Human-in-the-Loop Interpretability Prior

Isaac Lage
Department of Computer Science
Harvard University
isaaclage@g.harvard.edu

Andrew Slavin Ross
Department of Computer Science
Harvard University
andrew_ross@g.harvard.edu

Been Kim
Google Brain
beenkim@google.com

Samuel J. Gershman
Department of Psychology
Harvard University
gershman@fas.harvard.edu

Finale Doshi-Velez
Department of Computer Science
Harvard University
finale@seas.harvard.edu

Abstract

We often desire our models to be interpretable as well as accurate. Prior work on optimizing models for interpretability has relied on easy-to-quantify proxies for interpretability, such as sparsity or the number of operations required. In this work, we optimize for interpretability by *directly* including humans in the optimization loop. We develop an algorithm that minimizes the number of user studies to find models that are both predictive and interpretable and demonstrate our approach on several data sets. Our human subjects results show trends towards different proxy notions of interpretability on different datasets, which suggests that different proxies are preferred on different tasks.

1 Introduction

Understanding machine learning models can help people discover confounders in their training data, and dangerous associations or new scientific insights learned by their models [3, 9, 15]. This means that we can encourage the models we learn to be safer and more useful to us by effectively incorporating interpretability into our training objectives. But interpretability depends on both the subjective experience of human users and the specific features of a downstream application, which makes it difficult to incorporate into computational learning methods.

Human-interpretability can be achieved by learning models that are inherently easier to explain or by developing more sophisticated explanation methods; we focus on the first problem. This can be solved with one of two broad approaches. The first *defines* certain classes of models as inherently interpretable. Well known examples include decision trees [9], generalized additive models [3], and decision sets [13]. The second approach identifies some *proxy* that (presumably) makes a model interpretable and then optimizes that proxy. Examples of this second approach include optimizing linear models to be sparse [30], optimizing functions to be monotone [1], or optimizing neural networks to be easily explained by decision trees [34].

In many cases, the optimization of a property can be viewed as placing a prior over models and solving for a MAP solution of the following form:

$$\max_{M \in \mathcal{M}} p(X|M)p(M) \tag{1}$$

where $\mathcal{M}$ is a family of models, X is the data, $p(X|M)$ is the likelihood, and $p(M)$ is a prior on the model that encourages it to share some aspect of our inductive biases. Two examples of biases include the interpretation of the L1 penalty on logistic regression as a Laplace prior on the weights and the

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

class of norms described in Bach [2] that induce various kinds of structured sparsity. Generally, if we have a functional form for $p(M)$, we can apply a variety of optimization techniques to find the MAP solution. Placing an interpretability bias on a class of models through $p(M)$ allows us to search for interpretable models in more expressive function classes.

Optimizing for interpretability in this way relies heavily on the assumption that we can quantify the subjective notion of human interpretability with some functional form $p(M)$. Specifying this functional form might be quite challenging. In this work, we *directly* estimate the interpretability prior $p(M)$ from human-subject feedback. Optimizing this more direct measure of interpretability can give us models more suited to a task at hand than more accurately optimizing an imperfect proxy.

Since measuring $p(M)$ for each model M has a high cost—requiring a user study—we develop a cost-effective approach that initially identifies models M with high likelihood $p(X|M)$, then uses model-based optimization to identify an approximate MAP solution from that set with few queries to $p(M)$. We find that different proxies for interpretability prefer different models, and that our approach can optimize all of these proxies. Our human subjects results suggest that we can optimize for human-interpretability preferences.

2 Related Work

Learning interpretable models with proxies. Many approaches to learning interpretable models optimize proxies that can be computed directly from the model. Examples include decision tree depth [9], number of integer regression coefficients [31], amount of overlap between decision rules [13], and different kinds of sparsity penalties in neural networks [10, 25]. In some cases, optimizing a proxy can be viewed as MAP estimation under an interpretability-encouraging prior [30, 2]. These proxy-based approaches assume that it is possible to formulate a notion of interpretability that is a computational property of the model, and that we know a priori what that property is. Lavrac [14] shows a case where doctors prefer longer decision trees over shorter ones, which suggests that these proxies do not fully capture what it means for a model to be interpretable in all contexts. Through our approach, we place an interpretability-encouraging prior on arbitrary classes of models that depends directly on human preferences.

Learning from human feedback. Since interpretability is difficult to quantify mathematically, Doshi-Velez and Kim [8] argue that evaluating it well requires a user study. Many works in interpretable machine learning have user studies: some advance the science of interpretability by testing the effect of explanation factors on human performance on interpretability-related tasks [21, 19] while others compare the interpretability of two classes of models through A/B tests [13, 11]. More broadly, there exist many studies about situations in which human preferences are hard to articulate as a computational property and must be learned directly from human data. Examples include kernel learning [29, 32], preference based reinforcement learning [33, 5] and human based genetic algorithms [12]. Our work resembles human computation algorithms [16] applied to user studies for interpretability as we use the user studies to *optimize* for interpretability instead of just comparing a model to a baseline.

Model-based Optimization Many techniques have been developed to efficiently characterize functions in few evaluations when each evaluation is expensive. The field of Bayesian experimental design [4] optimizes which experiments to perform according to a notion of which information matters. In some cases, the intent is to characterize the entire function space completely [35, 17], and in other cases, the intent is to find an optimum [28, 27]. We are interested in this second case. Snoek *et al.* [27] optimize the hyperparameters of a neural network in a problem setup similar to ours. For them, evaluating the likelihood is expensive because it requires training a network, while in our case, evaluating the prior is expensive because it requires a user study. We use a similar set of techniques since, in both cases, evaluating the posterior is expensive.

