What Changed? Interpretable Model Comparison

Rahul Nair[1] , **Massimiliano Mattetti**[1] , **Elizabeth Daly**[1]
Dennis Wei[2] , **Öznur Alkan**[1] , **Yunfeng Zhang**[2]
[1]IBM Research Europe
[2]IBM Research Yorktown
{rahul.nair@ie., massimiliano.mattetti@, elizabeth.daly@ie.}ibm.com,
{dwei@us.,OAlkan2@ie., zhangyun@us.}ibm.com

Abstract

We consider the problem of distinguishing two machine learning (ML) models built for the same task in a human-interpretable way. As models can fail or succeed in different ways, classical accuracy metrics may mask crucial qualitative differences. This problem arises in a few contexts. In business applications with periodically retrained models, an updated model may deviate from its predecessor for some segments without a change in overall accuracy. In automated ML systems, where several ML pipelines are generated, the top pipelines have comparable accuracy but may have more subtle differences. We present a method for interpretable comparison of binary classification models by approximating them with Boolean decision rules. We introduce stabilization conditions that allow for the two rule sets to be more directly comparable. A method is proposed to compare two rule sets based on their statistical and semantic similarity by solving assignment problems and highlighting changes. An empirical evaluation on several benchmark datasets illustrates the insights that may be obtained and shows that artificially induced changes can be reliably recovered by our method.

1 Introduction

There is growing recognition of the multi-faceted nature of trust in machine learning (ML) systems. Considerations such as bias, fairness, explainability, adversarial robustness, and out-of-distribution generalization need to be better understood *before* deployment. Accuracy is therefore not the sole determinant of model performance.

Within this context, we consider the problem of human-interpretable comparison of two ML models built for the same task. This problem has been less studied compared to explanation of a single ML model. As models can fail or succeed in different ways, classical accuracy metrics and other summary statistics may mask qualitative differences that are crucial to understand.

An important application of model comparison occurs in the overall ML lifecycle when models are updated. A great deal of focus has been placed on understanding and analysing an ML model when it is being initially developed, and on explaining predictions to end users. In deployment however, it is common to re-train the model with more up-to-date data points and ensure that accuracy holds. Considering accuracy alone provides no insight into what has changed. Methods to provide interpretable summaries of changes before redeployment are therefore vital in building understanding and trust.

A similar problem arises in a model selection context. In automated ML (autoML) pipeline generation procedures, for example, top performing pipelines may be hard to distinguish on accuracy alone. AutoML systems tend to have a trust deficit with users [Wang *et al.*, 2019] and previous studies have found [Drozdal *et al.*, 2020] that including transparency features helps to reduce the gap. In this setting, interpretable model comparisons can serve to highlight model differences and aid with model selection.

Given two ML models, one problem is to identify whether there is a (statistically) significant change [Bu *et al.*, 2019; Geng *et al.*, 2019; Harel *et al.*, 2014]. While our proposed method does detect changes, our focus is on the next step of characterizing *what* has changed between the models. In particular, our aim is to support model developers and end-users in understanding model differences, beyond summary statistics.

Research has shown that ML model developers as well as end-users develop a mental model of AI solutions [Kulesza *et al.*, 2012]. In a real-world example on predicting hospital readmissions, [Bansal *et al.*, 2019] documents how, over time, end-users trusted the system for elderly patients where the system was very accurate. A model update that improved overall accuracy also introduced errors for elderly patients. This change could lead to poorer decisions overall. The example highlights the importance of preserving the mental model that end users have formed about the decision making process.

The core idea pursued in this paper is as follows. An ML model is approximated by a directly interpretable model. This approximation, also referred to as a surrogate model, provides a compact rule set to describe model behaviour globally. We focus herein on binary classification tasks related to tabular data, due in part to the choice of rule sets as the surrogate model. Updates to the model are also approximated similarly, however with stabilization conditions called grounding.

Grounding seeks to preserve rule structure such that the rule sets from the two model versions are more directly comparable and also aids in persisting the end user mental models of an ML system.

The final step is to compare the two rule sets and highlight changes. We develop a method for this that solves an assignment problem to map rules from one set to those in the other set. We propose two ways of measuring similarity between rules: semantic, based on interpreting the conditions in the rules, and statistical, based on the data samples covered by the rules. To our knowledge, quantifying similarity between rule sets has been little studied and may be of independent interest.

