26th International Symposium on Formal Methods (Doctoral Symposium),
SEFM 2024, Milan, Italy, September 9-13, 2024

Towards Comparing Learned Classifiers
Reviewed and Accepted at Doctoral Symposium of FM'24

Soaibuzzaman

Bauhaus-University Weimar, Germany

Abstract. Machine learning for classification has seen numerous applications to complex, real-world tasks. Learned classifiers have become important artifacts of software systems that, like code, require careful analyses, maintenance, and evolution. Existing work on the formal verification of learned classifiers has mainly focused on the properties of individual classifiers, e.g., safety, fairness, or robustness, but not on analyzing the commonalities and differences of multiple classifiers.
This PhD project envisions MLDiff, a novel approach to comparing learned classifiers where one is an alternative or variant of another. MLDiff will leverage an encoding to SMT and can discover differences not (yet) seen in available datasets. We outline the challenges of MLDiff and present an early prototype.

Keywords: machine learning, comparison, SMT

1 Introduction

Machine learning for classification has seen numerous applications to complex, real-world tasks. Existing work on the formal verification of learned classifiers has mainly focused on the verification of properties, e.g., safety [8,9], fairness [17,14], or robustness [4,1], of individual classifiers but not on the commonalities and differences of multiple classifiers. Learned classifiers become important artifacts of software systems that, like code, require careful analyses, maintenance, and evolution. Typically, alternative or evolved classifiers are compared by individual metrics on known datasets or variants of the above properties. While these are meaningful and important, they do not provide comparisons in terms of disagreements or common instances.

This PhD project envisions MLDiff, a novel approach to comparing learned classifiers. Our comparison leverages an encoding to SMT [11] and can uncover differences not (yet) seen in available datasets or show their absence. Typical use cases include checking disagreements of classifiers or uncovering safety-critical differences, e.g., computing instances that witness (mis-)classifications.

2 Motivating Example Comparison of two Classifiers

Consider the well-known dataset of Iris flowers [6] for classifying flowers by their petal and sepal length and width (four features) into three species. A team of

$$argMax(\begin{pmatrix} 0.21443255 & 0.38513516 & -0.79414658 & -0.44987553 \\ -0.11326783 & -0.59086647 & 0.84205491 & -1.65158089 \\ -0.75651555 & -0.91195898 & 1.08941819 & 1.75014748 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0.10703851 \\ 0.97002258 \\ -1.01034086 \end{pmatrix})$$

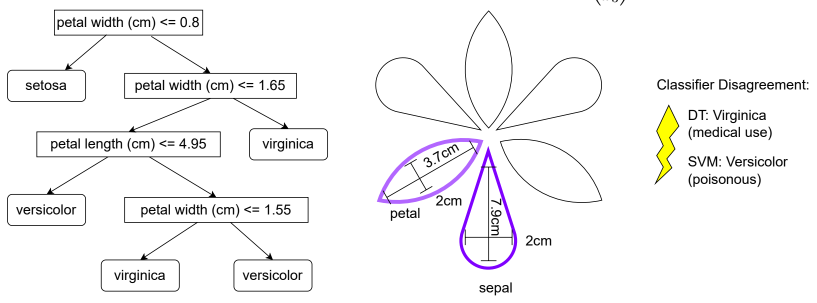

Fig. 1. A learned SVM (Linear Support Vector Classifier) [top], a learned decision tree [left] (both for the Iris dataset), and an analysis result computed by MLDiff predicted as *Virginica* (medical use) by the DT and as *Versicolor* (poisonous) by the SVM.

engineers has trained the SVM and the Decision Tree (DT) classifiers shown in Fig. 1 [top] and [left] for evaluation. A traditional comparison reveals an accuracy of 96% for both classifiers and an F1-score of 96% for the DT and 93% for the SVM, making the DT the preferred choice.