3 Framework and Modeling Considerations

Our high-level goal is to find a model M that maximizes $p(M|X) \propto p(X|M)p(M)$ where $p(M)$ is a measure of human interpretability. We assume that computation is relatively inexpensive, and thus computing and optimizing with respect to the likelihood $p(X|M)$ is significantly less expensive

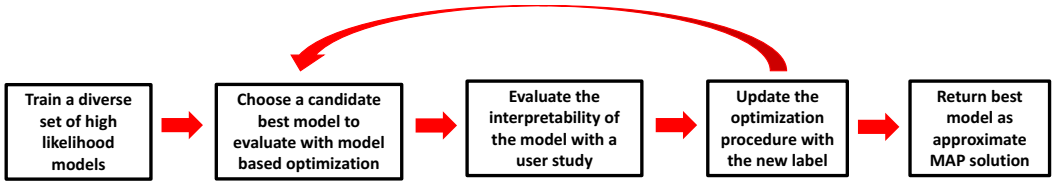


Figure 1: High level overview of the pipeline

than evaluating the prior $p(M)$, which requires a user study. Our strategy will be to first identify a large, diverse collection of models M with large likelihood $p(X|M)$, that is, models that explain the data well. This task can be completed without user studies. Next, we will search amongst these models to identify those that also have large prior $p(M)$. Specifically, to limit the number of user studies required, we will use a model-based optimization approach [28] to identify which models M to evaluate. Figure 1 depicts the steps in the pipeline. Below, we outline how we define the likelihood $p(X|M)$ and the prior $p(M)$; in Section 4 we define our process for approximate MAP inference.

3.1 Likelihood

In many domains, experts desire a model that achieves some performance threshold (and amongst those, may prefer one that is most interpretable). To model this notion of a performance threshold, we use the soft insensitive loss function (SILF)-based likelihood [6, 18]. The likelihood takes the form of

$$p(X|M) = \frac{1}{Z} e^{(-C \times \texttt{SILF}_{\epsilon,\beta}(1-\texttt{accuracy}(X,M)))}$$

where accuracy(X,M) is the accuracy of model M on data X and $\texttt{SILF}_{\epsilon,\beta}(y)$ is given by

$$\texttt{SILF}_{\epsilon,\beta}(y) = \begin{cases} 0, & 0 \leq y \leq (1-\beta)\epsilon \\ \frac{(y-(1-\beta)\epsilon)^2}{4\beta\epsilon}, & (1-\beta)\epsilon \leq y \leq (1+\beta)\epsilon \\ y-\epsilon, & y \geq (1+\beta)\epsilon \end{cases}$$

which effectively defines a model as having high likelihood if its accuracy is greater than $1-(1-\beta)\epsilon$.

In practice, we choose the threshold $1-(1-\beta)\epsilon$ to be equal to an accuracy threshold placed on the validation performance of our classification tasks, and only consider models that perform above that threshold. (Note that with this formulation, accuracy can be replaced with any domain specific notion of a high-quality model without modifying our approach.)

3.2 A Prior for Interpretable Models

Some model classes are generally amenable to human inspection (e.g. decision trees, rule lists, decision sets [9, 13]; unlike neural networks), but within those model classes, there likely still exist some models that are easier for humans to utilize than others (e.g. shorter decision trees rather than longer ones [24], or decision sets with fewer overlaps [13]). We want our model prior $p(M)$ to reflect this more nuanced view of interpretability.

We consider a prior of the form:

$$p(M) \propto \int_x \texttt{HIS}(x, M) p(x) dx \tag{2}$$

In our experiments, we will define $\texttt{HIS}(x, M)$ (human-interpretability-score) as:

$$\texttt{HIS}(x, M) = \begin{cases} 0, & \texttt{mean-RT}(x, M) > \texttt{max-RT} \\ \texttt{max-RT} - \texttt{mean-RT}(x, M), & \texttt{mean-RT}(x, M) \leq \texttt{max-RT} \end{cases} \tag{3}$$

where $\texttt{mean-RT}(x, M)$ (the mean response time given data point x and model M) measures how long it takes users to predict the label assigned to a data point x by the model M, and max-RT is a cap on response time that is set to a large enough value to catch all legitimate points and exclude outliers. The choice of measuring the time it takes to predict the model's label follows Doshi-Velez and Kim [8], which suggests this *simulation* proxy as a measure of interpretability when no downstream task has been defined yet; but any domain-specific task and metric could be substituted into our pipeline including error detection or cooperative decision-making.

3.3 A Prior for Arbitrary Models

In the interpretable model case, we can give a human subject a model M and ask them questions about it; in the general case, models may be too complex for this approach to be feasible. In order to determine the interpretability of complex models like neural networks, we follow the approach in Ribeiro *et al.* [23], and construct a simple *local* model for each point x by sampling perturbations of x and training a simple model to mimic the predictions of M in this local region. We denote this $\texttt{local-proxy}(M, x))$.

We change the prior in Equation 2 to reflect that we evaluate the HIS with the local proxy rather than the entire model:

$$p(M) \propto \int_x \texttt{HIS}(x, \texttt{local-proxy}(M, x))p(x)dx \tag{4}$$

We describe computational considerations for this more complex situation in Section 4.

4 Inference

Our goal is to find the MAP solution from Equation 1. Our overall approach will be to find a collection of models with high likelihood $p(X|M)$ and then perform model-based optimization [28] to identify which priors $p(M)$ to evaluate via user studies. Below, we describe each of the three main aspects of the inference: identifying models with large likelihoods $p(X|M)$, evaluating $p(M)$ via user studies, and using model-based optimization to determine which $p(M)$ to evaluate. The model from our set with the best $p(X|M)p(M)$ is our approximation to the MAP solution.