The main contributions of the paper are therefore (1) drawing attention to the problem of interpretable model comparison, and providing a solution, (2) a mechanism to stabilize rule generation via grounding, and (3) a rule set comparison algorithm to highlight changes.

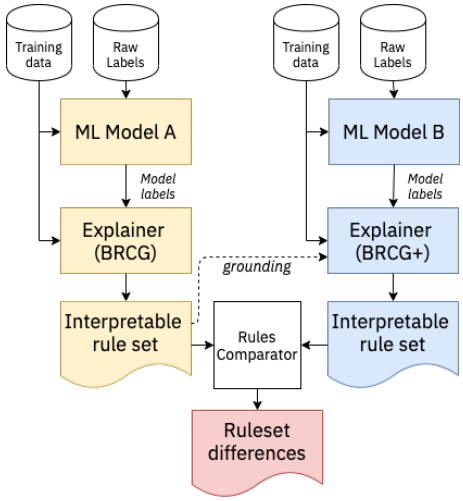

Figure 1: Method overview

2 Related Work

There is considerable literature on model change in general that studies changes in data distributions [Tsymbal, 2004] either in the features or target variables. *Concept drift* detection seeks to detect changes in the relationship between features and targets over time [Gama *et al.*, 2014]. A variety of methods exist, e.g. using empirical loss in data streams [Harel *et al.*, 2014], empirical difference tests between two sets of model parameters [Bu *et al.*, 2019], or specific methods for linear regression models [Geng *et al.*, 2019].

In contrast to the above works that aim to detect *whether* a change has occurred, the focus of this paper is in understanding what has changed. In this, our work is closest to Demšar and Bosnić [2018] where model explanations are used to study concept drift. Their method determines feature contributions over time and observes the changes in contributions. In this work we leverage rule set explanations which can readily identify feature *interactions*.

From the explainability side, the approximation of a supervised ML model by a simpler, more interpretable model likewise has a considerable history and many variations. In more recent literature, it has been called model distillation, extraction, compression, or global post hoc explanation.

Distillation/compression methods [Hinton *et al.*, 2015; Lopez-Paz *et al.*, 2016; Bucilă *et al.*, 2006; Ba and Caruana, 2014] have mainly considered simpler neural networks, which are generally still hard to interpret. Closer are methods that extract decision trees [Bastani *et al.*, 2017; Frosst and Hinton, 2017; Craven and Shavlik, 1995], which are related to rule sets. Domingos [1997] uses *ordered* rule sets (i.e. rule *lists*) and restricts the original model to be an ensemble of rule lists (we do not have such a restriction). The "distill-and-compare" approach of [Tan *et al.*, 2018] is most similar conceptually to our proposal but uses generalized additive models (GAMs) and fits one GAM to a black-box model and a second GAM to ground truth outcomes, not a second model as in our case. Sanchez *et al.* [2015] investigate logic rules and Bayesian networks to provide descriptive representations of matrix factorization models.

In terms of rule set learning algorithms, we have chosen that of [Dash *et al.*, 2018] (and its computationally simpler variant [Arya *et al.*, 2019]) because it explicitly optimizes the trade-off between accuracy (fidelity to the original model in our case) and rule set complexity and does so in a way that facilitates our addition of grounding. It is moreover representative of the state of the art. Recent alternatives that also control rule set complexity include [Lakkaraju *et al.*, 2016; Wang *et al.*, 2017], in contrast to older algorithms such as RIPPER [Cohen, 1995] that do not minimize complexity.

3 Methods

We are given two ML models, A and B, trained on two sets of training data and ground truth ("raw") labels. As shown in Figure 1, for each ML model, we learn an interpretable model as a surrogate using labels from the ML model (as opposed to the ground truth) and the corresponding training data. We leverage the method of Boolean Rules via Column Generation (BRCG), first presented in [Dash *et al.*, 2018] and later simplified and open-sourced [Arya *et al.*, 2019]. The algorithm generates a Boolean rule of the form `if (condition) then True else False`, where `condition` is in disjunctive normal form (DNF, OR of ANDs). Such a DNF rule constitutes an (unordered) rule set for the positive class. Following common terminology, we interchangeably refer to the conjunctive clauses within the DNF as "rules" or "conjunctions" within the rule set. It is equally possible to learn a DNF rule/rule set for the negative class, which is equivalent to a conjunctive normal form (CNF) rule for the positive class [Su *et al.*, 2016]. Rule sets generated for the two models being compared are processed by a rules comparator algorithm to determine differences. This algorithm produces an assignment of rules in one set to similar rules in the other set, or to none if the second set has fewer rules. Differences between corresponding rules can then be highlighted.

