Automated Dependence Plots

David I. Inouye
Purdue University
dinouye@purdue.edu

Liu Leqi, Joon Sik Kim
Carnegie Mellon University
{leqil,joonsikk}@cs.cmu.edu

Bryon Aragam
University of Chicago
bryon@chicagobooth.edu

Pradeep Ravikumar
Carnegie Mellon University
pradeepr@cs.cmu.edu

Abstract

In practical applications of machine learning, it is necessary to look beyond standard metrics such as test accuracy in order to validate various qualitative properties of a model. Partial dependence plots (PDP), including instance-specific PDPs (i.e., ICE plots), have been widely used as a visual tool to understand or validate a model. Yet, current PDPs suffer from two main drawbacks: (1) a user must manually sort or select interesting plots, and (2) PDPs are usually limited to plots along a single feature. To address these drawbacks, we formalize a method for automating the selection of interesting PDPs and extend PDPs beyond showing single features to show the model response along arbitrary directions, for example in raw feature space or a latent space arising from some generative model. We demonstrate the usefulness of our automated dependence plots (ADP) across multiple use-cases and datasets including model selection, bias detection, understanding out-of-sample behavior, and exploring the latent space of a generative model. The code is available at https://github.com/davidinouye/adp.

1 INTRODUCTION

Modern applications of machine learning (ML) involve a complex web of social, political, and regulatory issues as well as standard technical issues such as covariate shift or training dataset bias. Although the most common validation metric is test set accuracy, the aforementioned issues cannot be resolved by test set accuracy alone—and sometimes these issues directly oppose high test set accuracy (e.g., privacy concerns). Especially in certain applications such as autonomous cars or automated loan approval, practitioners have become concerned with unexpected and incorrect model behaviors, such as behaviors for data points that were not seen during training or testing (e.g., classifying a person as part of the road or approving a large fraudulent loan). Thus, average-based validation may be insufficient to validate a model. Rather, given the difficulty of specifying expected model behavior *a priori*, qualitative methods for validating a model and highlighting the most interesting or unusual model behaviors beyond average-based methods are needed.

One popular approach for qualitatively validating or understanding the effect of a particular feature on the model response is a partial dependence plot (PDP) [8]. A PDP plots the average model response across one feature marginalized over the other features and thus gives a global view of the feature effect on the model response. Because PDPs take an average over some features, they may not be as effective at showing unusual behaviors that may be important for certain applications as noted above. To address this gap, instance-specific PDPs such as Individual Conditional Expectation (ICE) plots [10] could be used to show the PDP plot for a single target instance rather than averaged over a set of instances. While PDPs were invented many years ago, they have continued to be widely used for understanding model behavior because they are simple to interpret (e.g., [33]).

In order to investigate unusual behaviors for safety-critical applications, practitioners must manually inspect $O(nd)$ instance-specific PDPs, where n is the number of samples they want to investigate and d is the number of features. This manual inspection is prohibitive, even for moderate n and d. Additionally, because PDPs only consider a single feature (i.e., axis-aligned directions), they can miss important interactions between features which may be critical in certain applications. One could use 2D PDP heatmaps, however, this would increase the number of plots from $O(nd)$ to $O(nd^2)$. Thus, despite widespread use, current PDPs suffer from two main drawbacks: (1) a user must manually select interesting plots, and (2) PDPs are usually limited to plots along a single feature.

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

To address these drawbacks, we formalize a method to automate the selection of interesting PDP plots and to extend PDPs to *directional* dependence plots, which show the model response in more general directions—either in raw feature space for tabular data or in a latent space for, e.g., visual or textual data. An illustrative example of such a directional dependence plot is given in Fig. 1. Here, multiple features are being changed and the specific plot is optimized to show the least monotonic direction. This optimization, which is one of our key contributions, is how we automate the selection of "interesting" or relevant plots. We summarize our contributions as follows:

1. We formalize the concept of interestingness or unexpectedness of dependence plots by defining two classes of plot utility functions that have multiple instantiations including utilities to compare two models and validate (or invalidate) certain properties such as linearity, monotonicity, and Lipschitz continuity.
2. We generalize PDPs beyond axis-aligned directions to consider the model response along sparse linear directions. In particular, we optimize a specified utility measure over sparse linear directions using a greedy coordinate pairs algorithm. For tabular data where the features are inherently interpretable, we optimize for a sparse linear direction in the raw feature space. For rich data such as images or text, we propose to find sparse linear directions in a latent representation space (e.g., via a VAE) but show the corresponding images along this direction in the original raw feature space.
3. We demonstrate the usefulness of the resulting automated dependence plots (ADPs) across multiple use-cases and datasets including model selection, bias detection, understanding out-of-sample behavior, and exploring the latent space of a generative model.

Related work. The importance of safety and robustness in ML is now well-acknowledged [24, 31], especially given the current dynamic regulatory environment surrounding the use of AI in production [13, 32]. A popular approach to auditing and checking models before deployment is "explaining" a black-box model post-hoc. Both early and recent work in explainable ML rely on local approximations (e.g., [11, 23]). Other recent papers have studied the influence of training samples or features [5, 6]. These have been extended to subsets of informative features [2, 5, 19] to explain a prediction. Other approaches employ influence functions [18] and prototype selection [34, 35]. A popular class of approaches take the perspective of local approximations, such as linear approximations [19, 23], and saliency maps [27, 28, 30]. A crucial

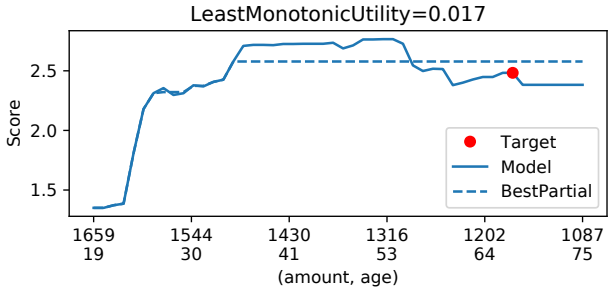


Figure 1: The above example highlights the two key innovations of our proposed automated dependence plots: (1) Finding interesting plots by formalizing and optimizing plot utility measures—in this case, the utility is based on the non-monotonicity of the curve. (2) Optimizing over directions that change multiple features rather than only varying a single feature as in standard PDP plots (as indicated on the x-axis). For the target loan application (designated by the red point), the directional dependence plot (solid line) shows the change in model scores (y-axis) along a direction in which two numeric features (amount and age) are varied. The plot was optimized so that the model response along the plot is the least monotonic (for comparison, the best monotone regression line is shown via the dotted line; see Sec 2 for more details).

caveat with such approximation-based methods is that the quality of the approximation is often unknown, and is typically only valid in small neighborhoods of an instance (although we note recent work on global approximations [12]). In contrast to these previous feature selection methods, our approach leverages a utility measure to select features or directions.