4.1 Identifying models with high likelihood $p(X|M)$

In the model-finding phase, our goal is to create a diverse set of models with large likelihoods $p(X|M)$ in the hopes that some will have large prior value $p(M)$ and thus allow us to identify the approximate MAP solution. For simpler model classes, such as decision trees, we find these solutions via running multiple restarts with different hyperparameter settings and rejecting those that do not meet our accuracy threshold. For neural networks, we jointly optimize a collection of predictive neural networks with different input gradient patterns (as a proxy for creating a diverse collection) [26].

4.2 Computing the prior $p(M)$

Human-Interpretable Model Classes. For any model M and data point x, a user study is required for every evaluation of $\texttt{HIS}(x, M)$. Since it is infeasible to perform a user study for every value of x for even a single model M, we approximate the integral in Equation 2 via a collection of samples:

$$\begin{aligned} p(M) &\propto \int_x \texttt{HIS}(x, M)p(x)dx \\ &\approx \frac{1}{N} \sum_{x_n \sim p(x)} \texttt{HIS}(x_n, M) \end{aligned}$$

In practice, we use the empirical distribution over the inputs x as the prior $p(x)$.

Arbitrary Model Classes. If the model M is not itself human-interpretable, we define $p(M)$ to be the integral over $\texttt{HIS}(x, \texttt{local-proxy}(M, x))$ where $\texttt{local-proxy}(M, x)$ locally approximates M around x (Equation 4). As before, evaluating $\texttt{HIS}(x, \texttt{local-proxy}(M, x))$ requires a user study; however, now we must determine a procedure for generating the local approximations $\texttt{local-proxy}(M, x)$.

We generate these local approximations via a procedure akin to Ribeiro *et al.* [23]: for any x, we sample a set of perturbations x' around x, compute the outputs of model M for each of those x', and then fit a human-interpretable model (e.g. a decision-tree) to those data.

We note that these local models will only be nontrivial if the data point x is in the vicinity of a decision boundary; if not, we will not succeed in fitting a local model. Let $B(M)$ denote the set of inputs x that are near the decision boundary of M. Since we defined HIS to equal max-RT when

$\texttt{mean-RT}(x, M)$ is 0 as it does when no local model can be fit (see Equation 3), we can compute the integral in Equation 4 more intelligently by only seeking user input for samples near the model's decision boundary:

$$
\begin{aligned}
p(M) &\propto \int_x \texttt{HIS}(x, \texttt{local-proxy}(M, x))p(x)dx \qquad (5)\\
&= \left(\int_{x \in B(M)} p(x)dx\right) \cdot \left(\int_{x \in B(M)} \texttt{HIS}(x, \texttt{local-proxy}(M, x))\tilde{p}(x)dx\right)\\
&\quad + \left(\int_{x \notin B(M)} p(x)dx\right) \cdot \texttt{max-RT}\\
&\approx \left(\frac{1}{N'} \sum_{x_{n'} \sim p(x)} \mathbb{I}(x \in B(M))\right) \cdot \left(\frac{1}{N} \sum_{x_n \sim \tilde{p}(x)} \texttt{HIS}(x_n, M)\right)\\
&\quad + \left(\frac{1}{N'} \sum_{x_{n'} \sim p(x)} \mathbb{I}(x \notin B(M))\right) \cdot \texttt{max-RT}
\end{aligned}
$$

where $\tilde{p}(x) = p(x) / \int_{x \in B(M)} p(x)dx$. The first term (the volume of $p(x)$ in $B(M)$), can be approximated without any user studies by attempting to fit local models for each point in x (or a subsample of points). We detail how we fit local explanations and define the boundary in Appendix C.

4.3 Model-based Optimization of the MAP Objective

The first stage of our optimization procedure gives us a collection of models $\{M_1, ..., M_K\}$ with high likelihood $p(X|M)$. Our goal is to identify the model M_k in this set that is the approximate MAP, that is, maximizes $p(X|M)p(M)$, with as few evaluations of $p(M)$ as possible.

Let L be the set of all labeled models M, that is, the set of models for which we have evaluated $p(M)$. We estimate the values (and uncertainties) for the remaining unlabeled models—set U—via a Gaussian Process (GP) [22]. (See Appendix A for details about our model-similarity kernel.) Following Srinivas *et al.* [28], we use the GP upper confidence bound acquisition function to choose among unlabeled models $M \in U$ that are likely to have large $p(M)$ (this is equivalent to using the lower confidence bound to minimize response time):

$$
\begin{aligned}
a_{LCB}(M; L, \theta) &= \mu(M; L, \theta) - \kappa\sigma(M; L, \theta)\\
M_{\texttt{next}} &= \arg\min_{M \in U} a_{LCB}(M; L, \theta)
\end{aligned}
$$

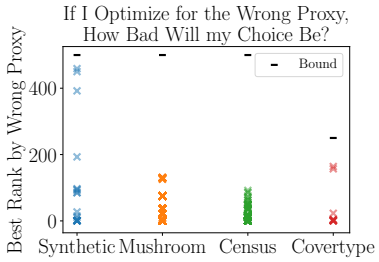
where κ is a hyperparameter that can be tuned, θ are parameters of the GP, μ is the GP mean function, and σ is the GP variance. (We find $\kappa = 1$ works well in practice.)

5 Experimental Setup

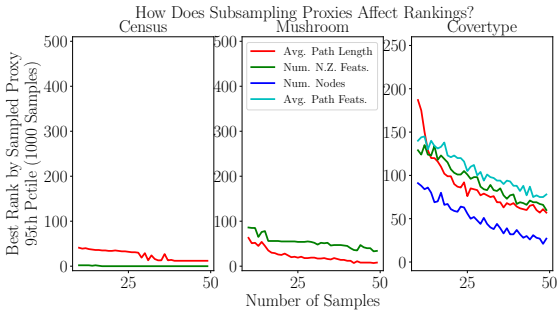
In this section, we provide details for applying our approach to four datasets. Our results are in Section 6.

