Interpretability via Model Extraction*

Osbert Bastani
Stanford University
obastani@cs.stanford.edu

Carolyn Kim
Stanford University
ckim@cs.stanford.edu

Hamsa Bastani
Stanford University
hsridhar@stanford.edu

ABSTRACT

The ability to interpret machine learning models has become increasingly important now that machine learning is used to inform consequential decisions. We propose an approach called *model extraction* for interpreting complex, blackbox models. Our approach approximates the complex model using a much more interpretable model; as long as the approximation quality is good, then statistical properties of the complex model are reflected in the interpretable model. We show how model extraction can be used to understand and debug random forests and neural nets trained on several datasets from the UCI Machine Learning Repository, as well as control policies learned for several classical reinforcement learning problems.

1 INTRODUCTION

Recent advances in machine learning have revolutionized our ability to use data to inform critical decisions, such as medical diagnosis [8, 19, 27], bail decisions for defendants [16, 17], and aircraft collision avoidance systems [25]. At the same time, machine learning algorithms have been shown to exhibit unexpected defects when deployed in the real world; examples include causality (i.e., inability to distinguish causal effects from correlations) [8, 21], fairness (i.e., internalizing prejudices present in training data) [13, 15], and algorithm aversion (i.e., lack of trust by end users) [11].

Interpretability is a promising approach to address these challenges [12, 24]—we can help human users diagnose issues and verify correctness of machine learning models by providing insight into the model's reasoning [3, 18, 20, 23, 26]. For example, suppose the user is trying to train a model that does not depend on a prejudiced feature (e.g., ethnicity). Omitting the feature might not suffice to avoid prejudice, since the model could reconstruct that feature from other features [22]. A better approach might be to train the model with the prejudiced feature, and then assess the dependence of the model on that feature. This approach requires the ability to understand the model's reasoning process, i.e., how model predictions are affected by changing the prejudiced feature [12]. Similarly, the user may want to determine whether dependence on a feature is causal, or understand the high-level structure of the model to gain confidence in its correctness.

In this paper, we propose an technique that we call *model extraction* for interpreting the overall reasoning process performed by a model. Given a model $f : \mathcal{X} \to \mathcal{Y}$, the interpretation produced by our algorithm is an approximation $T(x) \approx f(x)$, where T is an interpretable model. In this paper, we take T to be a decision tree, which has been established as highly interpretable [3, 20, 23]. Intuitively, if T is a sufficiently good approximation of f, then any issues in f should be reflected in T as well. Thus, the user can understand and debug f by examining T; then, the original model f can be deployed so that performance is not sacrificed.

Previous model extraction approaches have focused on specific model families [10, 28, 29], enabling them to leverage the internal structure of the model. In contrast, our approach is *blackbox*, i.e., it only requires the ability to obtain the output $f(x) \in \mathcal{Y}$ corresponding to a given input $x \in \mathcal{X}$. Thus, our approach works with any model family and is independent of the implementation. Complimentary approaches to interpretability focus on learning interpretable models [7, 26, 30] or on explaining the model's behavior on specific inputs rather than the model as a whole [23].

The key challenge to learning accurate decision trees is that they often overfit and obtain poor performance, whereas complex models such as random forests and deep neural nets are better regularized [4]. For example, random forests use ensembles of trees to avoid overfitting to specific features or training points.

*Presented as a poster at the 2017 Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT/ML 2017).

Thus, our algorithm uses active learning to construct T from f—it actively samples a large number of training points, and computes the corresponding labels $y = f(x)$. The large quantity of data ensures that T does not overfit to the small set of initial training points. We prove that under mild assumptions, by generating a sufficient quantity of data, the extracted tree T converges to the *exact* greedy decision tree, i.e., it avoids overfitting since the sampling error goes to zero.

We implement our algorithm and use it to interpret random forests and neural nets, as well as control policies trained using reinforcement learning. We show that our active learning approach substantially improves over using CART [6], a standard decision tree learning algorithm. Furthermore, we demonstrate how the decision trees extracted can be used to debug issues with these models, for example, to assess the dependence on prejudiced features, to determine why certain models perform worse, and to understand the high-level structure of a learned control policy.

