcal{Y}_{\text{test}}) \subset \mathcal{X}^S \times \mathcal{Y}^S$ be a test set of size S, where the ith entry $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{im}) \in \mathcal{Y}_{\text{test}}$ represents the labeling of the ith instance $\boldsymbol{x}_i \in \mathcal{X}_{\text{test}}$. Further, let $H \subset \mathcal{Y}^S$ with the ith entry given by $h(\boldsymbol{x}_i)$. Then, the subset 0/1 loss (exact match) is defined as[1]

$$L_{0/1}(\mathcal{Y}_{\text{test}}, H) = \frac{1}{S}\sum_{i=1}^{S} [\![\boldsymbol{y}_i \neq \boldsymbol{h}(\boldsymbol{x_i})]\!] \ ,$$

and the Hamming loss as

$$L_H(\mathcal{Y}_{\text{test}}, H) = \frac{1}{S}\sum_{i=1}^{S} \frac{1}{m}\sum_{j=1}^{m} [\![y_{ij} \neq h_j(\boldsymbol{x_i}))]\!] \ .$$

In slightly different tasks, such as ranking and probability estimation, the prediction of a classifier is not restricted to binary vectors. Instead, a hypothesis $\boldsymbol{h}$ is a mapping $\mathcal{X} \to \mathbb{R}^m$, which assigns scores to labels. Corresponding predictions also require other loss functions. An example is the rank loss, which compares a ground-truth labeling with a predicted ranking of the labels and counts the number of incorrectly ordered label pairs:

$$L_R(\mathcal{Y}_{\text{test}}, H) = \frac{1}{S}\sum_{i=1}^{S} \sum_{(j,j'): y_{ij} > y_{ij'}} \left(\frac{[\![h_j(\boldsymbol{x_i}) < h_{j'}(\boldsymbol{x_i})]\!]}{|\{(j,j') \mid y_{ij} > y_{ij'}\}|} \right), 1 \leq j, j' \leq m$$

1. $[\![\cdot]\!]$ is the indicator function.

Complementary to *instance-wise* losses, which are defined for (and averaged over) instances, losses are sometimes considered *label-wise*. An example is the macro-F1 measure:

$$\mathrm{F}(\mathcal{Y}_{\text{test}}, H) = \frac{1}{m}\sum_{j=1}^{m} \frac{2\sum_{i=1}^{S} y_{ij} h_j(\boldsymbol{x}_i)}{\sum_{i=1}^{S} y_{ij} + \sum_{i=1}^{S} h_j(\boldsymbol{x}_i)} \tag{1}$$

A linear combination of the four measures defined above is proposed by de Sá et al. (2017, 2018), who use it as an "objective function" in the respective AutoML tools:

$$\mathrm{L}_{\text{Fit}}(\mathcal{Y}_{\text{test}}, H) = \frac{1}{4}\Big(L_{0/1}(\mathcal{Y}_{\text{test}}, H) + L_H(\mathcal{Y}_{\text{test}}, H) + (1 - F(\mathcal{Y}_{\text{test}}, H)) + L_R(\mathcal{Y}_{\text{test}}, H)\Big).$$

As it combines different types of losses with different interpretations, this metric is debatable and difficult to interpret. Yet, in spite of our reservations, we will use it under the notion of "fitness loss" in our experimental study as well, mainly to reduce confounding factors and to ensure a fair comparison.

At first sight, MLC problems can be solved in a quite straightforward way, namely through decomposition into several binary classification problems: One binary classifier is trained for each label and used to predict whether, for a given query instance, this label is relevant or not. This approach is known as *binary relevance* (BR) learning. However, BR has been criticized for ignoring important information hidden in the label space, namely information about the interdependencies between the labels. Since the presence or absence of the different class labels has to be predicted *simultaneously*, it is arguably important to exploit any such dependencies. Correspondingly, a large repertoire of methods for MLC beyond BR has been proposed in the recent years. Most of these methods seek to improve predictive accuracy by exploiting label dependencies in one way or the other. We refer to Zhang and Zhou (2014) for an up-to-date survey on MLC algorithms.