Datasets and Training Details We test our approach on a synthetic dataset as well as the mushroom, census income, and covertype datasets from the UCI database [7]. All features are preprocessed by z-scoring continuous features and one-hot encoding categorical features. We also balance the classes of the first three datasets by subsampling the more common class. (The sizes reported are after class balancing. We do not include a test set because we do not report held-out accuracy.)

- *Synthetic* ($N = 90,000$, $D = 6$, continuous). We build a data set with two noise dimensions, two dimensions that enable a lower-accuracy, interpretable explanation, and two dimensions that enable a higher-accuracy, less interpretable explanation. We use an 80%-20% train-validate split. (See Figure 5 in the Appendix.)
- *Mushroom* ($N = 8,000$, $D = 22$ categorical with 126 distinct values). The goal is to predict if the mushroom is edible or poisonous. We use an 80%-20% train-validate split.



(a) For each pair of proxies (A, B) for interpretability, we first identify the best model if we only care about proxy A, then compute its rank if we now care about proxy B. This simulates the setting where we optimize for proxy A, but proxy B is the true HIS. This value for each pair of proxies is plotted with an $\times$. The large ranking value indicates that sometimes proxies disagree on which models are good.



(b) Rank of the best model(s) by each proxy across multiple samples of data points ('NZ' denotes non-zero). The lines dropping below the high values in Figure 2a indicate that computing the right proxy on a sample of points is better than computing the wrong proxy accurately. This benefit occurs at a human-accessible number of samples for all datasets and models, but it takes more samples for neural networks on Covertype than the others.

Figure 2: Determining interpretability on a few points is better than using the wrong proxy.

- *Census* ($N = 20,000$, $D = 13$—6 continuous, 7 categorical with 83 distinct values). The goal is to predict if people make more than $\$50,000$/year. We use their 60%-40% train-validate split.
- *Covertype* ($N = 580,000$, $D = 12$—10 continuous, 2 categorical with 44 distinct values). The goal is to predict tree cover type. We use a 75%-25% train-validate split.

Our experiments include two classes of models: decision trees and neural networks. We train decision trees for the simpler synthetic, mushroom and census datasets and neural networks for the more complex covertype dataset. Details of our model training procedure (that is, identifying models with high predictive accuracy) are in Appendix B. The covertype dataset, because it is modeled by a neural network, also needs a strategy for producing local explanations; we describe our parameter choices as well as provide a detailed sensitivity analysis to these choices in Appendix C.

Proxies for Interpretability An important question is whether currently used proxies for interpretability, such as sparsity or number of nodes in a path, correspond to some HIS. In the following we will use four different interpretability proxies to demonstrate the ability of our pipeline to identify models that are best under these different proxies, simulating the case where we have a ground truth measure of HIS. We show that (a) different proxies favor different models and (b) how these proxies correspond to the results of our user studies.

The interpretability proxies we will use are: mean path length, mean number of distinct features in a path, number of nodes, and number of nonzero features. The first two are local to a specific input x while the last two are global model properties (although these will be properties of local proxy models for neural networks). These proxies includes notions of tree depth [24] and sparsity [15, 21]. We compute the proxies based on a sample of 1000 points from the validation set (the same set of points is used across models).

Human Experiments In our human subjects experiments, we quantify $\texttt{HIS}(x, M)$ for a data point x and a model M as a function of the time it takes a user to simulate the label for x with M. We extend this to the locally interpretable case by simulating the label according to $\texttt{local-proxy}(x, M)$. We refer to the model itself as the explanation in the globally interpretable case, and the local model as the explanation in the locally interpretable case. Our experiments are closely based on those in Narayanan *et al.* [19]. We provide users with a list of feature values for features used in the explanation and a graphical depiction of the explanation, and ask them to identify the correct prediction. Figure 7a depicts our interface. These experiments were reviewed and approved by our institution's IRB. Details of the experiments we conducted with machine learning researchers and details and results of a pilot study [not used in this paper] conducted using Amazon Turk (which had very high variance) are in Appendix D.

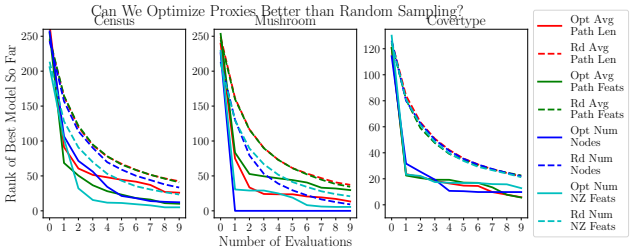


Figure 3: We ran random restarts of the pipeline with all datasets and proxies–denoted 'opt' (randomness from choice of start), and compared to randomly sampling the same number of models–denoted 'rd' (we account for models with the same score by computing the lowest rank of any model with that score). 'NZ feats' denotes non-zero features and 'path feats' denotes the number of features used in a path. The fact that the solid lines stay below the corresponding dotted lines indicates that we do better than random guessing.

