Fanglue: An Interactive System for Decision Rule Crafting

Chen Qian
Ant Group
Hangzhou, China
frank.qc@antgroup.com

Shiwei Liang
Ant Group
Hangzhou, China
shiwei.lsw@antgroup.com

Zhaoyang Wang
Ant Group
Beijing, China
mark.wzy@antgroup.com

Yin Lou
Ant Group
Sunnyvale, CA, USA
yin.lou@antgroup.com

ABSTRACT

In many applications the training data do not always contain sufficient information to produce high-quality decision rules for standard (end-to-end) rule mining algorithms, and human experts have to incorporate domain knowledge during rule induction in order to get meaningful results. In this work we present Fanglue, a home-grown system inside Alipay, for interactive decision rule crafting. Fanglue is a distributed in-memory system and is highly responsive when processing large-scale datasets. In addition, Fanglue extends the standard representation of a decision rule by introducing disjunctive clauses. Having disjunctive clauses can improve the coverage and robustness of a decision rule, especially for fraud prevention in Fintech applications.

PVLDB Reference Format:
Chen Qian, Shiwei Liang, Zhaoyang Wang, and Yin Lou. Fanglue: An Interactive System for Decision Rule Crafting. PVLDB, 16(12): 4062 - 4065, 2023.
doi:10.14778/3611540.3611621

1 INTRODUCTION

Decision rules are widely used in mission-critical tasks such as fraud prevention in Fintech applications since rules are highly interpretable thanks to their simple if-then structure. Standard decision rule has two parts; a conjunction of conditions and a prediction. A condition is of the form (feature, operator, value), e.g., `age` > 50. A rule makes certain prediction when all conditions are satisfied for a given input. In this case, we say that the rule covers this input.

Most existing rule mining systems (such as CN2 [3] or RIPPER [4]) are *end-to-end*; given a training set, the human expert specifies the optimization metric and hyperparameters for the rule mining algorithm, and then she has to wait for the algorithm to terminate in order to get a set of rules. In other words, hyperparameters and the optimization metric are the only "knobs" for human expert to incorporate domain knowledge to some extent, and there is nothing else that the human expert can do to guide the algorithm how to grow a rule once the rule mining process starts.

Unfortunately, in many applications the training data do not always contain sufficient information to produce high-quality decision rules for standard (end-to-end) rule mining algorithms, and human experts have to incorporate domain knowledge during rule induction to a deeper extent in order to get meaningful results.

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:10.14778/3611540.3611621

For example, assume a human expert at Alipay,[1] the largest third-party online payment platform in China, is authoring rules to cover a new kind of fraudulent activity that is only identified recently. She has prepared a dataset for the learning algorithm but it has only a few examples of such activity. Suppose a key step of this fraud is to ask the victim to send the fraudster multiple payment QR codes, and therefore the count of payment QR code refreshment within a short time should be an important identifier of such fraudulent activity. However, she finds that none of the rules returned by the mining algorithm uses this feature, and most rules employ `transaction amount` to distinguish the fraud from normal activities, as `transaction amount` in the dataset "coincidentally" separates these two kinds of activities. This is not uncommon in practice for applications such as fraud prevention, where there are usually very few positive points. Therefore, there is a high chance that an irrelevant feature can perfectly separate positive and negative points based on the input data. Although the count of payment QR code refreshment is indeed a very competitive feature during the rule induction process (e.g., often ranked high in the candidate evaluation), it simply never gets the chance to win over other features due to the noise in the dataset.

In this paper we present Fanglue, a home-grown system inside Alipay, for interactive decision rule crafting that helps human experts better incorporate their domain knowledge in the face of data scarcity. Fanglue provides a web interface for users to visualize and craft decision rules, and it has been serving risk control experts for fraud prevention rule authoring for more than 2 years. Users upload the data into Fanglue to start data-driven rule authoring, and Fanglue generates real-time condition suggestion and data analytics to provide helpful quantitative information. To be highly responsive when processing large datasets, Fanglue distributes the data in memory and employs Ray [8] as its computation engine.