3. A Multi-Label Extension of ML-Plan

As illustrated in Fig. 1 (left), multi-label classifiers may nest several classifiers in a recursive manner. Additionally, each of the classifiers has a set of parameters that need to be configured. While flattening these recursive structures to a single vector comprised of decision variables for the algorithm choices and a variable for each parameter that may occur for a specific layer, as done by Auto-WEKA and auto-sklearn, may work in principle, this approach would require many constraints to make sure that only relevant variables are considered. An arguably more natural way of representing these hierarchical dependencies is hierarchical task network (HTN) planning (Ghallab et al., 2004), or more specifically programmatic task network (PTN) planning (Mohr et al., 2018a) as incorporated in ML-Plan (Mohr et al., 2018b). Via HTN resp. PTN planning, the search space of possible algorithm choices and respective hyperparameters to be tuned is structured into *complex tasks* (blue), which are refined by *methods* to one or multiple complex tasks or *primitive tasks* (green). Intuitively, this formalism mimics a human expert who is tackling a (complex) problem by decomposing the original task into several sub-tasks until the resulting sub-tasks are (simple) primitive tasks.

Translated to AutoML for MLC, the initial task could be, for example, to do multi-label classification as shown on the right-hand side of Fig. 1. With the help of methods that are

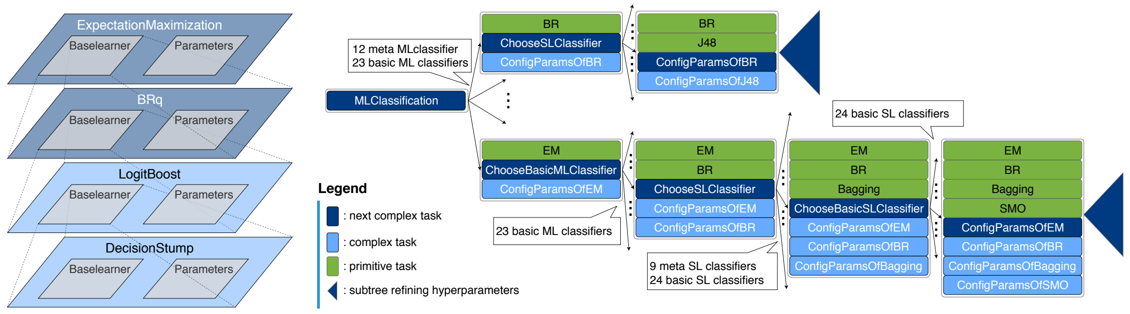

Figure 1: Visualization of the hierarchical structure of a machine learning pipeline (left) and an excerpt of the hierarchical planning search graph (right).

displayed in the form of arcs, this task can be refined by choosing a multi-label classifier. This can be either a meta multi-label classifier, e.g., Expectation Maximization (EM), or a basic one, e.g., Binary Relevance (BR). Depending on the decision, new decision-specific tasks arise, namely to choose base learners and to set the parameter values of the respective algorithm. Modeling the search space in this fashion yields a tree that can be used as a search graph for standard search algorithms. ML-Plan incorporates a Best-First search with random completions to complete partial specifications (decisions already made until a certain point) to fully specified classifiers that can be evaluated (using cross-validation or a holdout set). For more details on ML-Plan, we refer to (Mohr et al., 2018b) and (Wever et al., 2018a).

To derive ML^2-Plan from ML-Plan, we use the default configuration of ML-Plan and extend it in the following three ways:

- We extend the search space from SLC (WEKA) to MLC (MEKA+WEKA) but discarding preprocessors. This extension increases the size of the search space dramatically and yields in roughly 76,000 possible algorithm combinations (choice of main model and recursive base learner selections) to setup a multi-label classifier, compared to 234 as in the case of SLC. A breadth-first search to spawn all possible algorithm combinations, as it is done in ML-Plan, is thus unfeasible. Therefore, we adapt ML-Plan also to spawn only the first layer of algorithm choices, i.e., each multi-label classifier (basic and meta) is considered at least once as a main model. In contrast to (Wever et al., 2018a) we consider hyperparameter optimization as well. Hyperparameter optimization is done via a single decision for categorical and boolean parameters and by iteratively splitting the domain of numeric parameters into sub-intervals and refining those step-by-step until an interval size is reached that is considered atomic.

- Compared to evaluating single-label classifiers, the evaluations are much more expensive. Therefore, we introduced an early stopping criterion for the Monte Carlo cross-validation. After each iteration, we perform a significance test to check whether the currently considered candidate might be added to the pool of candidates for the selection phase, i.e., the difference of its performance and the best hitherto solution