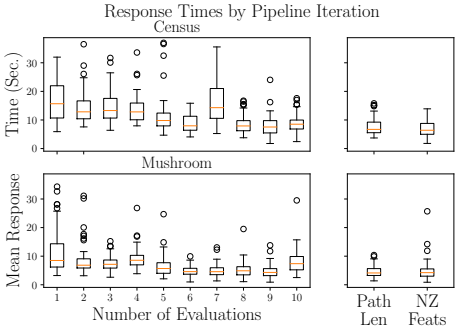
6 Experimental Results

Optimizing different automatic proxies results in different models. For each dataset, we run simulations to test what happens when the optimized measure of interpretability does not match the true `HIS`. We do this by computing the best model by one proxy–our simulated `HIS`, then identifying what *rank* it would have had among the collection of models if one of the other proxies–our optimized interpretability measure–had been used. A rank of 0 indicates that the model identified as the best by one proxy is the same as the best model for the second proxy; more generally a rank of r indicates that the best model by one proxy is the rth-best model under the second proxy. Figure 2a shows that choosing the wrong proxy can seriously mis-rank the true best model. This suggests that it is not a good idea to optimize an arbitrary proxy for interpretability in the hopes that the resulting model will be interpretable according to the truly relevant measure. Figure 2a also shows that the synthetic dataset has a very different distribution of proxy mis-rankings than any of the real datasets in our experiments. This suggests that it is hard to design synthetic datasets that capture the relevant notions of interpretability since, by assumption, we do not know what these are.

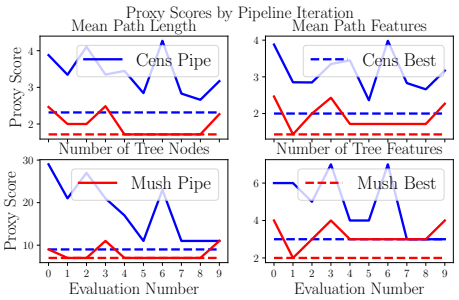
Computing the right proxy on a small sample of data points is better than computing the wrong proxy. For each dataset, we run simulations to test what happens when we optimize the true `HIS` computed on only a small sample of points–the size limitation comes from limited human cognitive capacity. As in the previous experiment, we compute the best model by one proxy–our simulated `HIS`. We then identify what rank it would have had among the collection of models if the same proxy had been computed on a small sample of data points. Figure 2 shows that computing the right proxy on a small sample of data points can do better than computing the wrong proxy. This holds across datasets and models. This suggests that it may be better to find interpretable models by asking people to examine the interpretability of a small number of examples—which will result in noisy measurements of the true quantity of interest—rather than by accurately optimizing a different proxy.

Our model-based optimization approach can learn human-interpretable models that correspond to a variety of different proxies on globally and locally interpretable models. We run our pipeline 100 times for 10 iterations with each proxy as the signal (the randomness comes from the choice of starting point), and compare to 1000 random draws of 10 models. We account for multiple models with the same score by computing the lowest rank for any model with the same score as the model we sample. Figure 3 shows that across all three datasets, and across all four proxies, we do better than randomly sampling models to evaluate.

Our pipeline finds models with lower response times and lower scores across all four proxies when we run it with human feedback. We run our pipeline for 10 iterations on the census and mushrooms datasets with human response time as the signal. We recruited a group of machine learning researchers who took all quizzes in a single run of the pipeline, with models iteratively chosen from our model-based optimization. Figure 4a shows the distributions of mean response times decreasing as we evaluate more models. (In Figure 7b in Appendix D we demonstrate that increases



(a) We computed response times for each iteration of the pipeline on two datasets. Each data point is the mean response time for a single user. In both experiments, we see the mean response times decrease as we evaluate more models. We reach times comparable to those of the best proxy models. The last 2 models are our baselines ('NZ feats' denotes non-zero features).



(b) We computed the proxy scores for the model evaluated at each iteration of the pipeline. On the mushroom dataset, our approach converges to models with the fewest nodes and shortest paths, and on the census dataset, it converges to models with the fewest features. 'Mush' denotes the mushroom dataset and 'Cens' denotes the census dataset.

Figure 4: Human subjects pipeline results show a trend towards interpretability.

in speed from repeatedly doing the task are small compared to the differences we see in Figure 4a; these are real improvements in response time.)

On different datasets, our pipeline converges to different proxies. In the human subjects experiments above, we tracked the proxy scores of each model we evaluated. Figure 4b shows a decrease in proxy scores that corresponds to the decrease in response times in Figure 4a (our approach did *not* have access to these proxy scores). On the mushroom dataset, our approach converged to a model with the fewest nodes and the shortest paths, while on the census dataset, it converged to a model with the fewest features. This suggests that, for different datasets, different notions of interpretability are important to users.

7 Discussion and Conclusion

We presented an approach to efficiently optimize models for human-interpretability (alongside prediction) by directly including humans in the optimization loop. Our experiments showed that, across several datasets, several reasonable proxies for interpretability identify different models as the most interpretable; all proxies do not lead to the same solution. Our pipeline was able to efficiently identify the model that humans found most expedient for forward simulation. While the human-selected models often corresponded to some known proxy for interpretability, which proxy varied across datasets, suggesting the proxies may be a good starting point but are not the full story when it comes to finding human-interpretable models.

That said, the direct human-in-the-loop optimization has its challenges. In our initial pilot studies [not used in this paper] with Amazon Mechanical Turk (Appendix D), we found that the variance among subjects was simply too large to make the optimization cost-effective (especially with the between-subjects model that makes sense for Amazon Mechanical Turk). In contrast, our smaller but longer within-subjects studies had lower variance with a smaller number of subjects. This observation, and the importance of downstream tasks for defining interpretability suggest that interpretability studies should be conducted with the people who will use the models (who we can expect to be more familiar with the task and more patient).

The many exciting directions for future work include exploring ways to efficiently allocate the human computation to minimize the variance of our estimates $p(M)$ via intelligently choosing which inputs x to evaluate and structuring these long, sequential experiments to be more engaging; and further refining our model kernels to capture more nuanced notions of human-interpretability, particularly across model classes. Optimizing models to be human-interpretable will always require user studies, but with intelligent optimization approaches, we can reduce the number of studies required and thus cost-effectively identify human-interpretable models.