One challenge in such comparison in practice is that minor changes in training data, for example due to sampling vari-

ability, can yield rules that have very different structure. Even for the same training data, one can have several clauses that use different variables but have similar fidelity due to strong correlations between the variables. We devise a mechanism, termed grounding, to stabilize the structure of the rule sets. To do this, we modify the BRCG formulation by introducing a penalty term in the objective function to make previously generated rules more likely to persist in the new rule set. Toward this end, we first summarize the BRCG method and then present the grounding and rule set comparison elements novel to this paper.

3.1 Boolean Rules Via Column Generation (BRCG)

[Dash *et al.*, 2018] formulate the search for an accurate (i.e. faithful) and interpretable rule set as a mixed-integer linear programming (MILP) problem based on training samples X.

To describe the formulation, let P denote the set of positive samples, in our case the observations classified as $\hat{y}_i = 1$ by the ML model, and Z denote negative samples. All features in X are assumed to be binarized as in a decision tree, by thresholding with multiple thresholds for continuous features, and by the usual one-hot encoding for categorical features. Let K be the set of (exponentially many) possible clauses, namely conjunctions of the binary features in X. Define $K_i, K_i \subseteq K$ as the subset of clauses satisfied by observation i.

The main decision variables are w_k for all k in set K — a binary variable indicating whether conjunction k is selected for the model. Each conjunction k in K has an associated complexity c_k. We take c_k to be the degree of the conjunction, i.e. the number of participating literals. The formulation also defines ξ_i for $i \in P$ (i.e. for all positive samples) to indicate incorrect classification, i.e. a false negative.

The objective looks to minimize total Hamming loss, where the Hamming loss for each sample is the number of conjunctions that must be added or removed to classify it correctly. Specifically, this can be written as

$$\min_{\xi,w} \quad \underbrace{\sum_{i\in P} \xi_i}_{\text{false negatives}} + \underbrace{\sum_{i\in Z} \sum_{k\in K_i} w_k}_{\text{false positives}} \tag{1}$$

False positives add more than 'one unit' if they satisfy multiple selected conjunctions, all of which must be removed. The objective is subject to constraints:

$$\xi_i + \sum_{k\in K_i} w_k \geq 1 \quad \xi_i \geq 0, \qquad \forall i \in P \tag{2}$$

$$\sum_{k\in K} c_k w_k \leq C \tag{3}$$

$$w_k \in \{0,1\} \qquad \forall k \in K \tag{4}$$

Constraint (2) identifies false negatives. It states, for each positive sample, we either have a false negative ($\xi_i = 1$) or include a rule that correctly represents this observation (i.e. a conjunction from the set K_i). Constraint (3) bounds the total complexity of the selected rule set by a parameter C. Constraint (4) restricts the decision variables w_k to be binary.

Problem (1)–(4) is intractable as written, even with capable MILP solvers, because the set K is very large and it is prohibitive to generate the entire set of conjunctions. In any case, only a few w_k tend to be selected in the final solution. The authors [Dash *et al.*, 2018] use a column generation (CG) procedure, which is an iterative algorithm by which candidate conjunctions are generated at each iteration only if they can improve the overall objective. The method to generate a new candidate is called the pricing problem and the original model (1)–(4) is called the master problem. The CG procedure can be summarized by the following steps:

1. Restrict the master problem to a small subset of conjunctions $J \subset K$ and solve its linear programming (LP) relaxation, obtained by relaxing the constraint $w_k \in \{0,1\}$ to $w_k \geq 0$.
2. Solve the pricing problem to find conjunctions omitted from J that can improve the objective. Add these conjunctions to J.
3. Repeat steps 1 and 2 until no improving conjunctions can be found.
4. Solve the unrelaxed master problem ($w_k \in \{0,1\}$) restricted to the final subset J.