2 MODEL EXTRACTION

We describe our model extraction algorithm.

2.1 Problem Formulation

Given a training set $X_{\text{train}} \subseteq \mathcal{X}$ and blackbox access to a function $f : \mathcal{X} \to \mathcal{Y}$, our goal is to learn an interpretable model $T : \mathcal{X} \to \mathcal{Y}$ that approximates f. In this paper, we take T to be an axis-aligned decision tree, since these models are both expressive highly interpretable. For simplicity, we focus on the case of classification, i.e., $\mathcal{Y} = [m] = \{1, ..., m\}$. We measure performance using accuracy relative to f on a held out test set, i.e., $\frac{1}{|X_{\text{test}}|} \sum_{x \in X_{\text{test}}} \mathbb{I}[T(x) = f(x)]$.

2.2 Algorithm

Our algorithm is greedy, both for scalability and because it is a natural fit for interpretability, since more relevant features occur higher in the tree.

Input distribution. First, our algorithm constructs a distribution $\mathcal{P}$ over the input space $\mathcal{X}$ by fitting a mixture of axis-aligned Gaussian distributions to the training data using expectation maximization.

Exact greedy decision tree. We describe the *exact greedy decision tree* T^*. We cannot construct T^* since we treat f as a blackbox; as we describe below, our algorithm approximates T^*. Essentially, T^* is constructed greedily as a CART tree [6], except the gain is computed exactly according to $\mathcal{P}$. For example, if the gain is the Gini impurity, then it is computed as follows:

$$\text{Gain}(f, C_N) = 1 - \sum_{y \in \mathcal{Y}} \Pr_{x \sim \mathcal{P}}[f(x) = y \mid C_N],$$

where C_N are the constraints encoding which points flow to the node N in T^* for which a branch is currently being constructed. Similarly, the most optimal leaf labels are computed exactly according to $\mathcal{P}$.

Estimated greedy decision tree. Given $n \in \mathbb{N}$, our algorithm estimates $\text{Gain}(f, C_N)$ using n i.i.d. samples $x \sim \mathcal{P} \mid C_N$, where C_N is a conjunction of axis-aligned constraints. We briefly describe how our algorithm obtains such samples. It is straightforward to show that the constraint C_N can be simplified so it contains at most one inequality $(x_i \leq t)$ and at most one inequality $(x_i > s)$ per $i \in [d]$. For simplicity, we assume C_N contains both inequalities for each $i \in [d]$:

$$C_N = (s_1 \leq x_1 \leq t_1) \wedge ... \wedge (s_d \leq x_d \leq t_d).$$

Then, the probability density function of $\mathcal{P} \mid C_N$ is

$$p_{\mathcal{P} \mid C_N}(x) \propto \sum_{j=1}^{K} \phi_j \prod_{i=1}^{d} p_{\mathcal{N}(\mu_{ji}, \sigma_{ji}) \mid (s_i \leq x_i \leq t_i)}(x_i).$$

Since the Gaussians are axis-aligned, the unnormalized probability of each component is

$$\begin{aligned}\tilde{\phi}'_j &= \int \phi_j \prod_{i=1}^{d} p_{\mathcal{N}(\mu_{ji}, \sigma_{ji}) \mid (s_i \leq x_i \leq t_i)}(x_i) dx \\ &= \phi_j \prod_{i=1}^{d} \left(\Phi\left(\frac{t_i - \mu_{ji}}{\sigma_{ji}} \right) - \Phi\left(\frac{s_i - \mu_{ji}}{\sigma_{ji}} \right) \right),\end{aligned}$$

where Φ is the cumulative density function of the standard Gaussian distribution $\mathcal{N}(0, 1)$. Then, the component probabilities are $\tilde{\phi} = Z^{-1}\tilde{\phi}'$, where $Z = \sum_{j=1}^{K} \tilde{\phi}'_j$. To sample $x \sim \mathcal{P} \mid C_N$, sample $j \sim \text{Categorical}(\tilde{\phi})$, and

$$x_i \sim \mathcal{N}(\mu_{ji}, \sigma_{ji}) \mid (s_i \leq x_i \leq t_i) \quad \text{(for each } i \in [d]).$$

We use standard algorithms for sampling truncated Gaussian distributions to sample each x_i.