Acknowledgments IL acknowledges support from NIH 5T32LM012411-02. All authors acknowledge support from the Google Faculty Research Award and the Harvard Dean's Competitive Fund. All authors thank Emily Chen and Jeffrey He for their support with the experimental interface, and Weiwei Pan and the Harvard DTaK group for many helpful discussions and insights.

References

[1] Eric E. Altendorf, Angelo C. Restificar, and Thomas G. Dietterich. Learning from sparse data by exploiting monotonicity constraints. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 18–26, Arlington, Virginia, United States, 2005. AUAI Press.

[2] Francis R. Bach. Structured sparsity-inducing norms through submodular functions. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 118–126. Curran Associates, Inc., 2010.

[3] 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.

[4] Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statist. Sci.*, 10(3):273–304, 08 1995.

[5] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4299–4307. Curran Associates, Inc., 2017.

[6] Wei Chu, S. S. Keerthi, and Chong Jin Ong. Bayesian support vector regression using a unified loss function. *IEEE Transactions on Neural Networks*, 15(1):29–44, Jan 2004.

[7] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[8] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv*, 2017.

[9] Alex A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, 15(1):1–10, March 2014.

[10] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

[11] Been Kim, Cynthia Rudin, and Julie A Shah. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1952–1960. Curran Associates, Inc., 2014.

[12] Alex Kosorukoff. Human based genetic algorithm. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 5, 05 2001.

[13] Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1675–1684, New York, NY, USA, 2016. ACM.

[14] Nada Lavrac. Selected techniques for data mining in medicine. *Artificial Intelligence in Medicine*, 16(1):3 – 23, 1999. Data Mining Techniques and Applications in Medicine.

[15] Zachary Chase Lipton. The mythos of model interpretability. *CoRR*, abs/1606.03490, 2016.

[16] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. Turkit: Human computation algorithms on mechanical turk. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 57–66, New York, NY, USA, 2010. ACM.

[17] Yifei Ma, Roman Garnett, and Jeff G. Schneider. Submodularity in batch active learning and survey problems on gaussian random fields. *CoRR*, abs/1209.3694, 2012.

[18] Muhammad A. Masood and Finale Doshi-Velez. A particle-based variational approach to bayesian non-negative matrix factorization. *arXiv*, 2018.

[19] Menaka Narayanan, Emily, Chen, Jeffrey He, Been Kim, Sam Gershman, and Finale Doshi-Velez. How do Humans Understand Explanations from Machine Learning Systems? An Evaluation of the Human-Interpretability of Explanation. *ArXiv e-prints*, February 2018.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] Forough Poursabzi-Sangdeh, Daniel G. Goldstein, Jake M. Hofman, Jennifer Wortman Vaughan, and Hanna M. Wallach. Manipulating and measuring model interpretability. *CoRR*, abs/1802.07810, 2018.

[22] Carl Edward Rasmussen. *Gaussian processes for machine learning*. MIT Press, 2006.

[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*, KDD '16, pages 1135–1144, New York, NY, USA, 2016. ACM.

[24] Lior Rokach and Oded Maimon. *Introduction to Decision Trees*, chapter Chapter 1, pages 1–16. WORLD SCIENTIFIC, 2nd edition, 2014.

[25] Andrew Ross, Isaac Lage, and Finale Doshi-Velez. The neural lasso: Local linear sparsity for interpretable explanations. In *Workshop on Transparent and Interpretable Machine Learning in Safety Critical Environments, 31st Conference on Neural Information Processing Systems*, 2017. `https://goo.gl/TwRhXo`.

[26] Andrew Ross, Weiwei Pan, and Finale Doshi-Velez. Learning qualitatively diverse and interpretable rules for classification. In *2018 ICML Workshop on Human Interpretability in Machine Learning (WHI 2018)*, 2018.

[27] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012.

[28] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian Process Bandits without Regret: An Experimental Design Approach. Technical Report arXiv:0912.3995, Dec 2009. Comments: 17 pages, 5 figures.

[29] Omer Tamuz, Ce Liu, Serge J. Belongie, Ohad Shamir, and Adam Tauman Kalai. Adaptively learning the crowd kernel. *CoRR*, abs/1105.1033, 2011.

[30] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[31] Berk Ustun and Cynthia Rudin. Optimized risk scores. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1125–1134, New York, NY, USA, 2017. ACM.

[32] Andrew G Wilson, Christoph Dann, Chris Lucas, and Eric P Xing. The human kernel. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2854–2862. Curran Associates, Inc., 2015.

[33] Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017.

[34] Mike Wu, Michael C. Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. Beyond Sparsity: Tree Regularization of Deep Models for Interpretability. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[35] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.

A Similarity Kernel for Models and GP parameters

Model-based optimization requires as input a notion of similarity. We use an RBF kernel between feature importances for decision trees, and between a gradient-based notion of feature importance for neural networks (average magnitude of the normalized input gradients for each class logit).

We use the scikit-learn implementation of Gaussian processes [20]. We set it to normalize y automatically, restart the optimizer 10 times, and add $\alpha = 10^{-7}$ to the diagonal of the kernal at fitting to mitigate numerical issues. We used the default settings for all other hyperparameters, including the RBF kernel (on the model features above) for the covariance function.

B Experimental Details: Identifying a Collection of Predictive Models

We train decision trees for the synthetic, mushroom and census datasets with a test accuracy thresholds of 0.9, 0.95 and 0.8 respectively. On the synthetic dataset, 0.9 is slightly higher than the accuracy we can achieve on the interpretable dimensions. We make this choice to avoid learning the same, simple model over and over again. On the mushroom dataset, we can achieve a validate accuracy of 1 with decision trees, and on the Census dataset, we can achieve a validate accuracy of 0.83 with decision trees. In both cases, we set the accuracy thresholds slightly below these numbers to ensure that we can generate distinct models that meet the accuracy threshold. For each of these, we train 500 models.

