Towards Dependable and Explainable Machine Learning Using Automated Reasoning

Hadrien Bride[2], Jie Dong[3], Jin Song Dong[1,2], Zhé Hóu[2]

[1] School of Computing, National University of Singapore, Singapore
[2] Institute for Integrated and Intelligent Systems, Griffith University, Australia
[3] Dependable Intelligence

Abstract. The ability to learn from past experience and improve in the future, as well as the the ability to reason about the context of problems and extrapolate information from what is known, are two important aspects of Artificial Intelligence. In this paper, we introduce a novel automated reasoning based approach that can extract valuable insights from classification and prediction models obtained via machine learning. A major benefit of the proposed approach is that the user can understand the reason behind the decision-making of machine learning models. This is often as important as good performance. Our technique can also be used to reinforce user-specified requirements in the model as well as to improve the classification and prediction.

1 Introduction

Philip Wadler once wrote that "powerful insights arise from linking two fields of study previously thought separate" [6]. This paper, although does not provide a similar correspondence relation between two fields such as logic and computation, aims at finding an interesting application that combines machine learning and automated reasoning. There have been various attempts at applying machine learning in automated reasoning. For instance, the automated reasoning tool Sledgehammer, which is a subsystem of the proof assistant Isabelle/HOL, has a module named MaSh [4] which uses machine learning to rank the relevance of known facts in the proof context based on previous successful proofs and select a subset of facts that is estimated most helpful in proving the existing goals. The other direction, i.e., applying automated reasoning in machine learning, has not seen an application as far as we are aware of.

Recently, eXplainable Artificial Intelligence (XAI) has been gaining attention. The prestigious International Joint Conference on Artificial Intelligence (IJCAI) has notably been running a workshop specialised in this topic. From the automated reasoning community, Bonacina recently envisaged that automated reasoning could be the key to the advances of XAI and machine learning [1]. To this end, she posed several questions and challenges in this direction. Specifically,

> "How can we bridge the gap between the statistical inferences of machine learning and the logical inferences of reasoning, applying the latter to extract, build, or speculate and test, explanations of the former?" [1]

This paper addresses the above challenge by proposing a novel framework which enables the application of automated reasoning in machine learning. First, we study a range of machine learning techniques and identify that models produced by (ensemble) decision trees based techniques are suitable for formal analysis and automated reasoning. We then propose to use satisfiability modulo theories (SMT) solvers to perform analysis on classification and prediction models given by machine learning. We discuss some preliminary results using a new machine learning tool called Silas [3].

2 Machine Learning Techniques Reviewed

To solve a problem one has to choose the right tool. In our context, to fully support the integration of automated reasoning techniques and to build towards a machine learning approach that is explainable and dependable, we require efficient machine learning techniques (in term of both memory and time) that produce interpretable models.

Linear regressions (LR), while being easily interpretable, often fall short in performance compared to other approaches [7].

Support vector machines (SVM) are popular and efficient tools, which, thanks to a large number of kernels, can be applied to a variety of classification problems. However, interpreting the models produced by SVM is far from trivial, especially when non-linear kernels are used. Hence, SVM is often used as black-boxes.

Neural networks (NN) and deep learning (DL) techniques have been exceptionally successful in analysing both structured and unstructured data, but the models they produce are intricate, hence NN and DL are often used as black-boxes, too. Also, NN and DL are often computationally expensive.

In contrast to SVMs and NNs, decision trees (DT) based techniques such as random forests (RF) and gradient boosting machines (GBM) are capable of producing interpretable and explainable models due to the formal semantics associated with there underlying tree structures. Moreover, when analysing structured data, RF and GBM often outperform other approaches including DL [7, 5].

Given all the above observations, we conclude that a decision tree based machine learning approach fits our needs. The formal semantics of decision trees offers an ideal support for the application of formal methods.

3 Model Analysis and Engineering using SMT

This sections briefly presents some of the ideas we are actively developing. They provide the basic building blocks for the application of automated reasoning tools such as SMT solvers to perform the analysis of decision tree based models.

Obtaining model predicates: Given a classification or a prediction model that consists of a set of decision trees, we first need to extract logical formulae from the trees. A decision tree in this context is a data structure in which every non-leaf node is associated with a logical formula that splits the data entries into two subsets. Assuming that $A, B, \cdots$ are the classes to be classified or predicted in a data set. Each leaf node contains a subset of data entries labelled by the classes. An algorithm such as majority voting is then needed to obtain the final decision. There are multiple ways to obtain logical formulae for model analysis. One can collect all the formulae from the root of a tree to a leaf node with decision *A*, and the *conjunction* of these formulae, called the *branch formula*, gives the reason why the subset of data entries in the leaf node are classified/predicted as *A*. There could be multiple leaf nodes whose decisions are all class *A*. The *disjunction* of all branch formulae which lead to class *A* represents the overall decision-making of the tree with respect to class *A*. We refer to this disjunction as the *decision formula* for class *A*. The decision formula for a class can then be used in the analysis to check inconsistencies and extract the core reason behind the decision-making. For a set of decision trees, we can extract the decision formula for class *A* on each tree and perform analysis on the conjunction of multiple decision formulae.