2.3 Theoretical Guarantees

The extracted tree T converges to T^* as $n \to \infty$:

Theorem 2.1. Assume the exact greedy tree T^* is well defined, and the probability density function $p(x)$ is bounded, continuous, and has bounded support. Then, for any $\epsilon, \delta > 0$, there exists $n \in \mathbb{N}$ such that the tree T extracted by our algorithm using n samples satisfies $\Pr_{x\sim\mathcal{P}}[T(x) = T^*(x)] \leq \epsilon$, with probability at least $1-\delta$ over the training samples.

3 EVALUATION

We use our model extraction algorithm to interpret several supervised learning models trained on datasets from the UCI Machine Learning Repository [2], as well as a learned control policy from OpenAI Gym [1], i.e., the learned control policy $\pi : \mathcal{S} \to \mathcal{A}$.

3.1 Comparison to CART

First, we compare our algorithm to a baseline that uses CART to train a decision tree approximating f on the training set $\{(x, f(x)) \mid x \in X_{\text{train}}\}$. For both algorithms, we restrict the decision tree to have 31 nodes. We show results in Table 1. We show the test set performance of the extracted tree compared to ground truth (or for MDPs, estimated the reward when it is used as a policy), as well as the relative performance compared to the model f on the same test set. Note that our goal is to obtain high relative performance: a better approximation of f is a better interpretation of f, even if f has poor performance. Our algorithm outperforms the baseline on every problem instance.

3.2 Examples of Use Cases

We show how the extracted decision trees can be used to interpret and debug models.

Use of invalid features. Using an invalid feature is a common problem when training models. In particular, some datasets contain multiple response variables; then, one response should not be used to predict the other. For example, the breast cancer dataset contains two response variables indicating cancer recurrence: the length of time before recurrence and whether recurrence occurs within 24 months. This issue can be thought of as a special case of using a non-causal feature, an important problem in healthcare settings. We train a random forest f to predict whether recurrence occurs within 24 months, where time to recurrence is incorrectly included as a feature. Then, we extract a decision tree approximating f of size $k = 7$ nodes, using 10 random splits of the dataset into training and test sets. The invalid feature occured in every extracted tree, and as the top branch in 6 of the 10 trees.

Use of prejudiced features. We can use our algorithm to evaluate how a model f depends on prejudiced features. For example, gender is a feature in the student grade dataset, and may be considered sensitive when estimating student performance. However, if we simply omit gender, then f may reconstruct it from the remaining features. For a model f trained with gender available, we show how a decision tree extracted from f can be used to understand how f depends on gender. Our approach does not guarantee fairness, but it can be useful for evaluating the fairness of f.

We extract decision trees T from the random forests f trained on 10 random splits of the student grades dataset. The top features are consistently grades in other classes or number of failing grades received in the past. Gender occurs below these features (at the fourth or fifth level) in 7 of 10 of the trees. We can estimate the overall effect of changing the gender label:

$$\Delta = \mathbb{E}_{x\sim\mathcal{P}}[f(x) \mid \text{male}] - \mathbb{E}_{x\sim\mathcal{P}}[f(x) \mid \text{female}].$$

When gender occurs, Δ is between 0.31 and 0.70 grade points (average 0.49) out of 20 total grade points. For the remaining models, Δ is between 0.11 and 0.32 (average 0.25). Thus, the extracted tree includes gender when f has a relatively large dependence on gender.