However, an MLDiff analysis shows that the DT sometimes classifies instances as Virginica species (used for medical purposes) when the SVM classifies them as the poisonous Versicolor [5]. This disagreement (shown in Fig. 1 [right]) and potentially serious misclassification alarms the engineers. They consult a horticulturist for clarification before deploying either classifier. Note that no such instance is contained in the dataset, i.e., even an exhaustive search through the dataset would not have revealed the potentially dangerous disagreement.

3 MLDiff for Classifier Comparison

We now introduce the conceptual framework MLDiff. The goal of MLDiff is to provide deeper insight into differences of classifiers. We consider classifiers as total functions that map instances to classes as stated in Def. 1. Def. 2 presents the MLDiff classifier combination we use for our analyses.

Definition 1 (Classifier). *A classifier over features X is a total function $cl : \mathbb{R}^{|X|} \to C$ that maps each instance $d \in D = \mathbb{R}^{|X|}$ to a class $c \in C$.*

Definition 2 (MLDiff Classifier Combination). *Given classifier $cl_1 : \mathbb{R}^{|X_1|} \to C_1$ and classifier $cl_2 : \mathbb{R}^{|X_2|} \to C_2$ we construct the MLDiff classifier combination $cl_1 \oplus cl_2 : \mathbb{R}^{|X_1 \cup X_2|} \to C_1 \times C_2$ such that*

$$\forall d \in \mathbb{R}^{|X_1 \cup X_2|} : \quad cl_1 \oplus cl_2(d) = cl_1(d|_{X_1}) \times cl_2(d|_{X_2})$$

where $d|_{X_i}$ projects $\mathbb{R}^{|X_1 \cup X_2|}$ to $\mathbb{R}^{|X_i|}$ with corresponding values for features X_i.

Cases where features $X_1 = X_2$ are very common when using classifiers on similar datasets. Note that common features are shared between classifiers, while classes and classifications are always kept separate. A straightforward implementation of classifier comparison in MLDiff is based on a translation of the classifiers to SMT formulas (see examples for individual classifiers in [16]).

4 Challenges and Open Problems

Limitations of SMT-Encodings It is well known that SMT solvers suffer from incompleteness, e.g., our prototype uses the Z3 SMT-solver [11] and currently does not support non-linear kernels for SVMs nor activation functions for the hidden layers of the MLP other than ReLU and identity. Frameworks such as Reluplex [8] and its successor Marabou [9] introduce specialized handling for some of the required arithmetic. Similarly, naive SMT encodings are not expected to perform well, and analysis tools for neural networks [3,9,12] employ approximation and parallelization techniques. Related approaches will have to be devised and applied in MLDiff.
Use Cases adequate Queries We envision various and flexible use cases of MLDiff for the comparison of learned ML models. Analyses may be customized using queries, i.e., assertions over feature variables and predicted classes. Basic queries about classifier disagreement may be encoded by the query $cl_1 \neq cl_2$ (used for Fig. 1). We consider extensions of property specification languages such as MLCHECK [14] and Vehicle [2] to multiple classifiers as necessary parts of MLDiff to support the formulation of advanced analysis queries.
Relevant and Interesting Examples Naively, MLDiff could produce example outputs with implausible results, e.g., negative petal lengths for iris flowers. Property specification languages [14,2], might allow users to formulate domain knowledge, but likely effective methods are required to also learn constraints from data [13]. An additional challenge for MLDiff will be the efficient encoding of these into the SMT solver.

SMT-based analyses are likely prone to adversarial examples. Many works have addressed various notions of *robustness* of individual classifiers [4,1,14,2] while MLDiff needs to extend these to multiple classifiers.
Example Exploration and Explanation Given domain-specific queries and constraints, it is likely still desired to inspect multiple examples. Efficient and effective methods will be needed for an exploration [7] of MLDiff's solution space.

Some examples found by MLDiff might not be anticipated, and thus, explanations of analysis results will be necessary. Adapting explainable AI [10] to MLDiff's setting requires the development of new multi-model concepts.