In addition, unlike standard rule mining algorithm that produces conjunction-only rules, Fanglue employs a conjunctive normal form (CNF) type of rule representation that allows disjunctive clauses in a decision rule. Introducing disjunctive clauses in a decision rule can increase its coverage while keeping the rule concise. Producing a succinct and concise rule set is usually preferred by human when authoring rules. We also introduce a special disjunctive condition in Fanglue, called similar condition, to increase the robustness of a rule while ensuring the change in coverage as small as possible. For example, when the human expert decides to pick some vulnerable condition during the rule authoring (such as `transaction amount` > 500), she might want to add another layer of protection by finding "semantically similar" conditions that behave just like `transaction amount` > 500 (although being less powerful), so that when the

[1] https://www.alipay.com/

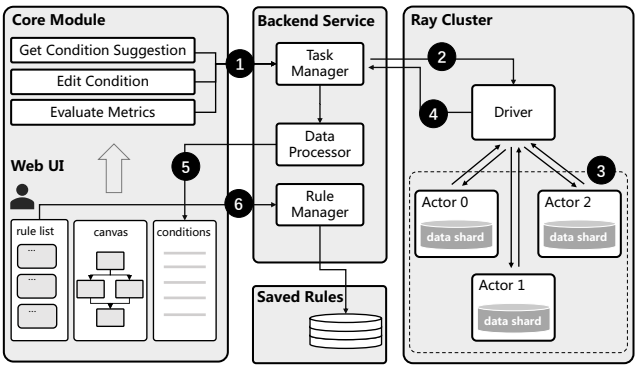

Figure 1: System overview.

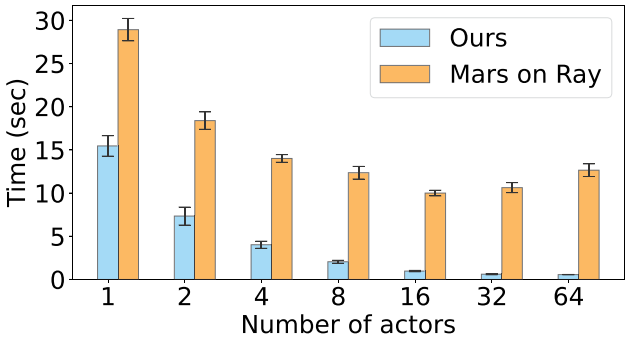

Figure 2: Running time comparison on evaluation of candidates for the next condition suggestion.

fraudster discovers the 500 threshold on transaction amount, this rule may still keep functioning for fraud prevention.

To enable multiple rule crafting, by default Fanglue automatically excludes the data covered by previous rules so that the next rule authoring can focus on covering the remaining data. Users can also choose to undo some exclusion for further experimentation.

Some commercial solutions, such as Sparkling Logic's SMARTS decision manager [2], also provide an interactive environment for rule authoring. Fanglue differs from those commercial solutions in that a) Fanglue can process large datasets with low latency and b) Fanglue extends the representation of a decision rule by introducing disjunctive clauses. There are also interactive rule refinement systems, such as Rudolf [6], that focus on post processing of decision rules while Fanglue provides an interactive environment during the whole process of rule crafting.

2 SYSTEM OVERVIEW

Figure 1 shows an overview of Fanglue. Users interact with Fanglue through a Web interface. There are three core modules in Fanglue; `Get Condition Suggestion`, `Edit Condition` (including manual editing or applying a suggested condition), and `Evaluate Metrics` of the current rule. The task manager is responsible for handling requests from different core modules (1), and it will launch the corresponding Ray job (2).

The data are distributed across a set of Ray actors and are persistent in memory. For a specific computation task, each actor builds its own local statistic. Those local statistics are then aggregated onto the driver to obtain the global statistic (3). The driver returns results to the task manager (4), which are then passed to data processor. The data processor proposes some operations based on the global statistic (e.g., evaluation results of a list of candidate conditions) (5), and the system waits for the user's response. Once the user makes some action (e.g., user picks one condition suggestion), the action triggers corresponding core module (e.g., add a particular condition) and the above process repeats. Authored rules are saved into a database (6).

3 TECHNICAL BACKGROUND

Fanglue employs a conjunctive normal form (CNF) type of rule representation. A CNF type of decision rule is a conjunction of one or more clauses and a prediction, where a clause is a disjunction of conditions. A condition is in the form of (feature, operator, value). In this work we focus on binary classification problems where labels are either positive or negative. We assume a training set and a validation set are given. Metrics such as precision, recall, F_1 score, or coverage on positive labels[2] can be used to evaluate a decision rule. Fanglue provides three types of real-time suggestion in a data-driven way to assist users in their decision rule authoring; "AND" condition suggestion, "OR" condition suggestion, and similar condition suggestion.