2 PLOT UTILITY MEASURES

In this section, we define two classes of utility measures for generic dependence plots including PDP plots, ICE plots, and our generalization to directional dependence plots described in the next section. The intuition is that the plot utility measure quantifies the "interestingness" or relevance of a particular feature or direction. As we will discuss in more detail in Sec. 4, it is implicit here that we wish to *maximize* the utility (cf. (7)).

Notation. We assume that the input space is $\mathcal{X} = \mathbb{R}^d$—we could consider categorical variables by showing bar charts as in categorical PDP plots but for simplicity, we will focus on continuous features. For instance-specific plots, we will denote the target instance as $\boldsymbol{x}_0 \in \mathbb{R}^d$. Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a black-box function, i.e., we can query f to obtain pairs $(\boldsymbol{x}, f(\boldsymbol{x}))$. We do not assume that f is differentiable or even continuous in order to allow non-differentiable models such as ran-

dom forests. We will denote a plot by it's corresponding univariate function $\tilde{f}(t)\colon \mathbb{R} \to \mathbb{R}$, where we visualize this function for some bounded interval, i.e., for $t \in [a, b]$—the bounds for standard PDP plots are usually based on the minimum and maximum values along each feature but we will generalize this for directional plots in the next section. For other multivariate functions $g(\boldsymbol{x})$, we will denote similarly the corresponding univariate function as $\tilde{g}(t)$. As an example, for PDP $\tilde{f}_i(t) = \mathbb{E}[f(\ldots, x_{i-1}, t, x_{i+1}, \cdots)]$ where the expectation is an empirical expectation with respect to some dataset (often a training dataset) and t ranges over the minimum and maximum of the i-th feature. For instance-specific PDP (ICE), $\tilde{f}_i(t) = f(\ldots, x_{i-1}, t, x_{i+1}, \cdots)$ where the input $\boldsymbol{x}$ is fixed to a target point $\boldsymbol{x}_0$ except for the i-th feature—this can be seen as a PDP plot where the dataset is a single point. We will denote $U(\tilde{f}, a, b)$ as a plot utility function where we usually suppress the dependence on the bounds a and b for simplicity and merely denote the utility as $U(\tilde{f})$.

In the next subsections, we carefully develop and define two general classes of plot utility measures. Both classes of utilities compare to another plot, which we will show as a dotted line as can be seen in Fig. 1. This helps give the reader an interpretable reference for the utility measure itself—e.g., showing the best bounded Lipschitz approximation to the plot. Of course, we do not claim that our proposed set of utility measures is exhaustive, and indeed, this paper will hopefully lead to more creative and broader classes of utility measures.

2.1 MODEL CONTRAST UTILITY MEASURES

A natural way to measure the utility of a plot with respect to one model is to contrast it to the same plot based on a different model. Given another multivariate model $g_{\boldsymbol{x}_0}(\boldsymbol{x})\colon \mathbb{R}^d \to \mathbb{R}$, which could be defined with respect to a target instance, we define the contrast utility measure:

$$U_{\text{c}}(\tilde{f}) = \int_a^b \mathcal{L}(\, \tilde{f}(t),\ \tilde{g}_{\boldsymbol{x}_0}(t)\,)dt\,, \tag{1}$$

where $\mathcal{L}$ is a loss function, e.g., squared or absolute loss, and $\tilde{g}_{\boldsymbol{x}_0}(t)$ is the corresponding plot function with respect to $g_{\boldsymbol{x}_0}(\boldsymbol{x})$. By maximizing $U_{\text{c}}(\tilde{f})$, we can find the plot $\tilde{f}$ that differs the *most* from the baseline model $\tilde{g}_{\boldsymbol{x}_0}(t)$. The baseline could be another simple model like logistic regression or some local approximation of the model around the target instance such as explanations produced by LIME [23]—this would provide a way to critique local approximation methods and show where and how they differ from the true model.

Example 1 (Variance of Plot)**.** *While the PDP paper [8] did not propose ways for sorting or selecting interesting plots, a commonly used method to sort PDP curves is by the variance of the plot. The utility of a plot based on variance can be seen as special case of the model contrast utility where $\tilde{g}(\boldsymbol{x}) = c = \mathbb{E}[\tilde{f}(s)] = 1/(b-a)\int_a^b \tilde{f}(s)ds$—i.e., an expectation of $\tilde{f}$ with respect to a uniform distribution between a and b—and the loss function is squared error. We will use this as the default sorting mechanism for comparison in our experiments because it is the simplest and seems to be common in practice.*

Example 2 (Contrast with Constant Model)**.** *Sometimes we may want to create a contrast model that is dependent on the target point $\boldsymbol{x}_0$. The simplest case is where the contrast model is a constant fixed to the original prediction value, i.e., $g_{\boldsymbol{x}_0}(\boldsymbol{x}) = c_{\boldsymbol{x}_0} = f(\boldsymbol{x}_0)$ Note that in this case the comparison function depends on $\boldsymbol{x}_0$. This contrast to a constant model can find directions that deviate the most from the prediction; this implicitly finds plots that are not flat and significantly affect the prediction value.*

