7th ICML Workshop on Automated Machine Learning (2020)

Federated Meta-Learning: Democratizing Algorithm Selection Across Disciplines and Software Libraries

Mukesh Arambakam ARAMBAKM@TCD.IE
Trinity College Dublin, School of Computer Science and Statistics, Ireland

Joeran Beel JOERAN.BEEL@UNI-SIEGEN.DE
University of Siegen, Department of Computer Science, Germany & Trinity College Dublin, School of Computer Science and Statistics, ADAPT, Ireland

Abstract

"Federated Meta-Learning" (FML), a concept that allows everyone to benefit from the data that is generated through software libraries including machine learning and data science libraries. We have built FMLearn, an application developed using the client-server model, which allows the exchange of meta-data about machine learning models and data in itself for the purpose of meta-learned algorithm selection and configuration. FMLearn, scikit-learn's toy datasets along with other datasets from UCI Machine Learning repository were used and evaluated against various machine learning algorithms using GridSearchCV and Cross-Validation for which the execution time was measured. In the case of scikit-learn's breast cancer dataset, an execution time of approx. 94.24 mins was recorded by performing Grid Search for the best algorithm. Whereas, when FMLearn was used only 3sec was recorded to fetch and execute the best algorithm along with its model parameters. The use of FMLearn takes approx. 3 sec to identify the algorithm with the best performance for this dataset. Where as for a large dataset like the skin segmentation, an execution time of approx. 869.74 mins was recorded and when using FMLearn was only 3sec sec was recorded for the same. Overall, the use of this application allows the user to scale down the repetitive effort and time consumed in rewriting and executing code and correcting possible human errors.

1. Overview and Related Work

An ever-growing number of algorithms are used to solve machine learning and data science tasks, and the challenge of algorithm selection and configuration is subject to intensive research. (Bischl et al., 2016; Vanschoren et al., 2014; Calandra, 2020; Collins et al., 2018; Romero et al., 2013; Vartak et al., 2017). Meta-learning is one of the most promising techniques to warm starting the algorithm selection and configuration process - (Hutter et al., 2019). With meta-learning, a machine learning model is trained to predict how algorithms perform on a given task. The meta-learning model is built based on the past performance of algorithms on a large number of tasks or datasets, which are described through meta-features. For unseen tasks, the best performing algorithms can be predicted through the meta-learner (and subsequently be optimized e.g. with Bayesian Hyper-parameter Optimization).

A challenge in algorithm selection and configuration is the (non) availability of data in some disciplines to build the meta-learning model, which is due to the workflow of machine learning, data science or other projects. Typically, software libraries – be it machine learning libraries like Auto-sklearn (Feurer et al., 2015), Auto-Weka (Kotthoff et al., 2017) or ML-

Plan (Mohr et al., 2018), recommender system libraries like LibRec-Auto (Mansoury and Burke, 2019), are used in isolation, either locally or in the cloud. By "in isolation" we mean that the information regarding how algorithms perform on a particular dataset, is neither published nor shared. Consequently, computationally expensive algorithm selection and hyper parameter optimization is performed by each machine learning engineer over and over again.

Our goal is to facilitate the algorithm selection process by leveraging historic performance data, produced on various devices and by various machine learning libraries to improve the performance of algorithm selection and save time in finding the best algorithm and it's hyper-parameters for a task. To improve the algorithm selection and configuration process we propose what we call "Federated Meta-Learning". In addition, we present the first prototype of Federated Meta Learning named "FMLearn".

2. Federated Meta-Learning

"Federated Meta-Learning" (FML) has similarities with "federated machine learning", which has been recently introduced by Google: "Federated [Machine] Learning enables [devices] to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the cloud." (McMahan and Ramage, 2017). However, Federated Machine Learning focuses on learning one machine learning task across multiple devices, whereas, Federated Meta Learning focuses on learning algorithm performance for arbitrary tasks across devices. We envision federated meta learning as an ecosystem where the raw data is kept on the original devices and the meta data, algorithm names, and performance of the algorithm on the tasks would be stored on a central FML server (though a peer-to-peer architecture might also be possible).