3.1 "AND"/"OR" Condition Suggestion

Assume we already have some clauses in a decision rule, to provide suggestion on "AND" (or "OR") condition, we search over all possible (feature, operator, value) triples and evaluate them by appending the candidate condition to the current decision rule. This is the same procedure in many rule induction algorithms such as OneR [5]. Standard rule induction algorithms would choose the candidate condition with the best metric score, while Fanglue displays a shortlist of those candidate conditions with metric scores on validation set for users to choose from.

To facilitate fast evaluation of (feature, operator, value) triples, each Ray actor builds the histogram on its local partition of the data, and all local histograms are reduced onto the driver to get the global histogram. Once the global histogram is computed, the evaluation of a condition candidate triplet can be efficiently computed.

To evaluate the efficiency of our system design, we perform an experiment on a proprietary dataset with 1.4 million points and 50 features. Each feature is discretized into 32 equi-frequency bins and we consider operators in $\{\geq, >, \leq, <\}$. Figure 2 illustrates the running time comparison (aggregated over 5 runs) of generating evaluation results of all candidates of the very first condition, using our implementation and an optimized Mars on Rays [1] implementation (using data frame operators). We can see Fanglue's

[2]Coverage on positive labels measures the count of covered positive examples while recall is the ratio of this count over the total number of positive examples.

implementation is very efficient on this large dataset and can be highly responsive in an interactive environment. We also notice that the efficiency cannot be improved with more actors for data frame-based implementation, due to the overhead introduced by the framework.

3.2 Similar Condition Suggestion

As we mentioned in Section 1, a semantically similar condition can increase the robustness of a decision rule. Similar condition is a special type of "OR" condition with additional constraints. Assume the current rule is $C_1 \wedge C_2 \wedge C_3$, where C_i's are disjunctive clauses, and suppose we want to add a similar condition to C_2. A "semantically similar" condition should have the similar coverage of C_2 on positive data, without introducing too many additional (negative) points in the meantime. We use $\mathcal{A}$ to denote the subset of data covered by the current rule $C_1 \wedge C_2 \wedge C_3$. We search over all possible (feature, operator, value) triples under $C_1 \wedge C_3$ (leave alone C_2). Each candidate condition will cover a subset of data, denoted as $\mathcal{B}$ and an ideal similar condition should have $\mathcal{A} = \mathcal{B}$.

In practice, however, we found such ideal similar condition is very rare. Therefore, for each candidate similar condition, we calculate the Jaccard similarity of the positive points as,

$$\texttt{PosJaccard}(\mathcal{A}, \mathcal{B}) = \frac{|\mathcal{A}^{pos} \cap \mathcal{B}^{pos}|}{|\mathcal{A}^{pos} \cup \mathcal{B}^{pos}|}, \tag{1}$$

where $\mathcal{R}^{pos}$ denotes a subset of $\mathcal{R}$ on positive labels. To avoid introducing too many negative points, we calculate the similarity of these two sets on negative points for all eligible candidates as

$$\texttt{NegRatio}(\mathcal{A}, \mathcal{B}) = \frac{|\mathcal{A}^{neg}|}{|\mathcal{A}^{neg} \cup \mathcal{B}^{neg}|}, \tag{2}$$

where $\mathcal{R}^{neg}$ denotes a subset of $\mathcal{R}$ on negative labels.

We define the "overall similarity" as the harmonic mean of `PosJaccard` and `NegRatio` as,

$$\texttt{OverallSim}(\mathcal{A}, \mathcal{B}) = \frac{2\texttt{PosJaccard}(\mathcal{A}, \mathcal{B}) \times \texttt{NegRatio}(\mathcal{A}, \mathcal{B})}{\texttt{PosJaccard}(\mathcal{A}, \mathcal{B}) + \texttt{NegRatio}(\mathcal{A}, \mathcal{B})}. \tag{3}$$

We filter out candidates with `PosJaccard` < 0.8 and then sort all eligible similar condition candidates according to their `OverallSim` in decreasing order so that users can choose a proper one based on domain knowledge and our suggestion.

For each condition candidate (feature, operator, value), $|\mathcal{A}^{pos} \cap \mathcal{B}^{pos}|$ and $|\mathcal{A}^{pos} \cup \mathcal{B}^{pos}|$ can be collected locally within each Ray actor, and `PosJaccard` and `OverallSim` can be computed on the driver once global statistics be aggregated.

3.3 Multiple Rule Authoring

When a single rule is authored and saved, its effects on the data should be excluded (by removing data covered by this rule), so that the next rule authoring can focus on covering the remaining data points instead of wasting the effort on those already covered points. This is typically known as the sequential covering algorithm [7], as illustrated in Figure 3.