Example 3 (Contrast with Validated Linear Model)**.** *Suppose an organization has already deployed a carefully validated linear model—i.e., the linear parameters were carefully checked by domain experts to make sure the model behaves reasonably with respect to all features. The organization would like to improve the model's performance by using a new model, but wants to see how the new model compares to their carefully validated linear model to see where it differs the most. In this case, the organization could let the contrast model be their previous model, i.e., $g_{\boldsymbol{x}_0}(\boldsymbol{x}) = g_{\text{Linear}}(\boldsymbol{x})$ where g does not depend on the target point $\boldsymbol{x}_0$.*

Example 4 (Contrast Random Forest and DNN)**.** *An organization may want to compare two different model classes such as random forests and deep neural networks (DNN) to diagnose if there are significant differences in these model classes or if they are relatively similar. In this case, the contrast model $g(\boldsymbol{x})$ would be the random forest or DNN.*

Example 5 (Contrast with local approximations used for explanations)**.** *We can also compare the true model with explanation methods based on local approximation such as LIME [23] or gradient-based explanation methods [26, 28, 30]. We can simply use the local approximation to the model centered at the target point as the contrast model,i.e., $g_{\boldsymbol{x}_0}(\boldsymbol{x}) = \hat{f}_{\boldsymbol{x}_0}(\boldsymbol{x})$, where $\hat{f}_{\boldsymbol{x}_0}$ is the local approximation centered around $\boldsymbol{x}_0$. Thus, the found diagnostic curve will show the maximum deviation of the true model from the local approximation model being used for an explanation. Importantly, this allows our diagnostic method to assess the goodness of local approximation explanation methods showing when they are reasonable and when they may fail; see Fig. 2.*

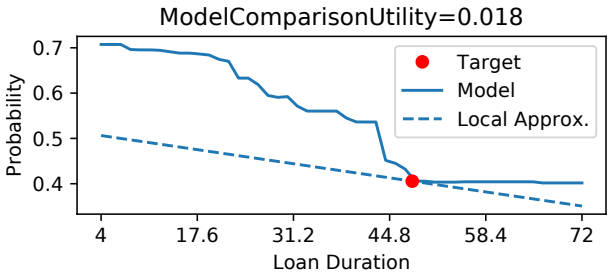


Figure 2: This plot illustrates using the model contrast utility where $g_{\boldsymbol{x}_0}$ (dotted line) is an explanation model based on the gradient similar to the local linear explanation models in LIME [23]. Notice how it shows where the approximation may be appropriate (duration > 46) and where it might be far from the true model (duration < 46).

2.2 FUNCTIONAL PROPERTY (IN)VALIDATION UTILITY MEASURES

In many contexts, a user may be more interested in validating (or invalidating) certain functional properties of a model, such as monotonicity or smoothness. For example, if it is expected that a model should be increasing with respect to a feature (e.g., income in a loan scoring model), then we'd like to check that this property holds true (at least approximately). Let $\mathcal{H}$ be a class of univariate functions $\tilde{h} : \mathbb{R} \to \mathbb{R}$ that represents a property that encodes acceptable or expected behaviors. To measure deviation of a plot from this class of functions, take the minimum expected loss over all $h \in \mathcal{H}$:

$$U_{\text{p}}(\tilde{f}) = \min_{\tilde{h} \in \mathcal{H}} \int_a^b \mathcal{L}(\tilde{f}(t), \tilde{h}(t)) dt \tag{2}$$

where as usual, $\mathcal{L}$ is a loss function. The minimization in (2) is a univariate regression problem that can be solved using standard techniques. This utility will find dependence plots that maximally or minimally violate the functional properties encoded by $\mathcal{H}$.

Example 6 (Linearity (in)validation via linear regression)**.** *A user might want to view the plots that are the least linear to see if there is some unusual plots that may need further investigation. For this example, the class of functions would merely be linear functions, i.e., $\mathcal{H}$ is the set of univariate linear functions. This problem could be solved easily using standard linear regression methods.*

Example 7 (Monotonicity (in)validation via isotonic regression)**.** *In many applications, it may be known that the model output should behave simply with respect to certain features. For example, one might expect that the score is monotonic in terms of income in a loan scoring model. In this case, the class of functions should be the set of monotonic functions, i.e., $\mathcal{H}$ is the set of all monotonic functions. The resulting problem can be efficiently solved using isotonic regression [1]—and this is what we do in our experiments; see Fig. 4 for an example of validating (or invalidating) the monotonic property.*

Example 8 (Lipschitz-boundedness (in)validation via constrained least squares)**.** *Another property that an organization might want to validate is whether the function has a small Lipschitz constant along the curve. Formally, they may wish to check if the following condition holds:*

$$\left| \frac{\tilde{f}(t_2) - \tilde{f}(t_1)}{t_2 - t_1} \right| \leq L\,, \quad \forall t_1, t_2 \in [a, b] \tag{3}$$

where L is a fixed Lipschitz constant. Thus, the corresponding class of functions $\mathcal{H}_L$ is the set of Lipschitz continuous functions with a Lipschitz constant less than L. In practice, we can solve this problem via constrained least squares optimization—similar to isotonic regression (details in supplement). An example of using Lipschitz bounded functions for $\mathcal{H}$ can be seen in Fig. 3. This utility will find the curve that maximally violates the bounded Lipschitz condition; this curve may also be useful in finding where the model is particularly sensitive to changes in the input since the derivative along the curve will be larger than the Lipschitz constant.

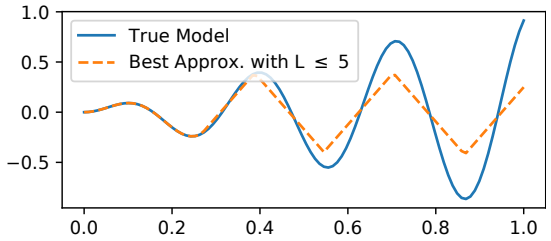


Figure 3: Example of contrast utility function when the function class is Lipschitz continuous with $L \leq 5$.