To produce a variety of high-performing decision trees, we randomly sample the following hyperparameters: max depth [1-7], minimum number of samples at a leaf [1, 10, 100], max features used in a split [2 - `num_features`], and splitting strategy [best, random]. The first two hyperparameters are chosen to encourage simple solutions, while the last two hyperparameters are chosen to increase the diversity of discovered trees. We use the scikit-learn implementation [20], of decision trees and perform a post-processing step that removes leaf nodes iteratively when it does not decrease accuracy on the validation set (as in Wu *et al.* [34]).

We train neural networks for the covertype dataset with an accuracy threshold of 0.75. We can achieve an accuracy of 0.71 with logistic regression, so we set the threshold slightly above that to justify the use of more complex neural networks. For the neural network models, we randomly sample the following hyperparameters: L1 weight penalty [0, 0.0001, 0.001, 0.01], L2 weight penalty [0, 0.0001, 0.001, 0.01], L1 gradient regularization [0, 0.01], activation function [relu, tanh], architectures [three 100-node layers, two 100-node layers, one 100-node layer, one 25-node layer, one 250-node layer]. These are then jointly trained according to the procedure in Ross *et al.* [26] for 50 epochs for batch size 512 with Adam. (We train between 1 and 4 models simultaneously, another randomly sampled hyperparameter). For this dataset, we train 250 models.

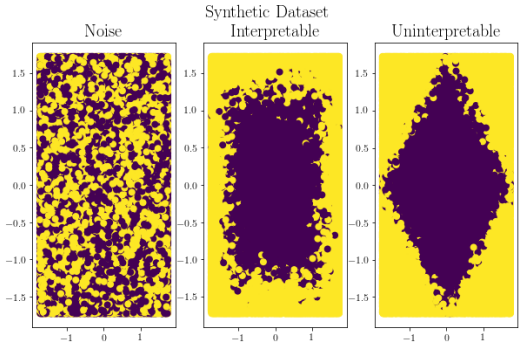
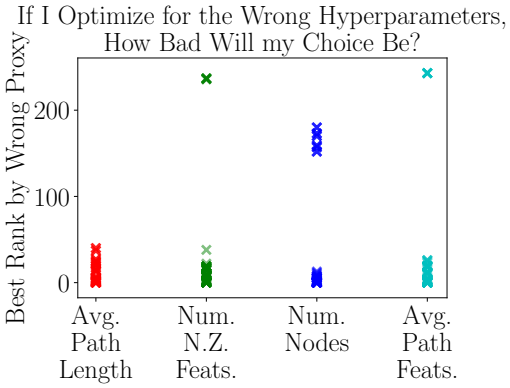


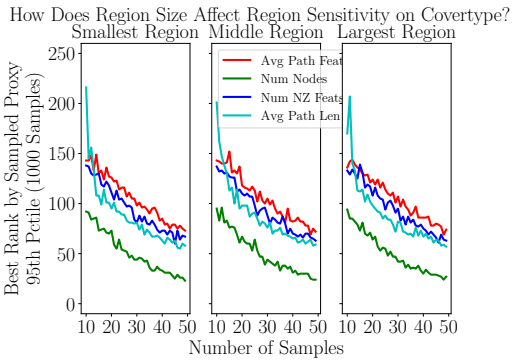
Figure 5: We build a synthetic data set with two noise dimensions, two dimensions that enable a lower-accuracy, interpretable explanation, and two dimensions that enable higher-accuracy, less interpretable explanation. The purple data points are positive and the yellow are negative. Data points were generated for each set of two features independently, then points sharing the same label in all dimensions were randomly concatenated to form the final dataset.

C Experimental Details: Parameters and Sensitivity to Local Region Choices

We can ask humans to perform the simulation task directly using decision trees, but for the neural networks, we must train simple, local models as explanations (we use local decision trees). This procedure requires first sampling a local dataset for each point x we explain. We modify the procedure in Ribeiro *et al.* [23] to sample



(a) We found the best model by each proxy for every setting of the region hyperparameters, and computed its rank by the same proxy for every other setting of the region hyperparameters. Each x corresponds to one of these pairs. The highest values all correspond to the variance scaling factor 0.1. The other two settings of this hyperparameter tend to agree on how to rank neural networks.



(b) We found the best model(s) by each proxy and computed their rank by the same proxy computed on a sample of data points. The comparable values of the lines across all three plots indicate that we need a similar number of samples to robustly rank neural networks for the smallest, middle and largest region settings (we do not include cross pairs).

$10,000$ points x' in a radius around the point x defined by its 20 nearest neighbors by Euclidean distance. We then binarize their predictions $M(x')$ to whether they match $M(x)$ and return the simplest tree we train on this local dataset with accuracy above a threshold on a validate set. We randomly set aside 20% of the sampled points for validation, and use the rest for training. (Note: if we were provided local regions by domain experts, we could use those.)

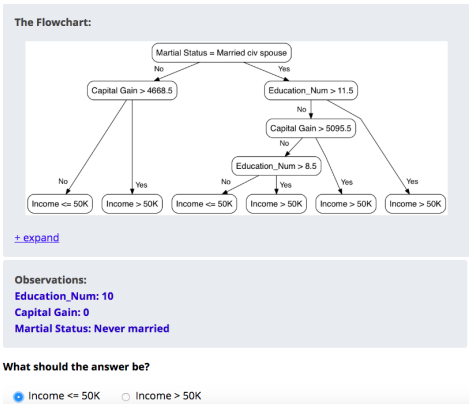
Our procedure for sampling points around some input x uses two hyperparameters: a scaling factor for the empirical variance, and a mixing weight for the uniform distribution for categorical features that we use to adjust the empirical distribution of the point's 20 nearest neighbors. We use 0.01 to weight the variance and 0.05 to weight the categorical distributions. Finally, when training the trees, we set a local fidelity accuracy threshold of 90% on a validation set and iteratively fit trees with larger maximum depth (up to depth 10) until one achieves this threshold. (We assume assume data points with local models deeper than this will not be interpretable, so fitting deeper trees will not improve our search for the most interpretable model.) We require at least 5 samples at each leaf. We use the scikit-learn implementation [20] to learn the trees and perform a post-processing step that removes leaf nodes iteratively when it does not decrease accuracy on the validation set (as in Wu *et al.* [34]).