Fanglue supports an extended version of sequential covering; users can choose whether to include or exclude the effects of particular rules. For example, users may find some equally good conditions when authoring one rule. Instead of making the choice right away,

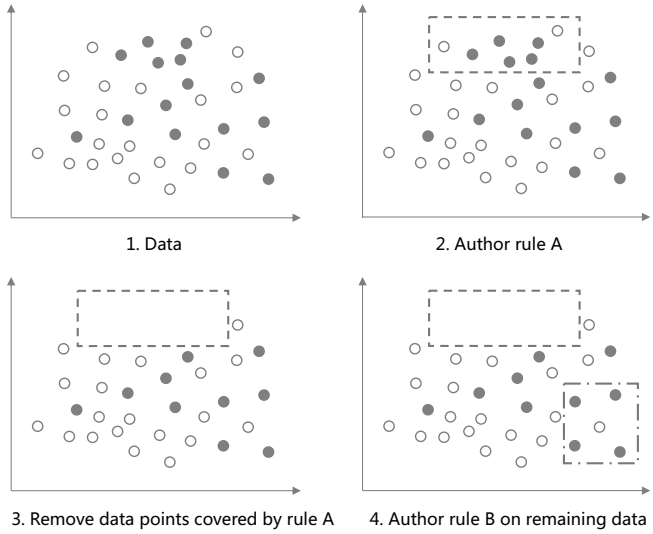

Figure 3: Illustration of the sequential covering algorithm.

Fanglue users can choose one of them to finish the current rule authoring, and then undo the exclusion of the newly crafted rule. By doing so, users can go back to previous states to explore other options. This gives users the freedom to go back and forth when authoring multiple decision rules.

4 DEMONSTRATION SCENARIOS

We will use the "Default" dataset from the UCI machine learning repository[3] to demonstrate Fanglue's core functionalities. The dataset contains 30,000 points and the task is to predict whether a credit card default will happen for the next month. The dataset will be pre-loaded into the system, and the audience will be able to experience interactive rule crafting in Fanglue through the following scenarios.

Finding Best "AND"/"OR" Conditions. We will ask the user to add the next best "AND"/"OR" condition, with the help of Fanglue's real-time condition suggestion. The user will be able to play with different conditions to see their effects on the target metric, and trade off metric improvement for interpretability. Figure 4 showcases a situation where there are multiple "AND" condition candidates with similar effects on the target metric but the interpretability of those candidates differs.

Adding Similar Conditions. In this scenario, the user will be asked to add similar condition to an existing rule to improve its robustness. We have prepared a rule that there are multiple places to add similar condition. Fanglue computes all eligible similar conditions and presents them to the user along with metric values of `OverallSim` and `PosJaccard`. The user will experience the tradeoff among rule robustness, metric changes and interpretability. Figure 5 illustrates one possibility of adding a similar condition to an existing rule.

Authoring Multiple Rules. Once a single rule is authored and saved, Fanglue by default will exclude its effect on the current

[3] https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

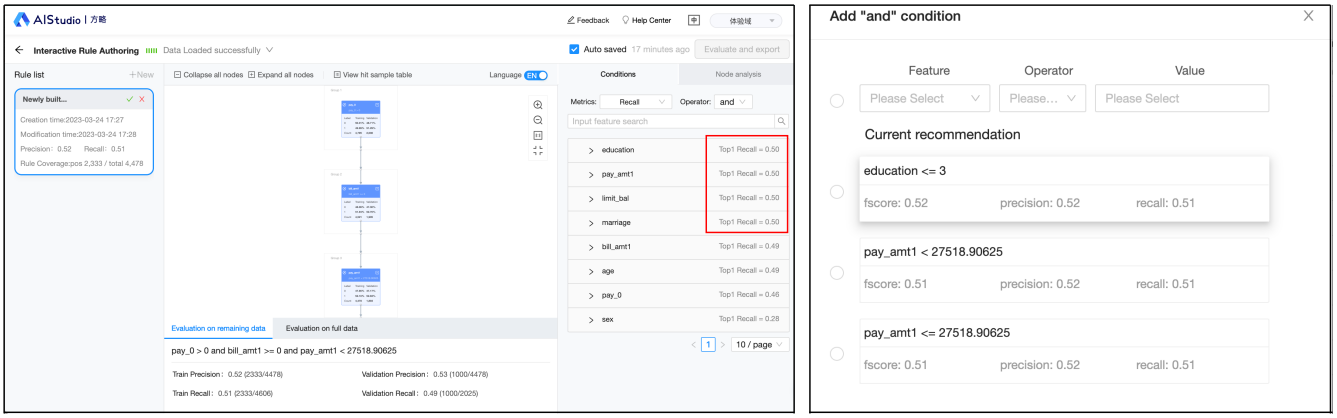

Figure 4: The left panel displays "AND" condition suggestions on the right of the canvas given current rule. Notice that there are multiple candidates on different features with similar metric value (0.5 recall in this case) for the next "AND" condition. Users will employ their domain knowledge to apply one of those candidates. The right panel shows the pop-up window for user manual adding an "AND" condition. Notice also the metrics for top 3 condition candidates are very close.