3 DIRECTIONAL DEPENDENCE PLOTS

In this section, we formally define a directional dependence plot, which generalizes classical PDPs and instance-specific PDPs. First, recall that a PDP $\tilde{f}_i$ varies the i-th feature and averages over the other features in a dataset, i.e., $\tilde{f}_i(t) = \mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x} + t\mathbf{e}_i)]$, where $\mathbf{e}_i$ is the i-th standard basis vector. In this form of $\tilde{f}_i$, the generalization to directions is quite straightforward as we can replace $\mathbf{e}_i$ by $\boldsymbol{v}$ where $\boldsymbol{v}$ is an arbitrary unit vector: $\tilde{f}_{\boldsymbol{v}}(t) = \mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x} + t\boldsymbol{v})]$. The instance-specific dependence plot with respect to $\boldsymbol{x}_0$ is:

$$\tilde{f}_{\boldsymbol{v},\boldsymbol{x}_0}(t) = f(\boldsymbol{x}_0 + t\boldsymbol{v})\,. \tag{4}$$

In future sections, we will often suppress the dependence on $\boldsymbol{v}$ or $\boldsymbol{x}_0$ if it is understood from the context. Because this paper focuses on finding unusual or interesting plots

for the purposes of model validation especially in safety-critical applications, we will focus on the local instance-specific directional plots in our experiments but these ideas can clearly be applied to "global" directional plots. We note that both for interpretability and computational reasons, we will usually assume that $\boldsymbol{v}$ is sparse or low-dimensional in the following sections. In the next sections, we will develop generalizations for directional plots in latent spaces, parameter spaces of transformations, and other general considerations for directional plots.

Directions in a latent space. For rich data such as images, directions in the original input space (e.g., raw pixels) may not be very informative. In these cases, it may be more intuitive to move along directions in a latent space, such as one arising from a generative model $G : \mathbb{R}^{\bar{d}} \mapsto \mathbb{R}^d$, that generates input vectors in $\mathbb{R}^d$ given latent vectors in $\mathbb{R}^{\bar{d}}$, where $\bar{d}$ is usually smaller than d (e.g., a low dimensional representation). Examples of such models include VAEs [17, 22] and deep generative models via normalizing flows [7, 9, 14, 20, 21]. We then optimize for directions in the latent space of the generative model $G(\cdot)$ rather than the raw input space itself. This allows us to define directions in the latent space that can correspond to arbitrary curves in the latent space:

$$\tilde{f}_{G,\boldsymbol{x}_0}(t) = f(G(\boldsymbol{z}_0 + t\boldsymbol{v}))\,, \tag{5}$$

where $\boldsymbol{z}_0 = G^{-1}(\boldsymbol{x}_0)$ and $G^{-1}(\boldsymbol{x}_0)$ is an (approximate) input to the generative model that would have generated $\boldsymbol{x}_0$, i.e., $G(G^{-1}(\boldsymbol{x}_0)) \approx \boldsymbol{x}_0$. For VAEs [17, 22], the decoder network acts as G and the encoder network acts as an approximate G^{-1}. For normalizing flow-based models, both $G(\boldsymbol{z})$ and $G^{-1}(\boldsymbol{x}_0)$ can be computed exactly by construction [7, 9, 14, 20, 21].

Directions in the parameter space of known transformations. In certain domains, there are natural classes of transformations that are semantically meaningful. Examples include adding white noise to an audio file or blurring an image, removing semantic content like one letter from a stop sign, or changing the style of an image using style transfer [15]. We can consider directions in the parameter space of a set of these known transformations, which could be arbitrarily complex and non-linear. We will denote each transformation as $\lambda_v \colon \mathbb{R}^d \to \mathbb{R}^d$ with parameter $v \in [0, 1]$, where λ_0 corresponds to the identity transformation. For example, if λ_v a rotation operator, then v would represent the angle of rotation where $v = 0$ is 0 degrees and $v = 1$ is 180 degrees. Given an ordered set of ℓ different transformations denoted by $(\lambda_v^{(1)}, \lambda_v^{(2)}, \ldots, \lambda_v^{(\ell)})$, we can define a plot function using a composition of these simpler transformations:

$$\begin{aligned} \tilde{f}_{\boldsymbol{v},\boldsymbol{x}_0}(t) &= f(\Lambda_{\boldsymbol{x}_0}(\boldsymbol{v}t)) \\ \Lambda_{\boldsymbol{x}_0}(\boldsymbol{v}) &\triangleq \lambda_{v_\ell}^{(\ell)}(\ldots \lambda_{v_2}^{(2)}(\lambda_{v_1}^{(1)}(\boldsymbol{x}_0))\ldots) \end{aligned} \tag{6}$$

where $\boldsymbol{v} \in [0, 1]^\ell$ is the optimization parameter. Thus, directions in the parameter space correspond to non-linear transformation curves in the original input space. We note that the transformations can be arbitrarily non-linear and complex—even deep neural networks.

Realistic plot bounds. In some contexts, it is desirable to explicitly audit the behavior of f off its training data manifold. This is helpful for detecting unwanted bias and checking robustness to rare, confusing, or adversarial examples—especially in safety critical applications such as autonomous vehicles. In other contexts, we may wish to ensure that the selected plots show realistic combinations of variables—that is, that they stay within the bounds of the training data distribution. For example, it may not be useful to show an example of a high income person who also receives social security benefits. Fortunately, it is not hard to enforce this constraint: First, bound the endpoints of the plot using a box constraint based on the minimum and maximum of each feature—this is all that classical PDP plots consider. However, this simple bounding may not be reasonable when we can move along multiple features. Thus, we can also train a density model on the training data to account for feature dependencies and further restrict the bounds of the plot to only lie in regions with a density value above a threshold. In our experiments, we use a simple multivariate Gaussian density estimate to create the boundaries of our plots but more complex deep density estimators could be used (e.g., [7, 14, 20, 21]).

Visualizing directional dependence plots. In typical PDP plots, the feature values are shown on the horizontal axis—yielding direct interpretability if the feature itself is interpretable. For directional plots, we show the set of feature values that are changing that correspond to each value of t (see the x-axis of Fig. 1). For example, if both age and duration of a loan application are changing across a directional plot, we show pairs of age and duration along the horizontal axis. For image-based examples, we show the image in the original feature space corresponding to different t values to maintain interpretability.