We refer to [Dash *et al.*, 2018] for the formulation of the pricing problem and more details in general.

3.2 Rule Sets With Grounding (BRCG+)

We now deal with the case where we use external information to guide the rule set generation procedure. The primary reason is the practical necessity for 'stabilization'. Decision rule sets learnt at increments as training data accumulates should be relatively stable to allow for comparison. There are other reasons as well when such grounding can be useful. Users may have domain knowledge that is not reflected in the data or may have formed mental models around existing ML models [Kulesza *et al.*, 2012]. Conjunctions generated by the algorithm may also have errors that need to be corrected.

Assume we are given a known set of conjunctions U where $U \subseteq K$. We seek to *ground* the subsequent search for conjunctions in this known set of rules. We consider this as a soft constraint, where each time an externally provided rule is violated we incur a small penalty.

To do this, we modify the objective (1) as:

$$\min_{\xi,w} \sum_{i\in P} \xi_i + \sum_{i\in Z} \sum_{k\in K_i} w_k + \underbrace{c_u n \sum_{k\in U} (1 - w_k)}_{\substack{\text{penalize violated} \\ \text{user constraints}}} \tag{5}$$

subject to constraints (2)–(4). The additional term serves as regularization and c_u is the regularization parameter as a fraction of the dataset size n. Since the first two terms in the objective represent the Hamming loss, $c_u n$ can be interpreted as the additional Hamming loss that needs to be incurred before a user provided constraint is dropped from the model.

This grounded problem can be solved by a similar CG procedure as above. First, a restricted master problem is solved by optimizing objective (5) subject to constraints (2) and (3).

It is naturally assumed that the grounding set $U \subseteq J$, the set considered in each iteration. In step 2, the pricing problem defined in [Dash *et al.*, 2018] (eq. (6)-(10) therein) can be used directly without modification (i.e. it is specified in the same way using the dual variables from the restricted master problem). The reason is that the new penalty term in (5) applies only to the grounding set $U \subseteq J$ and therefore does not alter the search for candidate conjunctions in $K \backslash J$.

The BRCG algorithm is sensitive to class imbalance and needs relatively balanced samples to generate rule sets. In practice sampling can be used to overcome this. Alternatively, the weights of the two terms in the objective (1) can be changed. Further, the feature vector X needs to be binarized. This can be accomplished by feature thresholding or by supervised approaches to determine optimal thresholds, such as by learning a decision tree.

A simple example of rule sets generated by BRCG and the BRCG+ extension can be seen for the `mushroom` dataset. For this case, the rule set generated consists of a single conjunction

$$(\text{odor} \neq \text{almond}) \wedge (\text{odor} \neq \text{anise}) \wedge (\text{odor} \neq \text{none}).$$

An `if-then-else` decision rule using this clause alone has a fidelity of 98.5%, i.e. correctly classifies if a mushroom is poisonous or not for 98.5% of training data.

3.3 Rule Comparisons

Rule sets can be similar in two distinct ways. On one hand, two rules may have few or no features in common but may be satisfied by very similar populations due to strong correlation. The reverse is also true: for example, the populations satisfying (age $>$ 26 AND education = 'Masters') and (age $<$ 26 AND education = 'Masters') have no overlap but the rules are semantically similar. Therefore, we consider two mechanisms, one statistical and another based on rule semantics to compare rule sets. A *rule set similarity score* between 0 and 1 (1 for identical rule sets) is computed using the following procedure.

Given two rule sets R_1 with m' rules and R_2 with n' rules, and a similarity function $S : R_1 \times R_2 \mapsto [0, 1]$, we solve an *assignment problem* where the objective is to find the maximum similarity matching $f : R_1 \mapsto R_2$ that maximizes $z = \sum_{r \in R_1} S(r, f(r))$. As the rule sets can be of different lengths, $m' \neq n'$, this can be an imbalanced assignment with some rules not matched, i.e. some rules in R_1 can be mapped to a "null rule" indicating no match. Computing f is a well studied problem and we use the algorithm described in [Crouse, 2016] as available in the `scipy` package.