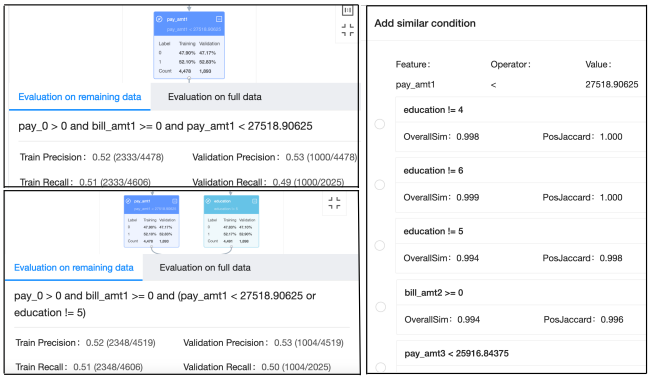

Figure 5: On the left we show the screenshots before and after adding a similar condition (in light cyan) for a selected row in current rule. The right panel shows the suggested similar conditions along with metric values of `OverallSim` and `PosJaccard` for users to make a sensible choice. We see that adding similar condition does not change the metrics too much on current rule.

dataset (by removing data covered by this rule) to prevent duplicate effort for subsequent rule crafting from covering points that are already covered. Users will also have the choice of not excluding a particular rule. In this scenario, the user will experience multiple rule authoring in Fanglue and in particular experience the different consequences of excluding and not excluding the effect of a particular rule. Figure 6 illustrates the interface of setting rule exclusion in Fanglue. Users can choose to undo the exclusion of rule 4, so that the next rule will have the exact same context with the context right before authoring rule 4. This usually happens when there were some equivalently good choices during rule 4's authoring, and users may want to go back to the previous states and try other

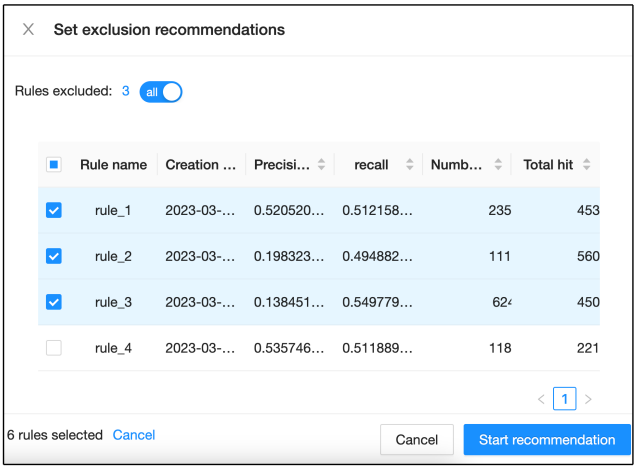

Figure 6: Fanglue's interface for setting rule exclusion.

choices. Once the new rule is crafted, it can be compared with rule 4 for further investigation.

REFERENCES

[1] 2023. *Mars on ray.* https://docs.ray.io/en/latest/ray-more-libs/mars-on-ray.html
[2] 2023. *Sparkling Logic's SMARTS decision manager.* https://www.sparklinglogic.com/
[3] P. Clark and T. Niblett. 1989. The CN2 induction algorithm. *Machine learning* 3, 4 (1989), 261–283.
[4] W.W. Cohen. 1995. Fast effective rule induction. In *ICML*.
[5] R.C. Holte. 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine learning* 11, 1 (1993), 63–90.
[6] T. Milo, S. Novgorodov, and W. Tan. 2016. Rudolf: interactive rule refinement system for fraud detection. *PVLDB* 9, 13 (2016), 1465–1468.
[7] C. Molnar. 2020. *Interpretable Machine Learning.* Lulu. com.
[8] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M.I. Jordan, and I. Stoica. 2018. Ray: A distributed framework for emerging AI applications. In *OSDI*.