Towards Interactive Curation & Automatic Tuning of ML Pipelines

Carsten Binnig[1,2] Benedetto Buratti[1] Yeounoh Chung[1] Cyrus Cousins[1] Dylan Ebert[1]
Tim Kraska[1,3] Zeyuan Shang[1] Isabella Tromba[3] Eli Upfal[1] Linnan Wang[1] Robert Zeleznik[1]
Emanuel Zgraggen[1]

[1] Brown University, USA [2] TU Darmstadt, Germany [3] Massachusetts Institute of Technology, USA

ABSTRACT

Democratizing Data Science requires a fundamental rethinking of the way data analytics and model discovery is done. Available tools for analyzing massive data sets and curating machine learning models are limited in a number of fundamental ways. First, existing tools require well-trained data scientists to select the appropriate techniques to build models and to evaluate their outcomes. Second, existing tools require heavy data preparation steps and are often too slow to give interactive feedback to domain experts in the model building process, severely limiting the possible interactions. Third, current tools do not provide adequate analysis of statistical risk factors in the model development. In this work, we present the first iteration of QuIC-M (pronounced quick-m), an interactive human-in-the-loop data exploration and model building suite. The goal is to enable domain experts to build the full machine learning pipelines an order of magnitude faster than machine learning experts while having model qualities comparable to expert solutions.

1 INTRODUCTION

Data is everywhere. Companies store customer and sales information, researchers collect data by running experiments and application or website developers store interaction logs. But all this data is useless without the means to analyze it. Extracting actionable insights from data has been left to highly trained individuals who have strong mathematics and computer science skills. They have the background to query databases to create insightful reports and visualizations, develop statistical models and implement scalable infrastructures to process large and complex data. For example, it is common practice for corporations to employ teams of data scientists that assist stakeholders in finding qualitative, data-driven insights to inform possible business decisions. Having such a high-entry bar to data analysis however presents several challenges. For one, it presents a bottleneck. While research is trying to understand and promote visualization and data literacy and educational institutions are ramping up their data science curricula there is still a shortage of skilled data scientists. And second, and more importantly, restricting data analysis to those with a computational background creates an inequality. Small business owners without programming skills or research domains where computational background might not be as prevalent are at a disadvantage as they can not capitalize on the power of data.

We believe that there is an opportunity for tool builders to create systems for people who are domain experts but neither mathematicians, statisticians or programmers. We introduce the first version of a system for Quality-aware Interactive Curation of Models, called QuIC-M (pronounced quick-m). Making sense of data is exploratory by nature, and demands rapid iterations and all but the simplest analysis tasks, require humans-in-the-loop to effectively steer the process. QuIC-M exposes this workflow through a novel pen-and-touch interface allowing domain experts to seamlessly interleave data exploration with curation of machine learning pipelines. Through QuIC-M domain experts can build these pipelines automatically from high level tasks specification and at a fast pace without the need to involve a data scientist and without sacrificing quality. Empowering novice users to directly analyze data also comes with drawbacks. It exposes them to “the pitfalls that scientists are trained to avoid” [2]. We discussed and described such “risk” factors and QuIC-M’s user interface in related works [1, 6]. In this paper we focus on QuIC-M’s architecture for automatic machine learning pipeline discovery.

2 OVERVIEW

To enable automatic discovery of data-driven insights, we propose a system which takes “problems” provided by domain experts as inputs and generates optimal machine learning pipelines as outputs. Over time, the system adopts and improves its performance by learning from past problems. This section contains definitions and an overview of our architecture and the following sections provide details about individual parts of our approach.

A problem consists of a dataset and a target attribute for prediction. Domain experts can specify such problems, and potential additional input such as user-defined features, through simple gestures in our pen-and-touch UI. Our system will then automatically attempt to find and present an optimal machine learning pipeline. Alternatively, users can “fix” certain parts of the pipeline, like preprocessing steps for example, and have the system fill in the blanks. This process is done progressively, where the system gradually optimizes over the space of possible pipelines, in order to display results to users at interactive speeds and allow users to provide feedback; e.g., by providing additional model constraints or visually changing decision planes based on domain knowledge. We denote a *pipeline* as an end-to-end solution to produce the predicted targets for a given problem. A pipeline consists of machine learning primitives (e.g., feature selection, pre-processing, model) and corresponding hyper-parameters. Given a specific problem, it is reasonable to firstly build the pipeline based on experiences, for example, for an image classification problem, we tend to use deep neural networks like AlexNet and ResNet. Based on this observation, we may generate candidate pipelines through rules collected from best practices, which are gathered by analyzing hand-crafted solutions for existing problems from various sources such as Kaggle competitions [5] or the OpenML website [1]. To efficiently organize these candidate pipelines, we define the *search space* as an interface to enumerate candidate pipelines. Note that these best practice rules are usually parametric, which means that they can be further fine-tuned with feedbacks from interactions. Therefore the search

[1]https://www.openml.org/

space can be evolved through time, from a general range of possible answers to more problem-specific.

Although we apply rules to construct the search space, the search space itself is still huge considering the complexity of machine learning problems, and it is impossible to run every pipeline in the search space without violating interactivity guarantees. To address this, we further propose the cost model to provide a rough estimate of "promisingness" of the candidate pipelines. With the cost model, we may just drag the top K promising pipelines out of the search space for validation. The cost model may be rough as first, but it can be learned over iterations and becomes more accurate.