The rule set similarity score, z^*, is normalized to $[0, 1]$ by taking $\frac{2z}{m'+n'}$. This normalization is inspired by the Sørenson-Dice coefficient and penalizes unmatched rules. The largest possible value of z is $\min\{m', n'\}$, corresponding to all rules in the shorter rule set being perfectly matched. A denominator of $\max\{m', n'\}$ would fully penalize unmatched rules. Our choice of the average $(m' + n')/2$ for the denominator is therefore intermediate.

We now describe two methods to compute the similarity function S, which is the pairwise similarity of all rules within rule sets R_1 and R_2.

Statistical similarity. Two rules can be considered similar if they cover a similar subset of a dataset X. If $X(r)$ denotes a subset satisfying rule r, then a statistical similarity measure can be computed as the Jaccard Index

$$S(r_1, r_2) = \frac{|X(r_1) \cap X(r_2)|}{|X(r_1) \cup X(r_2)|}. \tag{6}$$

This measure is straightforward to compute but requires knowledge of the underlying dataset X. For datasets with primarily categorical features, this similarity measure tends to be conservative as the intersection may be low. On the other hand, rules involving different features that are correlated can exhibit high statistical similarity.

Semantic similarity. To compare rule semantics, we regard a rule as an unordered set of literals where each literal is a triple $\langle$*variable, comparison operator, value*$\rangle$ Two literals can differ by any of these 3 elements. In order to take into account the semantic meaning of the literals, we define a *literals similarity function* L which assigns a similarity score between two literals based on the four cases: (*a*) 0.0 if they have different variables, (*b*) 0.25 if they have the same variables but different comparison operators, (*c*) 0.5 if they share the same variable and comparison operator but different values, (*d*) 1.0 if they are identical. We note that although these values $(0.0, 0.25, 0.5, 1.0)$ are arbitrary and adopted for concreteness, only relative magnitudes are of importance as they are used for comparison. We also have one exception to case (*b*). The literals produced by BRCG use 6 different comparison operators $(=, \neq, <, \leq, >, \geq)$. The two pairs $(<, \leq)$ and $(>, \geq)$ are operators that have similar meaning and are considered to be the same, falling in case (*c*).

As done for matching of rule sets, to find a match between the literals of two rules, we first compute pairwise literals similarity scores and then solve an assignment problem. Given two conjunctions, A_1 consisting of n literals and A_2 of m literals, we define a *rules similarity function* between the two rules/conjunctions as follows:

1. Interpreting A_1 as a set of n literals and A_2 as a set of m literals, and given the literals similarity function $L : A_1 \times A_2 \mapsto [0, 1]$, we define an *assignment problem* where the objective is to find a matching $f : A_1 \mapsto A_2$ so as to maximize the similarity $v = \sum_{a \in A_1} L(a, f(a))$
2. Given v^* as the maximum value of the cost function from the assignment problem in step 1, we compute the rules similarity score as $\frac{2v^*}{n+m}$. The same discussion as before applies to the choice of normalization.

This rules similarity function is taken as the *semantic similarity cost function* for computing the pairwise similarity $S(r_1, r_2)$ of all rules within two rule sets. We emphasize that semantic similarity scores can be computed without any knowledge of the underlying data.

4 Experiments

An empirical evaluation of the proposed methods is performed on seven binary classification datasets. The datasets range from 569 observations to 1.3 million observations. Full

details are in the supplementary material. The experiments support four claims around (a) stability of rules generated by the grounding mechanism, (b) recovery of artificially induced perturbations by our method, (c) comparison of semantic and statistical similarity computation, and (d) qualitative insights from interpretable model comparison.

Data binarization. For all datasets, categorical features were one-hot encoded with negations (i.e. both $=$ and $\neq$). Numerical features were compared to decile thresholds in both directions ($\leq$, $>$) and encoded for all thresholds. Only for the largest dataset on lending, a supervised data binarization method was used, where optimal splits were determined by a decision tree. This method yields considerably fewer splits than using decile thresholds and significantly reduces run times. Additional details are included in the supplement.

Algorithms. We used three base ML models - a standard logistic regression classifier with ℓ_2 regularization, Random Forest, and XGBoost. These are the models we seek to explain for all experiments. For the BRCG method, we used the open source implementation[1] described in [Arya *et al.*, 2019] as "BRCG-light". The rule comparator and BRCG+ method were implemented in Python 3.7 using CPLEX 12.10 as the solver for linear and integer programs.