Furthermore, because T approximates f, we can use it to identify a subgroup of students where f has particularly strong dependence on gender. In particular, points that flow to the internal node N of T branching on gender are a subset of inputs whose label $T(x) \in \mathcal{Y}$ is determined in part by gender. We can use T to measure the dependence on gender within this subset:

$$\Delta_N = \mathbb{E}_{x\sim\mathcal{P}}[f(x) \mid C_{N_L}] - \mathbb{E}_{x\sim\mathcal{P}}[f(x) \mid C_{N_R}],$$

where N_L and N_R are the left and right children of N.

Also, we can estimate the fraction of students in this subset using the test set, i.e., $P = \sum_{x\in X_{\text{test}}} \mathbb{I}[x \in \mathcal{F}(C_N)]$. Finally, $P \cdot \Delta_N/\Delta$ measures the fraction of the overall dependence of f on gender that is accounted for by the subtree rooted at N. For models where gender occurs in the extracted tree, the subgroup effect size Δ_N ranged from 0.44 to 0.77 grade points, and the estimated fraction of students in this subroup ranged from 18.3% to 39.1%. The two trees that had the largest effect size had

Description of Problem Instance					Absolute			Relative	
Dataset	Task	Samples	Features	Model	f	T	T_{base}	T	T_{base}
breast cancer [31]	classification	569	32	random forest	0.966	**0.942**	0.934	**0.957**	0.945
student grade [9]	regression	382	33	random forest	4.47	**4.70**	5.10	**0.40**	0.64
wine origin [14]	classification	178	13	random forest	0.981	**0.925**	0.890	**0.938**	0.890
wine origin [14]	classification	178	13	neural net	0.795	**0.755**	0.751	**0.913**	0.905
cartpole [5]	reinforcement learning	100	4	control policy	200.0	**190.0**	35.6	**86.8%**	83.8%

Table 1: Comparison of the decision tree T extracted by our algorithm to the one T_{base} extracted by the baseline. We show absolute performance on ground truth and performance relative to the model f. For classification (resp., regression), performance is F_1 score (resp., MSE) on the test set. For reinforcement learning, it is accuracy on the test set for relative performance, and estimated reward using the decision tree as the policy for absolute performance. We bold the higher score between T and T_{base}.

Δ_N of 0.77 and 0.43, resp., and P of 39.1% and 35.7%, resp. The identified subgroup accounted for 67.3% and 65.6% of the total effect of gender, resp.

Having identified a subgroup of students likely to be adversely affected, the user might be able to train a better model specifically for this subgroup. In 5 of the 7 extracted trees where gender occurs, the affected students were students with low grades, in particular, the 27% of students who scored fewer than 8.5 points in another class. This fine-grained understanding of f relies on the extracted model, and cannot be obtained using feature importance metrics alone.

Comparing models. We can use the extracted decision trees to compare different models trained on the same dataset, and gain insight into why some models perform better than others. For example, random forests trained on the wine origin dataset performed very well, all achieving an F_1 score of at least 0.961. In contrast, the performance of the neural nets was bimodal—5 had F_1 score of at least 0.955, and the remaining had an F_1 score of at most 0.741.

We examined the top 3 layers of the extracted decision trees T, and made two observations. First, occurrence of the feature "chlorides" in T was almost perfectly correlated with poor performance of the neural nets. This feature occured in only one of the 10 trees extracted from random forests, and in none of the trees extracted from high performing neural nets. A weaker observation was the branching of T on the feature "alcohol", which is a very important feature—it is the top branch for all but one of the 20 extracted decision trees. For the high performing models, the branch threshold t tended to be higher (749.8 to 999.6) than those for the poorly performing models (574.4 to 837.3). The latter observation relies on having an extracted model—feature influence metrics alone are insufficient.