4 OPTIMIZATION OF PLOT UTILITY

So far we have discussed: 1) how to measure the plot utility and 2) how to generalize beyond axis-aligned directions to arbitrary directions. With these pieces in place, the objective is to find directional dependence plots that

are simultaneously interpretable and interesting. To do this, we restrict the directions under consideration (usually to sparse directions) and optimize for the direction that shows the best (either highest or lowest) utility:

$$\boldsymbol{v}^* = \arg\max_{\boldsymbol{v}} U(\tilde{f}_{\boldsymbol{v},\boldsymbol{x}_0}), \tag{7}$$

where $\boldsymbol{v}$ is restricted to some interpretable subset (e.g., sparse vectors). This optimization problem can also be optimized over a set of target instances as follows:

$$\boldsymbol{v}^*, \boldsymbol{x}_0^* = \arg\max_{\boldsymbol{v},\boldsymbol{x}_0} U(\tilde{f}_{\boldsymbol{v},\boldsymbol{x}_0}), \tag{8}$$

where $\boldsymbol{x}$ is one instance in a specified dataset (e.g., a random sample of the training dataset). This finds both a target data point and a direction where f exhibits unusual or interesting behavior defined by the utility U, thus enabling users to see the worst case behavior of the model. Note that the dataset used for optimization may not be part of the training data because it may be new data where no class labels are given.

Since f is assumed to be an arbitrary black-box, where only function evaluations are possible, the optimization problem even for a single target instance in (7) is usually nonconvex and nonsmooth. This model-agnostic setup generalizes to many interesting settings including ones in which the model is private, distributed or nondifferentiable (e.g., boosted trees) such that exact gradient information would be impossible or difficult to compute. Given these restrictions, we must resort to zeroth-order optimization. While we could use general zeroth-order optimization techniques, we require that the directions are sparse so that the resulting directional plots are interpretable. Thus for both computational and practical reasons, we propose a greedy optimization scheme called *greedy coordinate pairs* (GCP) that adds non-zero coordinates in a greedy fashion, as outlined in Algorithm 1. We initialize this algorithm by finding the single feature with the highest utility—this is the same as computing the best axis-aligned direction for standard PDP plots.

Computational complexity. The total complexity of the GCP algorithm—which is easily parallelized—is $O(dkMI)$, where M is the number of angles tested and I is the number of iterations. In our experiments, GCP typically found an interesting direction within 10 iterations. By comparison, the computational complexity of standard PDP plots for all dimensions in terms of the number of model evaluations is $O(ndk)$. Thus, compared to PDP plots, our method is at most $O(MI)$ slower.

Optimization evaluation. To test the effectiveness of GCP (Algorithm 1), we ran two tests. First, we compared the optimal utility values returned by GCP to the utility

Algorithm 1 Greedy Coordinate Pairs (GCP)

Input: Plot utility U, target point $\boldsymbol{x}_0$, max number of features D, grid size M
Output: Optimized direction $\boldsymbol{v}^*$
$G(i,j,\theta) \equiv$ Givens rotation matrix for coordinates i and j with angle θ
$\Theta = \{0, \frac{\pi}{M}, \frac{2\pi}{M}, \cdots, \pi\}$
$\boldsymbol{v} \leftarrow \arg\max_{i\in\{1,\ldots,d\}} U(\tilde{f}_{\mathbf{e}_i})$ {Coordinate-wise optimization}
while $\boldsymbol{v}$ not converged **do**
  $i^*, j^*, \theta^* \leftarrow \arg\max_{i,j,\theta\in\Theta} U(\tilde{f}_{G(i,j,\theta)\boldsymbol{v}})$
    s.t. $\sum_k \mathbf{I}([G(i,j,\theta)\boldsymbol{v}]_k \neq 0) \leq D$
  $\boldsymbol{v} \leftarrow G(i^*, j^*, \theta^*)\boldsymbol{v}$
end while

of 10,000 randomly selected directional plots based on a random forest classifier. In all cases, GCP returned the highest values. For example, GCP found a directional plot $\tilde{f}$ with $U(\tilde{f}) = 0.0016$, compared with 0.0007 for the best random curve. In the second experiment, we generated random synthetic models in which the directional plot with the highst utility could be determined in advance, making it possible to evaluate how often GCP selects the optimal one. We evaluated its performance on directions with at most one, two, or three nonzero coordinates, and found that in 100%, 97%, and 98% of simulations, GCP found the optimal directions. In the few cases GCP did not find the optimal directions, this was due to the randomness in generating examples whose optimal directions are more difficult to identify (e.g.,, the optimal curve was nearly constant). Details and further results on these experiments can be found in the appendix. Thus, we conclude that despite our algorithm being greedy, our GCP algorithm is empirically reasonable.

5 EXPERIMENTS

We present five concrete use cases: 1) Selecting among several models, 2) Bias detection, 3) Out-of-sample behavior in computer vision, 4) Discovering interesting model properties via generative models, and 5) Evaluating robustness to covariate shift. We have put (5) and some details of the experiments in the appendix. A python module implementing the proposed framework is available at https://github.com/davidinouye/adp.

5.1 SELECTING A MODEL FOR LOAN PREDICTION

Suppose we have trained multiple models, each with similar test accuracies. Which of these models should be deployed? To answer this question, directional dependence