Model anlaysis and engineering: Once the logical formulae are extracted, we can perform a number of analyses on the formulae using automated reasoning techniques. We list some of the specific techniques we have been successfully using so far.

Maximum Satisfiable Subset (MSS): To obtain the MSS, we assign a weight to each sub-formula, and try to maximise the accumulated weight in the MAX-SMT optimisation problem. This can be achieved by certain SMT solvers such as Z3 and MathSAT 5. The weight assigned to each formula can be optimised to reflect the predictive performance of the decision node or the decision tree. For instance, a decision node with more information gain may have more weight, and a decision tree with higher predictive accuracy may have more weight. The resulting MSS can give an indication of the core attributes and the range of the attributes that lead to the decision-making of the classification and prediction model. Note that a similar analysis is to extract *maximal satisfiable subsets*, which is computationally cheaper, but we prefer the maximum subset because it may give more insight about the decision-making.

Minimal Unsatisfiable Core (MUC): Solvers such as Z3 provide a straightforward way to compute the MUC of a set of formulae. We can use this functionality to obtain the inconsistencies in the model and use this information to fine-turn the model by trimming the decision tree. This can form a recursive procedure in which we repeatedly find the MUCs in a decision tree and trim the tree accordingly until the tree becomes a consistent model. Another application is to use the MUC in boosting. A boosting algorithm usually consists of iterative learning steps in which weak classifiers are introduced to compensate the shortcomings of existing weak learners. The MUC can be effectively used as the shortcomings of multiple learners, because it represents the disagreements of multiple decision trees. We can then build weak classifiers around the MUC to boost the model performance.

Model Verification: In certain applications, the user may specify some requirements that a decision making procedure must satisfy. For instance, if machine learning is applied to classify whether a node in a network cluster is secure, our method may produce a logical condition for deciding network security. If the user has other security requirements that must be satisfied, we can use SMT solving and model checking to verify that the requirements hold in the learned model. If this is not true, then the MUC analysis can pinpoint the reason why the learned condition fails and we can use the MUC to tweak the model by inserting decision nodes that enforce the user requirements and obtain machine learning results that conform the user's specifications.

4 Discussion

The model analysis and engineering component for machine learning can provide several benefits to users at different levels: (1) The analysis can pinpoint the reason behind the classification and prediction. This will help the user (e.g., decision maker) understand what the key attributes are and how they lead to the result. Therefore, the user can use the analytical information provided by this approach to make the final decision based on their discretion. Moreover, the analytical information can help transform the machine learning algorithm into a transparent process in which every decision can be inspected and verified. (2) The analysis can also provide the reason why some models have good performance while others have bad performance. Data scientists can use this information to improve the learning process and perform hyperparameter tuning. (3) Machine learners can use the MUC to fine-tune the models and improve classification and prediction results. They can also use the MSC to build new and consistent models that potentially have better results. (4) Model verification helps obtain machine learning results that conform with user-specified requirements. This is vital in providing a machine learning technique that can be trusted.

We have implemented and experimented with the approach introduced in this paper. We have produced a module for the machine learning tool Silas [3]. As an example, on a diabetes data set [2], we are able to analyse a random forest model and obtain a set of "core reasons" behind each class (negative/positive diabetes). By comparing the core reasons, we derive that $30 \leq \textit{age} \leq 34$ and $0 < \textit{number of times pregnant} \leq 2$ are among the key indicators for classifying positive diabetes, whereas $21 \leq \textit{age} \leq 22$ and $\textit{number of times pregnant} \leq 0$ strongly indicate negative diabetes. On the other hand, we were able to deduce that *2-hour serum insulin* is not a strong indicator for either classes, which implies that data scientists can perform certain feature engineering on the data set to improve the results. Note that the data set only contains 768 data entries (patients), so the analysis may not be representative for a large population. Nonetheless, our implementation demonstrates the feasibility of the proposed method and shows that the combination of machine learning and automated reasoning has the potential to provide a new explainable and dependable data analysis technology.

References

1. Maria Paola Bonacina. Automated reasoning for explainable artificial intelligence. In *ARCADE Workshop (in association with CADE-26), Gothenburg, Sweden*, 2017.
2. Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.
3. Dependable Intelligence. Silas. `https://www.depintel.com/silas_about.html`, 2018.
4. Daniel Kühlwein, Jasmin Christian Blanchette, Cezary Kaliszyk, and Josef Urban. Mash: machine learning for sledgehammer. In *ITP*, pages 35–50. Springer, 2013.
5. Szilard Pafka. A minimal benchmark for scalability, speed and accuracy of commonly used open source implementations of the top machine learning algorithms for binary classification. https://github.com/szilard/benchm-ml, 2018.
6. Philip Wadler. Propositions as types. *Communications of the ACM*, 58(12):75–84, 2015.
7. Chongsheng Zhang, Changchang Liu, Xiangliang Zhang, and George Almpanidis. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, 82:128–150, 2017.