Federated Meta-Learning is, to the best of our knowledge, novel. The term "Federated Meta Learning" has only been used once before by Chen et al, but in a different context. (Chen et al., 2018)

3. FMLearn

We introduce "FMLearn" as a simple proof of concept of Federated Meta-Learning. FMLearn allows everyone to benefit from the data that is generated through machine learning and data science libraries. FMLearn consists of a server[1] and a client, in our case a modified scikit-learn[2], but it could be any machine learning library.

The input to FMLearn's API via the package built in scikit-learn is the dataset, which is preprocessed in scikit-learn via the help of auto-sklearn using it's dataset's meta-feature detection functionality. The meta-features of the dataset thus obtained along with the hash of the dataset is the input to FMLearn's API, and the output is a recommendation for the potentially best performing algorithm(s) and it's hyper-parameters to solve that task (see Figure 1). This recommendation consists either of a list of the best algorithms or simply the best performing algorithm and their predicted performance values along with it's (model's)

1. `https://github.com/mukeshmk/fm-learn`
2. `https://github.com/mukeshmk/scikit-learn`

hyper-parameters. As of now, FMLearn is a knowledge base or directory of algorithm-data performance measures, a meta-learning algorithm is a work in progress and is in need of improvement. This knowledge base is built and updated by users who use FMLearn by sending meta-data of their model and dataset via scikit-learn to FMLearn.

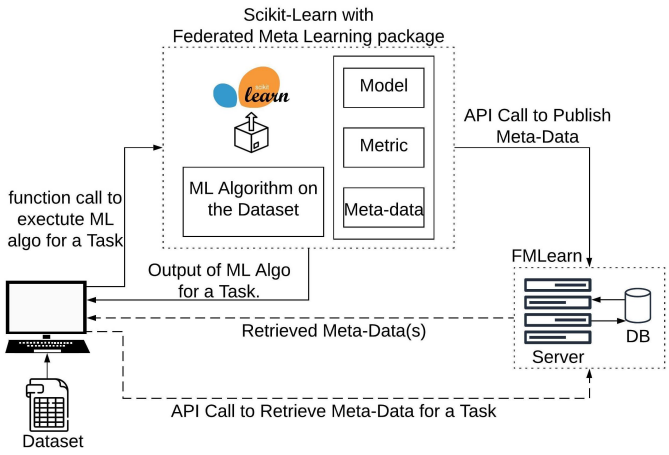

Figure 1: Architecture Diagram

The modified version of scikit-learn is used to make function calls to auto-sklearn (an external library which is now introduced as a dependency) to obtain the meta-features about the dataset, these meta-features include information like *ClassEntropy, SymbolsSum, ClassProbabilitySTD, InverseDatasetRatio, RatioNominalToNumerical, NumberOfCategoricalFeatures, NumberOfNumericFeatures, NumberOfFeaturesWithMissingValues, etc.* about 24-30 features depending on the dataset. These meta-features along with the hash of the dataset is to used to make an API call to FMLearn internally to publish the dataset and retrieve meta-data about the model. The API to publish the meta-data is:

```
POST: /metric
```

These API calls are used to retrieve all / the best algorithm with minimum metric value / the best algorithm with maximum metric value respectively from the FMLearn given a hashed-dataset as a data parameter along with it's meta-features:

```
POST:/metric/retrieve/{all/min/max}
```

These API's are exposed via scikit-learn library by importing the following package and by making appropriate function calls:

```
from sklearn.fml import FMLClient
```