5 Conclusion

We have proposed MLDiff, a novel approach to the comparison of learned classifiers. We have motivated and illustrated the use of MLDiff on real-world datasets and examples. Our prototypical implementation supports Decision Tree, Support Vector Machine, Logistic Regression, and Neural Networks classifiers as the basis for the advanced features and to confirm most challenges from Sect. 4.

Data Availability

We have made the MLDiff framework and a prototypical implementation available on GitHub as [15].

References

1. Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A.V., Criminisi, A.: Measuring neural net robustness with constraints pp. 2613–2621 (2016), https://proceedings.neurips.cc/paper/2016/hash/980ecd059122ce2e50136bda65c25e07-Abstract.html
2. Daggitt, M.L., Kokke, W., Atkey, R., Arnaboldi, L., Komendantskaya, E.: Vehicle: Interfacing neural network verifiers with interactive theorem provers. CoRR abs/2202.05207 (2022)
3. Ehlers, R.: Formal verification of piece-wise linear feed-forward neural networks. In: ATVA 2017. LNCS, vol. 10482, pp. 269–286. Springer (2017). https://doi.org/10.1007/978-3-319-68167-2_19
4. Einziger, G., Goldstein, M., Sa'ar, Y., Segall, I.: Verifying robustness of gradient boosted models. In: AAAI 2019. vol. 33, pp. 2446–2453 (2019)
5. Elias, T.S., Dykeman, P.A.: Edible wild plants: a North American field guide. Sterling Publishing Company, Inc. (1990)
6. Fisher, R.A.: Iris. UCI Machine Learning Repository (1988), DOI: https://doi.org/10.24432/C56C76
7. Kang, E., Jackson, E.K., Schulte, W.: An approach for effective design space exploration. In: Monterey Workshop 2010. LNCS, vol. 6662, pp. 33–54. Springer (2010). https://doi.org/10.1007/978-3-642-21292-5_3
8. Katz, G., Barrett, C.W., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: CAV 2017. LNCS, vol. 10426, pp. 97–117. Springer (2017). https://doi.org/10.1007/978-3-319-63387-9_5
9. Katz, G., Huang, D.A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., Dill, D.L., Kochenderfer, M.J., Barrett, C.: The Marabou Framework for Verification and Analysis of Deep Neural Networks. In: CAV 2019. pp. 443–452. Springer. https://doi.org/10.1007/978-3-030-25540-4_26
10. Marques-Silva, J., Ignatiev, A.: Delivering trustworthy AI through formal XAI. In: AAAI 2022. pp. 12342–12350. AAAI Press (2022). https://doi.org/10.1609/AAAI.V36I11.21499
11. de Moura, L.M., Bjørner, N.S.: Z3: an efficient SMT solver. In: TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_24
12. Paulsen, B., Wang, J., Wang, C.: Reludiff: differential verification of deep neural networks. In: ICSE 2020. pp. 714–726. ACM (2020). https://doi.org/10.1145/3377811.3380337

13. Popescu, A., Erdeniz, S.P., Felfernig, A., Uta, M., Atas, M., Le, V., Pilsl, K., Enzelsberger, M., Tran, T.N.T.: An overview of machine learning techniques in constraint solving. J. Intell. Inf. Syst. 58(1), 91–118 (2022). https://doi.org/10.1007/S10844-021-00666-5
14. Sharma, A., Demir, C., Ngomo, A.N., Wehrheim, H.: MLCHECK- property-driven testing of machine learning classifiers. In: ICMLA 2021. pp. 738–745. IEEE (2021). https://doi.org/10.1109/ICMLA52953.2021.00123
15. Soaibuzzaman, Döring, J., Kasi, S., Ringert, J.O.: MLDiff github repository (2025), available from https://github.com/se-buw/MLDiff
16. Urban, C., Miné, A.: A review of formal methods applied to machine learning. CoRR abs/2104.02466 (2021)
17. Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viégas, F., Wilson, J.: The what-if tool: Interactive probing of machine learning models. IEEE TVCG 26(1), 56–65 (2019)