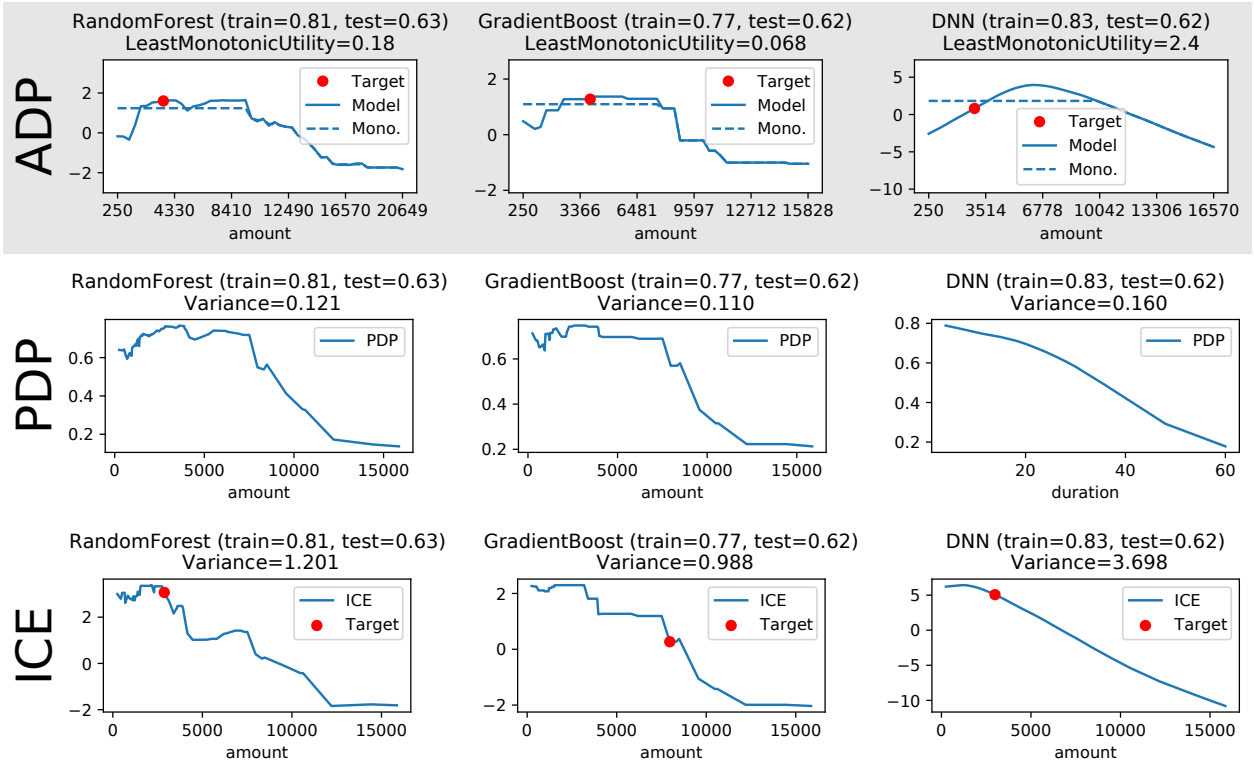


Figure 4: Our dependence plots (top, gray) selected by finding the least monotonic direction elucidates potentially problematic behavior of the various models whereas PDP (middle) and instance-specific ICE (bottom) plots do not highlight this non-monotonic behavior. The PDP and ICE plots were selected according to the simple highest variance utility, which is used in practice for standard PDP plots. All plots are based on 100 random samples, where the PDP plots average over the samples and our plots and the ICE plots additionally select a specific target point. The dotted line (labeled "Mono.") is the best monotonic regression plot.

plots can be used to detect undesirable or unexpected behaviours. We explore this use-case of qualitative model selection on a dataset of German loan application data in order to find non-monotonic plots. For example, it may be undesirable to deploy a model that decreases a candidate's score in response to an increase in their income. We train a random forest, a gradient boosted tree, and a deep neural network. For this simple example, we consider axis-aligned directions with only one non-zero so that we can compare to PDP and ICE plots; additionally, we optimize the utility over 100 random target points $\boldsymbol{x}_0$ as in Eq. 8—thus providing an estimate of the worst-case behavior of the model. The test accuracies of these models were all close to 62%—thus, a stakeholder would not be able to select a model based on accuracy alone. The plots for our method using the monotonicity utility compared to PDP and ICE using the simple variance utility can be seen in Fig. 4. In addition to a single number that quantifies non-monotonicity, our directional dependence plots show the user both the *location* and *severity* of the worst-case non-monotonicity. On the other hand, PDP and instance-specific PDP (ICE) plots selected by variance prefer models that have large ranges, but may have expected or uninteresting patterns within this range. By contrast, our dependence plots can highlight more subtle and interesting patterns such as non-monotonicity. Our directional dependence plots suggest that random forests or gradient boosted trees may be preferable since their worst-case behaviors are nearly monotonic, whereas the DNN model is far from monotonic.

5.2 BIAS DETECTION IN RECIDIVISM MODEL PREDICTION

In many real-world applications, it is essential to check models for bias. A contemporary example of significant interest in recent years concerns *recidivism prediction instruments* (RPIs), which use data to predict the likelihood of a convicted criminal to re-offend [3, 4]. Given an instance $\boldsymbol{x}_0 = (u, x_2, \ldots, x_d)$, consider what the output of f would have been had the protected attribute $u \in \{0, 1\}$ (e.g., race or gender) been flipped. In certain

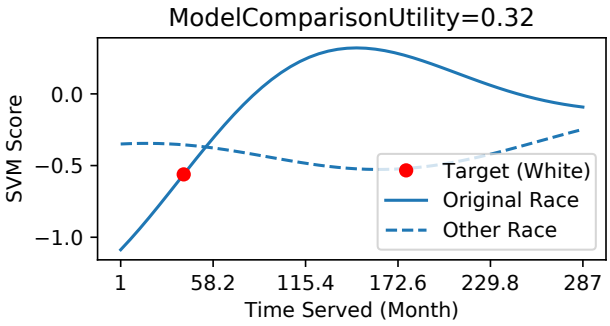


Figure 5: A directional plot using the model contrast utility where the comparison model is the same model but the race is flipped. Notice that bias between races is far from uniform even switching bias direction. Expanded figure with flipping both gender and race on two target points can be found in the appendix.