A sample of the output from the API call via scikit-learn for one of it's toy data-sets (Diabetes data-set) see figure 2 (Tibshirani et al., 2004).

```
"Algorithm Name": "AdaBoostClassifier",
"Metric Name": "Accuracy",
"Metric Value": 0.970144387517466,
"Params": [
        "algorithm" : "SAMME",
        "learning_rate" : "0.95"
        "n_estimators" : "80"
]
```


Figure 2: Sample Output

4. Evaluation

We used FMLearn's client on two computers. The first computer trained eight machine learning algorithms[3] on five small datasets having about 500 instances: Breast Cancer (McMahan and Ramage, 2017), Diabetes (Tibshirani et al., 2004), Wine (lic, 1991), Boston (Harrison and Rubinfeld, 1978) and Iris (Fisher, 1950) and 5 large datasets having about 15,000 - 250,000 instances: Adult, MAGIC Gamma Telescope, Skin Segmentation (Bhatt et al., 2010), Statlog-Shuttle and Nursery (Dua and Graff, 2017).

The training used Grid-Search for hyper parameter optimization and cross-validation. The total execution time was between 13.67 minutes (Iris) and 94.24 minutes (Breast Cancer) for the small datasets (see Table: 1), and between 256.42 minutes (Nursery) and 869.74 minutes (Skin Segmentation) for the large dataset (see Table 2). FMLearn automatically submits all performance metrics and algorithm names along with the meta-features and hashes of the datasets to the FMLearn application via the API.

On a second machine, we run the same experiments. But, before the training started, FMLearn's client requested the algorithm recommendations via the API. In the scenario that the client just used the returned best algorithm with its hyper parameters, no training was needed. Hence, for a small dataset, the user saves an average of 48.79 minutes and about 92.24 minutes (for Breast-Cancer Dataset) in a best case scenario. Whereas, for a large dataset the user saves an average of 533.21 minutes and about 869.74 minutes (for Skin Segmentation Dataset) in a best case scenario. This amounts to about 86.72% and 95.762% (for small and large datasets respectively) of time spent waiting by the user, when the machine learning algorithm performs hyper-parameter tuning and selects the best parameters for the model. In a scenario where the user would want to re-optimize hyper parameters, re-training was required for only the best algorithm suggested by FMLearn. Under these circumstances, time saved on an average by the user was about 40.864 minutes for small datasets and 513.15 minutes for large datasets.

5. Limitations and Future Work

Currently as a prototype, FMLearn suggests algorithm(s), if the hashed dataset provided as the input is an exact match to that in the database. The meta-features which is obtained from auto-sklearn is being used to run meta-learning algorithms which is currently a work in progress. We hope to improve the algorithm and introduce it to our system which makes

3. https://github.com/mukeshmk/toy-datasets

Execution Time (in minutes) for Small Datasets				
Datasets	Optimize All Algorithms	Re-Optimise best algorithm	FMLearn	Saving in %
Breast-Cancer	94.24	18.79	0.05	80
Boston	47.36	6.96	0.05	85.01
Diabetes	62.17	10.37	0.04	83.25
Wine	26.54	3.25	0.04	87.6
Iris	13.67	0.29	0.02	97.73
Average	48.796	7.932	0.04	86.718

Table 1: Execution time when using GridSearch vs FMLearn for small datasets

Execution Time (in minutes) for Large Datasets				
Datasets	Optimize All Algorithms	Re-Optimise best algorithm	FMLearn	Saving in %
Adult	582.51	19.01	0.05	96.72
MAGIC Gamma Telescope	279.01	14.63	0.04	94.74
Nursery	256.42	15.47	0.04	93.95
Skin Segmentation	869.74	29.86	0.05	96.56
Statlog-Shuttle	678.37	21.35	0.04	96.84
Average	533.21	20.06	0.044	95.762

Table 2: Execution time when using GridSearch vs FMLearn for large datasets

it powerful enough to handle user request for algorithm and hyper-parameter recommendations of unseen datasets. FMLearn only stores values like algorithm name, metric name, value, the hash of dataset, meta-features of the dataset and the model-parameters. In future implementation, we hope to store meta data about the model. In the long run, social questions need consideration such as preventing manipulation (developers of algorithms may have an interest that their algorithms are 'recommended') and free-rider problems (users benefiting from the system without sharing their data). A potential challenge could be if the system focuses only on the globally best algorithm and there arises a need to focus on per-instance algorithm selection as well. Ultimately, FMLearn should be able to predict algorithm performance for unseen tasks.