Rule comparisons. We take 80% of each dataset as the training data. The ML model is used to generate labels for the training data, which is then input to the BRCG explainer to derive an initial rule set, referred to as "A". The remaining 20% of data is used for testing of model accuracy and rule set fidelity (accuracy of rules in predicting base learner labels). These, along with clause complexity metrics, are reported in the supplement. The training data is then artificially perturbed using the procedure described below. For this perturbed dataset, we learn a revised ML model and learn two versions of the explainer - BRCG and BRCG+, to get two rule sets, termed "B" and "B with grounding". The resulting rule sets are compared with the rule comparator (Section 3.3), i.e. we compare A-B and A-B with grounding.

Perturbation. We artificially induce a known change to the underlying data and see if our rule comparison procedure can recover the change as an interpretable clause. We pick an arbitrary rule from set A and change target labels for instances in the training data that satisfy the rule, with a perturbation probability p. As an instance can satisfy several rules we impose an additional criterion that perturbed data points be covered by the chosen rule exclusively. Without this, the perturbation can influence several rules all at once.

Complexity considerations. For BRCG-light, the complexity of explanation is controlled by two parameters λ_0 and λ_1 as opposed to a complexity bound C in Eq. (3). These parameters were determined by grid search. The parameters were changed in the mushroom case to provide larger rules, as the optimal rule had only three literals. The complexity of the rule sets resulting from BRCG-light was computed as the total number of literals in the rule set. This varied from 7 to 52 in our experiments. This complexity was input to the BRCG+ model as the complexity bound C.

[1] https://github.com/Trusted-AI/AIX360

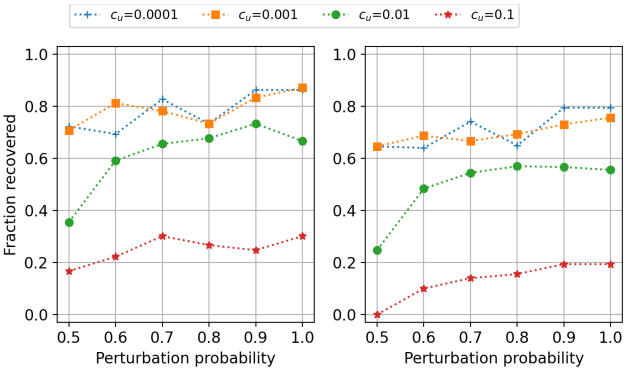

Figure 2: Fraction of instances for which perturbed rule was successfully recovered using statistical (left) and semantic (right) similarity

Experiments. We perturb each dataset with perturbation probabilities $p \in \{0.5, 0.6, .., 1.0\}$, which denotes the likelihood that an instance label is changed. We vary the grounding penalty $c_u \in \{0.0001, 0.001, 0.01, 0.1\}$ to see its impact. Overall, we analyze 7 datasets, 3 base ML models, 6 values of p, and 4 grounding penalties. For each run, we run both the BRCG and BRCG+ algorithms on 5 random perturbations for all datasets, except for the 'lending' dataset, where we run one draw.

4.1 Results

Change detection. For each perturbation run, if the perturbed rule (from A) was identified by the rule comparison as a change, we count this as being a successful recovery of the change. We are interested in the fraction of cases for which the induced change was recovered. The strength of the perturbation depends, in part, on the perturbation probability, and recovery rates were also found to depend on the regularization parameter c_u used. In Figure 2, we see that as the perturbation probability increases, the fraction of recovered changes also increases, to almost all cases when c_u is small enough. The change is thus successfully recovered if the perturbation is large enough compared to the regularization c_u.

The recovery rate is clearly impacted by the choice of the regularization parameter c_u. High values, e.g. $c_u = 0.1$, make the rules in A persist in B even when the data no longer support them. For lower values, the algorithm accounts for both grounded rules and changes in the underlying ML model as evidenced by the recovery fraction. One might think that a very low penalty c_u would cause grounded rules to be lost in set B. The generally high similarity scores with grounding in our current experiments (Figure 3) suggest otherwise.

Grounding. To see the difference grounding makes in generating stable rule sets, we compute the similarity measures between rule sets A and B (the perturbed data) using the BRCG and BRCG+ algorithms. While BRCG+ is grounded in rules from A, BRCG is not. Figure 3 shows that similarity scores are higher when the rule sets are grounded. This is regardless of if we use rule semantics or statistical comparisons.