Understanding control policies. We can use the extracted decision tree to understand a control policy. For example, we extracted a decision tree of size $k = 7$ from the cartpole control policy. While its estimated reward of 152.3 is lower than for larger trees, it captures a significant fraction of the policy behavior. The tree says to move the cart to the right exactly when

$$(\text{pole velocity} \geq -0.286) \wedge (\text{pole angle} \geq -0.071),$$

where the pole velocity is in $[-2.0, 2.0]$ and the pole angle is in $[-0.5, 0.5]$. In other words, move the cart to the right exactly when the pole is already on the right relative to the cart, and the pole is also moving toward the left (or more precisely, not moving fast enough toward the right). This policy is asymmetric, focusing on states where the cart is moving to the left. Examining an animation of simulation, this bias occurs because the cart initially moves toward the left, so the portion of the state space where the cart is moving toward the right is relatively unexplored.

4 CONCLUSIONS

We have proposed model extraction as an approach for interpreting blackbox models, and shown how it can be used to interpret a variety of different kinds of models. Important directions for future work include devising algorithms for model extraction using more expressive input distributions, and developing new ways to gain insight from the extracted decision trees.

REFERENCES

[1] Openai cartpole-v0 environment. https://gym.openai.com/envs/CartPole-v0. Accessed: 2017-05-18.

[2] Uci machine learning repository. http://archive.ics.uci.edu/ml. Accessed: 2017-05-18.

[3] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists for categorical data. *KDD*, 2017.

[4] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[5] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[6] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[7] Rich Caruana, Yin Lou, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Citeseer, 2012.

[8] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730. ACM, 2015.

[9] Paulo Cortez and Alice Maria Gonçalves Silva. Using data mining to predict secondary school student performance. 2008.

[10] Houtao Deng. Interpreting tree ensembles with intrees. *arXiv preprint arXiv:1408.5456*, 2014.

[11] Berkeley J Dietvorst, Joseph P Simmons, and Cade Massey. Algorithm aversion: People erroneously avoid algorithms after seeing them err. *Journal of Experimental Psychology: General*, 144(1):114, 2015.

[12] Finale Doshi-Velez and Been Kim. A roadmap for a rigorous science of interpretability. *arXiv preprint arXiv:1702.08608*, 2017.

[13] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 214–226. ACM, 2012.

[14] M Forina et al. An extendible package for data exploration, classification and correlation. *Institute of Pharmaceutical and Food Analisys and Technologies, Via Brigata Salerno*, 16147, 1991.

[15] Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, pages 3315–3323, 2016.

[16] Jongbin Jung, Connor Concannon, Ravi Shroff, Sharad Goel, and Daniel G Goldstein. Simple rules for complex decisions. *arXiv preprint arXiv:1702.04690*, 2017.

[17] Jon Kleinberg, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. Human decisions and machine predictions. Technical report, National Bureau of Economic Research, 2017.

[18] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

[19] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001.

[20] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[21] Judea Pearl. *Causality*. Cambridge university press, 2009.

[22] Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 560–568. ACM, 2008.

[23] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[24] Cynthia Rudin. Algorithms for interpretable machine learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1519–1519. ACM, 2014.

[25] Selim Temizer, Mykel Kochenderfer, Leslie Kaelbling, Tomas Lozano-Pérez, and James Kuchar. Collision avoidance for unmanned aircraft using markov decision processes. In *AIAA guidance, navigation, and control conference*, page 8040, 2010.

[26] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391, 2016.

[27] Gilmer Valdes, José Marcio Luna, Eric Eaton, Charles B Simone, et al. Mediboost: a patient stratification tool for interpretable decision making in the era of precision medicine. *Scientific Reports*, 6, 2016.

[28] Anneleen Van Assche and Hendrik Blockeel. Seeing the forest through the trees. In *International Conference on Inductive Logic Programming*, pages 269–279. Springer, 2007.

[29] Gilles Vandewiele, Olivier Janssens, Femke Ongenae, Filip De Turck, and Sofie Van Hoecke. Genesim: genetic extraction of a single, interpretable model. *arXiv preprint arXiv:1611.05722*, 2016.

[30] Fulton Wang and Cynthia Rudin. Falling rule lists. In *AISTATS*, 2015.

[31] William H Wolberg and Olvi L Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the national academy of sciences*, 87(23):9193–9196, 1990.