Dataset	Auto-MEKA$_{\text{GGP}}$	GA-Auto-MLC	Random Search	ML^2-Plan
`arts1`	**0.5196±0.01** (1) ○	0.5509±0.01 (3)	0.5527±0.13 (4)	0.5508±0.04 (2)
`bibtex`	- (3)	- (3)	0.5822±0.02 (2) ●	**0.4910±0.05** (1)
`birds-fixed`	0.3259±0.02 (3) ●	0.3239±0.03 (2) ●	0.3277±0.04 (4) ●	**0.2962±0.03** (1)
`business1`	0.4711±0.00 (3) ●	- (4)	0.4034±0.13 (2)	**0.3452±0.01** (1)
`emotions`	0.3643±0.02 (4) ●	0.3356±0.02 (3) ●	**0.2858±0.02** (1)	0.2944±0.02 (2)
`enron-f`	0.4750±0.00 (3) ●	0.4793±0.01 (4) ●	0.4540±0.03 (2) ●	**0.3997±0.02** (1)
`flags`	0.4407±0.03 (3) ●	0.4427±0.03 (4) ●	0.3613±0.04 (2) ●	**0.3303±0.02** (1)
`genbase`	0.0701±0.02 (2)	0.1044±0.02 (3) ●	0.1511±0.15 (4) ●	**0.0684±0.01** (1)
`health1`	0.5153±0.00 (4) ●	**0.4499±0.01** (1)	0.5141±0.07 (3)	0.4564±0.03 (2)
`llog-f`	0.5054±0.01 (3) ●	0.4981±0.01 (2) ●	0.5183±0.02 (4) ●	**0.4792±0.02** (1)
`medical`	0.2770±0.05 (3)	0.2648±0.01 (2) ●	0.3167±0.10 (4) ●	**0.2425±0.01** (1)
`scene`	0.1998±0.02 (2) ●	0.2614±0.03 (4) ●	0.2003±0.04 (3) ●	**0.1746±0.01** (1)
`science1`	0.5495±0.00 (3)	**0.5318±0.00** (1) ○	0.6052±0.04 (4) ●	0.5462±0.02 (2)
`yeast`	0.4495±0.01 (3) ●	0.4773±0.01 (4) ●	0.3935±0.05 (2)	**0.3632±0.01** (1)
average-rank	2.86	2.86	2.93	1.29
#best	1	2	1	10
sig (i/t/d)	12 / 1 / 0	9 / 2 / 1	13 / 0 / 1	- / - / -

Table 1: L_{Fit} (mean ± standard deviation) for the test data of 20 runs per dataset. The rank per dataset of the respective approach is enclosed in parentheses.

is not more than 3%. If case the hypothesis test fails, we abort the evaluation of the classifier and return the mean of the evaluated iterations so far.

- We adapt the internal evaluations part of ML-Plan to the MLC setting, incorporating L_{Fit} as the objective function instead of error rate for SLC, and creating train and test splits at random instead of class-stratified splits.

4. Experimental Evaluation

We evaluate ML^2-Plan as introduced in the previous section on various datasets and compare it to three baselines: a random search, GA-AutoMLC (de Sá et al., 2017), and Auto-MEKA$_{\text{GGP}}$ (de Sá et al., 2018). The random search evaluates candidates that are picked uniformly at random from the set of 76,000 possible nested classifiers and chooses values of the resulting parameters at random as well. Note that while ML^2-Plan, random search and Auto-MEKA$_{\text{GGP}}$ operate on the same search space, GA-Auto-MLC is based on a much simpler space (see (de Sá et al., 2017)). All approaches underly the same timeout and resource limitations and use the implementations of the basic loss functions provided by MEKA. The implementation of ML^2-Plan is publicly available[2].

Results were obtained by carrying out 20 runs on 14 datasets with a timeout of 1 hour for each run and a timeout of 10 minutes for evaluating a single candidate. The datasets stem from the MULAN project website[3]. In each run, we used 70% of a randomized split of the data for learning (search) and 30% for testing. We used the *same* splits for all candidates, i.e., for each split, we ran ML^2-Plan as well as each baseline exactly once. The significance

2. Implementation of ML2-Plan: `https://github.com/fmohr/AILIbs`,
Dataset splits, seed project, and ReadMe: `https://github.com/mwever/ML2PlanAtAutoML2019`
3. `http://mulan.sourceforge.net/datasets-mlc.html`

of an improvement (marked by •) resp. degradation (∘) per dataset was determined using a Wilcoxon signed-rank test (Wilcoxon, 1945) with a threshold for the p-value of 0.05.

The experiments were run on up to 220 Linux machines in parallel, each of which with a resource limitation of 8 cores (Intel Xeon E5-2670, 2.6Ghz) and 32GB RAM. Runs that did not adhere to the time or resource limitations (plus a tolerance threshold of 10%) were canceled without considering their results for the respective approach.