Model comparison example. We conclude by highlighting the qualitative benefits of interpretable model comparisons

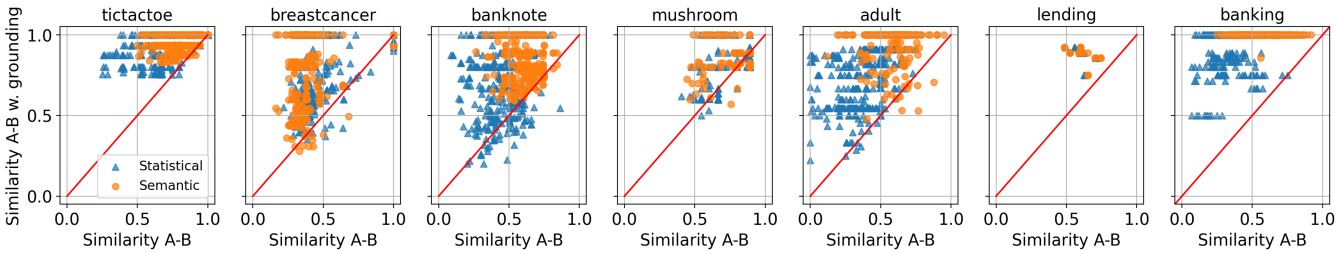

Figure 3: Rule set similarity with and without grounding showing grounded rule sets are more stable (above the diagonal line).

Figure 4: Model comparison for the 'adult' dataset. Both models are approximated by rule sets that predict incomes >50K. Each row shows a rule from the XGB rule set, the matched rule from the RF rule set if any, and the statistical similarity score S from Eq.(6) for a match.

through an example. We compare two ML models, Random Forest (RF) and Gradient Boosted Trees (XGB), for the income classification task of the 'adult' dataset, this time without artificial changes. The model comparison, summarized in Figure 4, shows that RF and XGB have test accuracies of 0.80 and 0.84 respectively. However, the statistical similarity score (z^*) of 0.44 between their rule set approximations suggests a low similarity between the models. The low similarity can be explained by the matching of the two rule sets, computed as described in Section 3.3 and depicted in Figure 4. The bottom two rules indicate a cohort in the training data for which the models behave similarly. The XGB model additionally captures four additional cases which are not covered by the RF model. Rule fidelity, i.e. the accuracy of the rule sets in explaining the two models, is 0.81 (RF) and 0.94 (XGB). The top row of the plot shows the "missed samples", the fraction of data for which the rule set infers the wrong label compared to the base learner. There is some overlap in the missed samples which is 1.0 minus the fidelity in each case. The high fidelity for XGB shows the rule set generated by our method serves as a meaningful surrogate for highly non-linear models. Additional model comparison examples for other datasets are provided in the supplement.

5 Conclusions

We present a solution to the problem of interpretable model comparison. The solution seeks to describe an ML model by compact Boolean rules. Subsequent versions of the model are similarly described using an explainer that aims to preserve rule structure across model versions. The algorithm is demonstrated on seven benchmark datasets to show that it can recover artificially induced perturbations and uncover qualitative changes.

The work has the following limitations. The BRCG+ algorithm is computationally intensive, particularly for large datasets. Faster heuristics or search methods for generating new conjunctions using rule based models like in [Arya *et al.*, 2019] could be leveraged to address this. The selection of regularization parameter c_u is important. Large c_u values can cause rules to persist that may no longer be valid. Low values do not appear to harm the stabilization of rules in our experiments. We have not systematically addressed the problem of a suitable choice for this parameter. Finally, the perturbation experiments performed in this paper are a surrogate endpoint and may not fully capture model change dynamics in real-world applications.

We believe nevertheless that this work makes a contribution in advancing interpretability in the context of model change. We have demonstrated that our solution can not only identify *when* a model has changed but is also able to uncover *what* has changed. Given the increasing reliance on AI solutions for real world problems, providing insights on changes beyond accuracy measures is important in supporting end users and model developers alike.

Acknowledgements

We thank Akihiro Kishimoto for feedback on an initial draft.

References

[Arya *et al.*, 2019] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012*, 2019.