How sensitive are the results to these choices? In Figure 6a, we first identify which of our $K = 250$ models would be preferred by each interpretability proxy if the local regions were determined by variance parameters set to $[0.001, 0.01, 0.1]$ and the mixing weights set to $[0.01, 0.05, 0.1]$ (9 combinations). Next, for each of those 9 models, we identify what *rank* it would have had among the K models if one of the other variance or weight parameters had been used. Thus, a rank of 0 indicates that the model identified as the best by one parameter setting is the same as the best model under the second setting; more generally a rank of r indicates that the best model by one parameter setting is the r-best model under the second setting. The generally low ranks in the figure indicate agreement amongst the different choices for local parameter settings. The highest mismatch values for the number of nodes proxy all correspond to the variance scaling factor 0.1 (which we do not use).

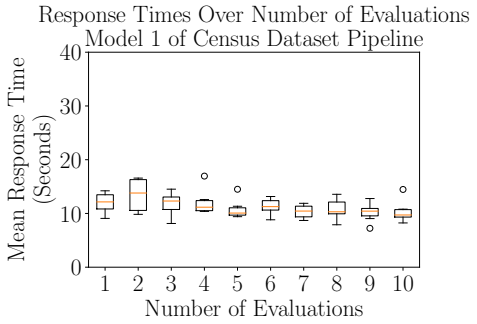
Do we need more points to estimate model rank correctly for any of these region settings? We find the best model(s) by each proxy, then we re-rank models using a small sample of points to compute the same proxy. We do this for the smallest, middle and largest settings of the local region parameters (we do not include cross-pairs of parameters in these results). Figure 6b shows that different hyperparameter settings require similar numbers of input samples x to robustly approximate the integral for $p(M)$ in equation 4 for a variety of interpretability proxies substituted for `HIS`.

D Experimental Details: Human Subject Experiments

In our experiments, we needed to sample input points x to approximate the prior $p(M)$ in equations 2 and 4. For globally interpretable models, we ask users about the same data points across all models to reduce variance. In the locally interpretable case, we only conduct user studies for points near the boundary (in $B(M)$) and thus we sampled points specific to each model's boundary. Each quiz contained 8 or 16 questions per model (8 for the pipeline experiments, 16 for the Amazon Turk experiments), with the order randomized across participants.



(a) An example of our interface with a tree trained on the census dataset with the fewest non-zero features. In our experiments, we show people a decision tree explanation and a data point including only the features that appear in the tree. We then ask them to simulate the prediction according to the explanation.



(b) We asked a single user to take the same quiz 10 times to measure the effect of repetition on response time. The difference in mean response time between the first and last quiz is around 2 seconds. The y-axis scale is the same as that in 4a so the magnitude of the learning effect can be directly compared to the magnitude of the differences between models in our experiment.

There was also an initial set of 3 practice questions. If the participant answered these correctly, we allowed them to move directly to the quiz. If they did not, we gave them an additional set of 3 practice questions. We excluded people who answered fewer than 3 of each set of practice questions correctly from the Amazon Mechanical Turk experiments.

Experiments with Machine Learning Graduate Students and Postdocs For the full pipeline experiment, models were chosen sequentially based on the subjects' responses. We collected responses from 7 subjects for each model in the experiment with the census dataset, and from 9 subjects for each model with the mushroom dataset.[1] We ran 10 iterations of the algorithm, each a quiz consisting of 8 questions about one model, and two evaluations at the end of the same format. We used the mean response time across users to determine $p(M)$. We did not exclude responses, and participants were compensated for their participation. Using the same set of subjects across all of these experiments substantially reduced response variance, although the smaller total number of subjects means we did not see statistically significant differences in our results.

Experiments with Amazon Mechanical Turk We had initially hoped to use Amazon Mechanical Turk for our interpretability experiments. Here, we were forced to use a between subjects design (unlike above), because it would be challenging to repeatedly contact previous participants to take additional quizzes as we chose models to evaluate based on the acquisition function.

In pilot studies, we collected 33 and 24 responses for the two models selected by the pipeline (the first had a medium mean path length, and the second had a high mean path length), after excluding people who did not get one of the two sets of practice questions right, or who took less than 5 seconds or more than 5 minutes for any of the questions on the quiz. The majority of respondents were between 18 and 34. We asked participants 16 questions with a 30 second break halfway through. We paid them $2 for completing the quiz.

The first model, which had a medium mean path length, had a mean time of 31.62s (28.61s - 34.63s), and a median time of 26.86s (23.09s - 30.63s) (standard error and median standard error in parentheses respectively). The second model with a high mean path length had a mean response time of 30.94s (28.66s - 33.22s), and a median response time of 30.32s (27.47s - 33.17s). These intervals are clearly overlapping. We could gather more samples to reduce the variance, but cost grows quickly; running one experiment with the end-to-end pipeline with these sample sizes would have cost around $\$1,000$.

[1] We recorded 2 extra responses for iteration number 4, and 2 fewer responses for iteration number 5 in the census experiment, and 1 extra response in iteration number 3 for the mushroom experiment due to a technical error discovered after the experiment, but we do not believe these affected our overall results. (Extra responses are from the same set of participants.)