cases, such protected attributes might not be explicitly available, in which case, we could use proxy attributes or factors, though we do not focus on that here. A model that ignores the protected attributes would see little or no change in f as a result of this change. In this simple example, we explicitly ignore dependencies between the protected attribute and other features though this would be important to consider for any significant conclusions to be made. Given this situation, we select the model contrast utility (Sec 2) with a special definition for the comparison model defined as follows: $g_{\boldsymbol{u}_0}(\boldsymbol{u}) = f_{\boldsymbol{u}_0}(\sigma(\boldsymbol{u}))$, where $[\sigma(\boldsymbol{u})]_i$ is $1-u_i$ if the i-th feature is the protected attribute, and u_i otherwise, essentially flipping only the protected attribute and leaving all other features untouched. There are two cases: Either (a) No such plot deviates too much, in which case this is evidence that f is not biased, or (b) there is a dimension along which f is significantly biased. A directional plot for flipping race from white to black based on data from [25] using a kernel SVM model can be seen in Fig 5. One can clearly see that the model behaves in biased ways with respect to race: The effect of time served on the risk score clearly depends on race and even switches bias direction. For this high stakes application, the use of such a model could be problematic, and our directional plots highlight this fact easily even for non-technical users. Finally, these directional plots avoid averaging the data into a single metric and offer more insight into the location and form of the model bias that may depend on the inputs in complex ways.

5.3 UNDERSTANDING OUT-OF-SAMPLE BEHAVIOR FOR TRAFFIC SIGN DETECTION

When the model is deployed in practice, understanding how the model behaves outside of the training dataset is often critical. For example, for a traffic sign detector on autonomous vehicles, will the detector predict correctly if a traffic sign is rotated even though the training data did not contain any rotated signs? What other variations of the data is the detector susceptible to? For this use-case, we first trained a convolution neural network on German Traffic Sign Recognition Benchmark (GTSRB) dataset [29] which achieves 97% test accuracy, that will act as the stop sign detector. We consider transformation curves based on five standard image transformations: rotate, blur, brighten, desaturate, and contrast. Each of these transformations creates images outside of the training data, and we are interested in finding the interesting combinations of these transformations that influence the prediction score of the detector in different ways. Fig. 6 depicts the resulting plots we generated through optimizing the corresponding utilities. The least constant direction (top) simultaneously adjusts contrast and brightness, which is expected since this transformation gradually removes almost all information about the stop sign. The most constant direction (middle) adjusts saturation and contrast, which may be surprising since it suggests that the model ignores color. Finally, the least monotonic direction (bottom) is rotation, which suggests that the model identifies a white horizontal region but ignores the actual letters "STOP" since it still predicts correctly when the stop sign is flipped over.

5.4 DISCOVERING INTERESTING MODEL PROPERTIES VIA GENERATIVE MODELS

Our directional dependence plots can also search non-linear directions (i.e., curves) in the input space, by discovering linear directions in some latent feature space learned by generative models. This can be particularly useful when linear directions in the input feature space are less semantic compared to non-linear directions, like images. In this example, we demonstrate how directional plots effectively select the most interesting non-linear directions in the pixel space.

On the MNIST data, we train a VAE [17] for learning a 10-dimensional latent data representation, as well as a CNN classifier. The directions in the latent space of deep networks are found to be semantic in several ways [16], so we optimize in this space for the least constant, most constant, and least monotonic in terms of the prediction score change of the classifier. In Figure 7, given a test instance of digit 1, we show the varying prediction probability (left) along with the set of reconstructed images along the direction (right). Least-constant utility (top row) discovers directions that add distinctive curves and loops to the image inducing a steep drop in the prediction probability, while the most-constant utility (middle row) finds directions that keep the relative shape of the digit 1 con-

Figure 6: The dependence plots for traffic sign detection show which transformation is the most sensitive (top), the least sensitive (middle), and the least monotonic (bottom). These curves highlight both expected model trends (top) and unexpected trends (middle and bottom), where the model seems to ignore color and fails when the stop is rotated partially but works again when the stop sign is almost flipped over.

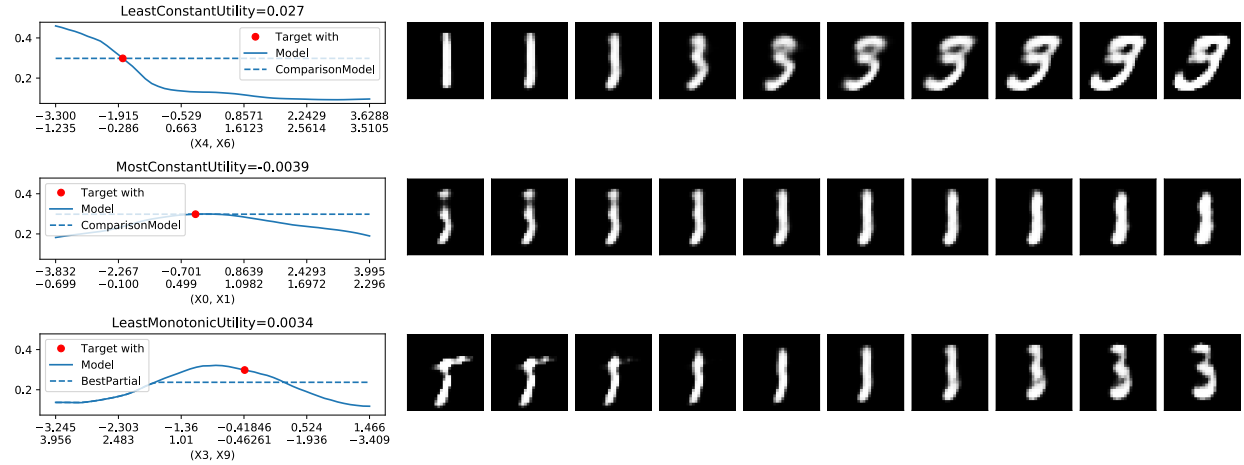


Figure 7: Discovering various non-linear directions in images using VAE. Given the target instance of digit 1, the least-constant utility highlights that adding more curves and loops to the image induces the prediction probability to drop the most (top row), while the most-constant utility identifies directions that preserve the relative shapes of the digit (middle row). The least-monotonic utility exposes the directions that make the prediction probability fluctuate the most – moving along these directions alter the digit to different numbers, e.g., 5 and 3.

stant throughout. Least-monotonic utility (bottom row) successfully exposes directions that change the digit 1 to 5 and 3, inducing the most non-monotonic prediction probability changes. Note that such discovery is not limited to images — as long as some semantically meaningful feature embedding is available (e.g., word embedding), the same process can be used to automatically discover interesting directions within the input feature space. While these use cases demonstrate our method across a wide range of scenarios, a user study for various contexts such as exploratory data analysis, model checking, or model comparison would be an excellent area for future research.