[Ba and Caruana, 2014] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2654–2662, 2014.

[Bansal *et al.*, 2019] Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S Weld, Walter S Lasecki, and Eric Horvitz. Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2429–2437, 2019.

[Bastani *et al.*, 2017] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpretability via model extraction. In *Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT/ML)*, 2017.

[Bu *et al.*, 2019] Yuheng Bu, Jiaxun Lu, and Venugopal Veeravalli. Model change detection with application to machine learning. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5341–5346, 2019.

[Bucilă *et al.*, 2006] Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, page 535–541, 2006.

[Cohen, 1995] William W. Cohen. Fast effective rule induction. In *International Conference on Machine Learning (ICML)*, pages 115–123, 1995.

[Craven and Shavlik, 1995] Mark W. Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 24–30, 1995.

[Crouse, 2016] David Frederic Crouse. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679–1696, 2016.

[Dash *et al.*, 2018] Sanjeeb Dash, Oktay Günlük, and Dennis Wei. Boolean decision rules via column generation. In *Advances in Neural Information Processing Systems*, pages 4655–4665, 2018.

[Demšar and Bosnić, 2018] Jaka Demšar and Zoran Bosnić. Detecting concept drift in data streams using model explanation. *Expert Syst. Appl.*, 92:546–559, 2018.

[Domingos, 1997] Pedro Domingos. Knowledge acquisition from examples via multiple models. In *International Conference on Machine Learning (ICML)*, pages 98–106, 1997.

[Drozdal *et al.*, 2020] Jaimie Drozdal, Justin Weisz, Dakuo Wang, Gaurav Dass, Bingsheng Yao, Changruo Zhao, Michael Muller, Lin Ju, and Hui Su. Trust in automl: Exploring information needs for establishing trust in automated machine learning systems. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, IUI '20, page 297–307, New York, NY, USA, 2020. Association for Computing Machinery.

[Frosst and Hinton, 2017] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. In *Comprehensibility and Explanation in AI and ML (CEX) Workshop, 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA)*, 2017.

[Gama *et al.*, 2014] João Gama, Indrundefined Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4), March 2014.

[Geng *et al.*, 2019] Jun Geng, Bingwen Zhang, Lauren M. Huie, and Lifeng Lai. Online change-point detection of linear regression models. *IEEE Transactions on Signal Processing*, 67(12):3316–3329, 2019.

[Harel *et al.*, 2014] Maayan Harel, Koby Crammer, Ran El-Yaniv, and Shie Mannor. Concept drift detection through resampling. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1009–II–1017. JMLR.org, 2014.

[Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[Kulesza *et al.*, 2012] Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. Tell me more? the effects of mental model soundness on personalizing an intelligent agent. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1–10, 2012.

[Lakkaraju *et al.*, 2016] Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1675–1684, 2016.

[Lopez-Paz *et al.*, 2016] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. In *International Conference on Learning Representations (ICLR)*, 2016.

[Sanchez *et al.*, 2015] Ivan Sanchez, Tim Rocktaschel, Sebastian Riedel, and Sameer Singh. Towards extracting faithful and descriptive representations of latent variable models. In *AAAI Spring Syposium on Knowledge Representation and Reasoning (KRR): Integrating Symbolic and Neural Approaches*, volume 1, pages 4–1, 2015.

[Su *et al.*, 2016] Guolong Su, Dennis Wei, Kush R. Varshney, and Dmitry M. Malioutov. Learning sparse two-level Boolean rules. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, September 2016.

[Tan *et al.*, 2018] Sarah Tan, Rich Caruana, Giles Hooker, and Yin Lou. Distill-and-compare: Auditing black-box models using transparent model distillation. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, page 303–310, 2018.

[Tsymbal, 2004] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Technical Report, Department of Computer Science, Trinity College Dublin*, 2004.

[Wang *et al.*, 2017] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. A Bayesian framework for learning rule sets for interpretable classification. *Journal of Machine Learning Research*, 18(70):1–37, 2017.

[Wang *et al.*, 2019] Dakuo Wang, Justin D. Weisz, Michael Muller, Parikshit Ram, Werner Geyer, Casey Dugan, Yla Tausczik, Horst Samulowitz, and Alexander Gray. Human-ai collaboration in data science: Exploring data scientists' perceptions of automated ai. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW), November 2019.