A summary of the results is given in Table 1. While the upper part of the table describes the observed values for L_{fit} on the test data, the bottom part gives a summary regarding the average rank and statistics about number of times an approach has been the best solver. On the last row of the table, it is counted how many times ML^2-Plan achieved significantly improved, significantly degraded or equally performing results compared to a baseline.

The general impression is that ML^2-Plan performs clearly superior to the baselines and for the majority of datasets manages to return significantly better solutions compared to the competitor tools and the random search. Nevertheless, ML^2-Plan does not win in every case and it must admit defeat on arts against Auto-MEKA$_{\text{GGP}}$, on `emotions` against the random search, and on `health1` and `science1` against GA-Auto-MLC.

Furthermore, it was surprising to see that, according to the average rank statistic, Auto-MEKA$_{\text{GGP}}$ and GA-Auto-MLC perform only slightly better than the random search baseline. This might be due to the relatively small timeout of 1 hour we gave each tool for a single run but note that on the contrary ML^2-Plan already manages to outperform the random search. Moreover, Auto-MEKA$_{\text{GGP}}$ did not return any result for the dataset `bibtex` and GA-Auto-MLCfor the datasets `bibtex` and `business1` which may also be due to the low time budget. Nevertheless, we will consider larger timeouts in future work to investigate the long-term behavior of the different approaches as well.

5. Conclusion

In this paper, we presented an AutoML approach to multi-label classification. ML^2-Plan builds on ML-Plan and combines hierarchical task network planning with a global best-first search as proposed by Mohr et al. (2018b). Compared with previous AutoML tools for single-label classification, ML^2-Plan has to deal with more deeply nested structures to automatically select and configure multi-label classifiers for a given dataset. In an experimental study, we showed that ML^2-Plan outperforms the baselines, including the only existing approaches to AutoML for multi-label classification (de Sá et al., 2018, 2017). Future work will be dedicated to improving scalability, e.g., by moving to a service-oriented architecture (Mohr et al., 2018d,c) or incorporating meta-learning techniques for warmstarting (Feurer et al., 2015), and to improving efficiency during search, e.g., by biasing the random completion towards in general more promising solutions. Finally, the wide spectrum of loss functions in MLC motivates a multi-objective optimization process that seeks for trade-offs between different (and potentially conflicting) performance metrics.

Acknowledgement

This work was supported by the German Research Foundation (DFG) within the Collaborative Research Center “On-The-Fly Computing” (SFB 901).

References

Alex Guimarães Cardoso de Sá, Gisele L. Pappa, and Alex Alves Freitas. Towards a method for automatically selecting and configuring multi-label classification algorithms. In *Genetic and Evolutionary Computation Conference, Berlin, Germany, July 15-19, 2017, Companion Material Proceedings*, pages 1125–1132, 2017.

Alex Guimarães Cardoso de Sá, Alex Alves Freitas, and Gisele L. Pappa. Automated selection and configuration of multi-label classification algorithms with grammar-based genetic programming. In *PPSN (2)*, volume 11102 of *Lecture Notes in Computer Science*, pages 308–320. Springer, 2018.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pages 2962–2970, 2015.

Ilche Georgievski and Marco Aiello. HTN planning: Overview, comparison, and beyond. *Artif. Intell.*, 222:124–156, 2015.

Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning - theory and practice.* Elsevier, 2004. ISBN 978-1-55860-856-6.

Brent Komer, James Bergstra, and Chris Eliasmith. Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In *ICML workshop on AutoML*, 2014.

Felix Mohr, Theodor Lettmann, Eyke Hüllermeier, and Marcel Wever. Programmatic task network planning. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling.* AAAI, 2018a.

Felix Mohr, Marcel Wever, and Eyke Hüllermeier. Ml-plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107(8-10):1495–1515, 2018b.

Felix Mohr, Marcel Wever, and Eyke Hüllermeier. Automated machine learning service composition. *CoRR*, abs/1809.00486, 2018c.

Felix Mohr, Marcel Wever, Eyke Hüllermeier, and Amin Faez. (WIP) towards the automated composition of machine learning services. In *SCC*, pages 241–244. IEEE, 2018d.

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA*, pages 847–855, 2013.

Marcel Wever, Felix Mohr, and Eyke Hüllermeier. Automated multi-label classification based on ml-plan. *CoRR*, abs/1811.04060, 2018a. URL `http://arxiv.org/abs/1811.04060`.

Marcel Wever, Felix Mohr, and Eyke Hüllermeier. *ML-Plan for Unlimited-Length Machine Learning Pipelines.* 2018b.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6): 80–83, 1945.

Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE Trans. Knowl. Data Eng.*, 26(8):1819–1837, 2014.