References

Uci machine learning repository: Wine dataset, 1991. URL `https://archive.ics.uci.edu/ml/datasets/wine`.

Rajen Bhatt, Abhinav Dhall, Gaurav Sharma, and Santanu Chaudhury. Efficient skin region segmentation using low complexity fuzzy decision tree model. pages 1 – 4, 01 2010. doi: 10.1109/INDCON.2009.5409447.

Kerschke P. Kotthoff L. Lindauer M. Malitsky Y. Fréchette A. Hoos H. Hutter F. Leyton-Brown K. Tierney K. Bischl, B. et al. A benchmark library for algorithm selection. *Artificial Intelligence*, (237):41–58, 2016.

Roberto Calandra. Workshop on meta-learning (metalearn 2017), 2020. URL `http://metalearning.ml/2017/`.

Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning for recommendation. 02 2018.

Andrew Collins, Dominika Tkaczyk, and Joeran Beel. A novel approach to recommendation algorithm selection using meta-learning. In *AICS*, pages 210–219, 2018.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2755–2763, Cambridge, MA, USA, 2015. MIT Press.

Ronald Aylmer Fisher. *Contributions to mathematical statistics*. Wiley, 1950.

David Harrison and Daniel Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5:81–102, 03 1978. doi: 10.1016/0095-0696(78)90006-2.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning - Methods, Systems, Challenges*. 01 2019.

Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research*, 18(25):1–5, 2017. URL `http://jmlr.org/papers/v18/16-261.html`.

Masoud Mansoury and Robin Burke. Algorithm selection with librec-auto. 04 2019.

Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data, 2017. URL `https://ai.googleblog.com/2017/04/federated-learning-collaborative.html`.

Felix Mohr, Marcel Wever, and Eyke Hüllermeier. Ml-plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107, 07 2018. doi: 10.1007/s10994-018-5735-z.

Cristóbal Romero, Juan Luis Olmo Ortiz, and Sebastian Ventura. A meta-learning approach for recommending a subset of white-box classification algorithms for moodle datasets. 01 2013.

Robert Tibshirani, Iain Johnstone, Trevor Hastie, and Bradley Efron. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004. doi: 10.1214/009053604000000067.

Joaquin Vanschoren, Carlos Soares, Pavel Brazdil, and Lars Kotthoff. *Meta-Learning and Algorithm Selection*. 08 2014.

M Vartak, A Thiagarajan, C Miranda, J Bratman, and H Larochelle. A meta-learning perspective on cold-start recommendations for items. advances in neural information processing systems. page 6907–6917, 2017.

Appendix A. Code Availability

All the code which is used for this paper is made available on GitHub:

- The modified scikit-learn is available at:
  https://github.com/mukeshmk/scikit-learn
- The code for FMLearn application is available at:
  https://github.com/mukeshmk/fm-learn
- The code used for the evaluation of the data is available at:
  https://github.com/mukeshmk/toy-datasets

The FMLearn application has also been deployed on heroku at https://fmlearn.herokuapp.com/.
Note: this is just a API server, the website doesn't provide any functionality per-se.

Appendix B. Machine Details

The configuration and details of the machine used to measure these metrics are as follows:

- Laptop: Lenovo Legion Y540
- Processor: Intel i5 9300H
- RAM: 8GB
- GPU: Nvidia GTX 1650
- Hard Drive: 1TB HDD + 125GB SSD
- OS: Windows 10