We may apply different strategies to schedule computing resources on these pipelines to validate their performance on the given problem to save resources and guarantee low-latency (e.g., we may validate these pipelines on a sample of data and throw ones with bad performance). Finally, the back-end system returns the best pipeline back to the front-end for visualization and interaction, while the above-mentioned iteration will be running in the background, thus the answer pipeline will be improved continuously.

During the design of our system we particularly focused on the following aspects. (1) *Interactivity*: data analysis is an exploratory and iterative process, high-latencies are counterproductive and limit the rate at which domain experts can produce insights; (2) *Progressiveness*: instead of waiting minutes or hours on results, we want users to see early results as soon as possible. We consciously tradeoff some performance for interactivity by making all of our components output results in a progressive-fashion through incremental computation; (3) *Modularity*: there usually exist numerous implementations for each component in our system, by explicitly specifying the interfaces of each component, each implementation is interchangeable, thus making our system modular and highly configurable.

3 RULE-BASED SEARCH SPACE

Before applying each rule, we check the problem description and its corresponding dataset schema to make sure it fulfills the condition of the rule, and only apply rules applicable (e.g., we use random forest classifier only for classification problem). Users are also able to provide some hints (e.g., please use SVM first).

Based on the definition of *pipeline*, we know that a pipeline consists of two parts: (1) the structure of a pipeline, i.e., what primitives are in the pipeline (e.g., label encoder, min-max scaler, SVM, Random Forest, etc.) and how they are connected to form the pipeline; (2) the hyper-parameters of these primitives. Based on this observation, we have two kinds of rules: (1) *structure rule*, which specifies the primitives in the pipeline (e.g., for categorical features, we may use label encoding or one hot encoding); (2) *parameter rule*, which specifies how to generate hyper-parameters for each primitive (e.g., for linear SVM, if the performance with large regularization factor λ is bad, we may decrease it). These rules can be learned as well, for example, we could also adjust the parameters of some rules based on the performance of generated pipelines.

4 COST-BASED PIPELINE SELECTION

Cost models are important as a filtering for candidate pipelines, since an accurate cost model could greatly save the computing resources and decrease latency. For now, we use the history performance as the estimated cost, i.e., if the average performances (measured by the metrics, e.g., accuracy, F1 score) of pipelines generated by some rules are obviously better than others, we may select pipelines from these rules with higher priority.

In the future, we are going to improve the cost model by considering other factors, e.g., training time, estimated performance, rules used; and adjusting the parameters in the cost model using the results of pipeline runs (similar to back propagation in neural networks) to adaptively optimize the cost model. In other words, the cost models can be learned as well.

5 OPTIMIZATION ALGORITHMS

There are different strategies to validate these candidate pipelines on the given dataset. The most straightforward way is to use the brute force search, which trains all candidate pipelines on the train dataset, then evaluates their metrics, and returns the best one. However, since it is expensive to validate a pipeline, we could apply more sophisticated methods to find the optimal pipeline with less budget (e.g., CPU/IO resources, latency). However, optimization algorithms like Bayesian Optimization [3] and Hyperband [4] does not work well under this scenario because they only optimize hyper-parameters without considering selection of primitives, and they are designed for batch processing, which is not suitable for interactive analytics. Based on these observations, we propose the following adaptive algorithm via validation error.

First we split the dataset to get independent train (into several equal sized batches of sub-samples) and validation sets, and pick a fixed amount of pipelines from the search space. During an epoch, we train these pipelines on some batches of sub-samples, and at the end of each epoch, we apply a halting criteria on these pipelines by checking their training and validation performance (e.g., accuracy) to prune un-promising ones. We consider the training accuracy as an upper bound of the true accuracy and we prune pipelines whose training accuracy drops below the best validation accuracy seen thus far. After that, for those surviving pipelines, we start another new epoch with increased number of batches. We repeat these epochs until we use the whole training set, and we pick the best pipeline from the survivors as the final answer. Since one batch of sub-sample is usually small, we may return the best pipeline in the first epoch as the first answer and iteratively update it to achieve low latency. This method has statistical guarantees that a finite set of pipelines trained over a finite number of epochs have validation accuracies close to their true accuracies at the end of each epoch.

6 CONCLUSION

To demonstrate the advantages of QuIC-M, we implemented a prototype with all above mentioned techniques. We find that the current implementation of our system is able to generate solutions for simple classification and regression tasks which are on par with hand-crafted solutions with friendly UI and short latency. We plan to extend our prototype to support more problem types (e.g., text, image, etc.), include more rules (e.g., using deep neural networks), utilize more complex cost models, implement different optimization strategies. And we plan to extensively benchmark our system to understand more about the differences between our system and hand-crafted solutions and to run user studies to evaluate the user interface aspect of our system.

REFERENCES

[1] Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2015. Vizdom: interactive analytics through pen and touch. *Proceedings of the VLDB Endowment* 8, 12 (2015), 2024–2027.

[2] Danyel Fisher, Rob DeLine, Mary Czerwinski, and Steven Drucker. 2012. Interactions with big data analytics. *interactions* 19, 3 (2012), 50–59.

[3] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. 2016. Fast bayesian optimization of machine learning hyperparameters on large datasets. *arXiv preprint arXiv:1605.07079* (2016).

[4] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2016. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560* (2016).

[5] Mark Senn. 2017 (accessed December 29, 2017). *Kaggle Competitions*. https://www.kaggle.com/competitions

[6] Zheguang Zhao, Lorenzo De Stefani, Emanuel Zgraggen, Carsten Binnig, Eli Upfal, and Tim Kraska. 2017. Controlling false discoveries during interactive data exploration. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 527–540.