Overall, we believe our framework opens up new possibilities and challenges for automating and extending PDP plots.

Acknowledgements

The authors acknowledge the support of DARPA via FA87501720152 and Accenture. DII acknowledges support from Northrop Grumman. JSK acknowledges support from Kwanjeong Educational Foundation. BA acknowledges support from the Robert H. Topel Faculty Research Fund.

References

[1] M. J. Best and N. Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47(1-3):425–439, 1990.

[2] J. Chen, L. Song, M. Wainwright, and M. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *ICML*, pages 883–892, 2018.

[3] A. Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

[4] A. Chouldechova and M. G'Sell. Fairer and more accurate, but for whom? *Workshop on Fairness, Accountability, and Transparency in Machine Learning*, 2017.

[5] A. Datta, S. Sen, and Y. Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *IEEE Symposium on Security and Privacy (SP)*, pages 598–617. IEEE, 2016.

[6] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: towards contrastive explanations with pertinent negatives. In *NeurIPS*, pages 590–601. Curran Associates Inc., 2018.

[7] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. In *ICLR*, 2017.

[8] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5): 1189–1232, 2001.

[9] M. Germain, K. Gregor, I. Murray, and H. Larochelle. MADE: Masked autoencoder for distribution estimation. In *ICML*, 2015.

[10] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65, jan 2015.

[11] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. Local rule-based explanations of black box decision systems, 2019.

[12] W. Guo, S. Huang, Y. Tao, X. Xing, and L. Lin. Explaining deep learning models–a bayesian non-parametric approach. In *NeurIPS*, pages 4514–4524, 2018.

[13] G. Hadfield and J. Clark. Regulatory markets for AI safety. *Safe Machine Learning Workshop at ICLR*, 2019.

[14] D. I. Inouye and P. Ravikumar. Deep density destructors. In *ICML*, pages 2172–2180, jul 2018.

[15] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 694–711, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46475-6.

[16] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, pages 2668–2677, 2018.

[17] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[18] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017.

[19] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *NIPS*, 2017.

[20] J. B. Oliva, A. Dubey, M. Zaheer, B. Póczos, R. Salakhutdinov, E. P. Xing, and J. Schneider. Transformation autoregressive networks. In *ICML*, 2018.

[21] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *NIPS*, 2017.

[22] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[23] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *KDD*, 2016.

[24] S. Saria and A. Subbaswamy. Tutorial: Safe and reliable machine learning. *arXiv preprint arXiv:1904.07204*, 2019.

[25] P. Schmidt and A. D. Witte. *Predicting Recidivism in North Carolina, 1978 and 1980*. Inter-university Consortium for Political and Social Research, 1988.

[26] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. *ICML*, 2017.

[27] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *Workshop at ICLR*, 2014.

[28] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *Visualization for Deep Learning Workshop at ICML*, 2017.

[29] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

[30] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *ICML*, 2017.

[31] K. R. Varshney and H. Alemzadeh. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big data*, 5(3):246–255, 2017.

[32] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *SSRN Electronic Journal*, 11 2017. doi: 10.2139/ssrn.3063289.

[33] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65, 2019.

[34] C.-K. Yeh, J. Kim, I. E.-H. Yen, and P. K. Ravikumar. Representer point selection for explaining deep neural networks. In *NeurIPS*, pages 9291–9301, 2018.

[35] X. Zhang, A. Solar-Lezama, and R. Singh. Interpreting neural network judgments via minimal, stable, and symbolic corrections. In *NeurIPS*, pages 4874–4885, 2018.

A Evaluating robustness under covariate shift in loan prediction

Using the same dataset as in the previous example, we compare the behavior of the *same* model over different regions of the input space. The motivation is covariate shift: A bank has successfully deployed a loan prediction model but has historically only focused on high-income customers, and is now interested in deploying this model on low-income customers. Is the model robust? Once again we use dependence plots to detect undesirable behaviour such as non-monotonicity. We trained a deep neural network that achieves 80% accuracy on high-income data and a comparable 76% accuracy on the unseen low-income data. Moving beyond accuracy, we generated directional plots using the least monotonic utility optimizing over all target points in the training data (i.e., high-income) and the unseen test data (i.e., low-income) as can be seen in Fig. 8. The curves explicitly display the difference between the worst-case non-monotonicity for high-income (Fig. 8, top) and low-income (Fig. 8, bottom), which appears to be minimal, giving stakeholders confidence for deploying this model.

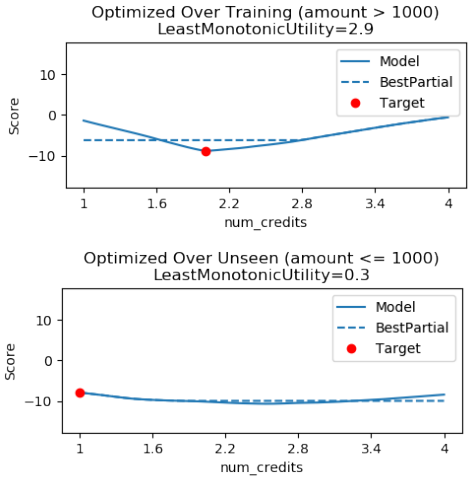


Figure 8: Our dependence plots selected by finding the least monotonic direction for deep neural network model trained on large loan applications (amount > 1,000) when optimizing over (top) the training distribution (i.e., amount > 1,000) and over (bottom) the unseen novel small loan distribution (i.e., amount $\leq$ 1,000). The dotted line (labeled "BestPartial") is the best isotonic regression model.

B Lipschitz-bounded property validation formulated as a constrained least squares problem

Suppose we have a grid of t_i values and corresponding to model outputs along the curve $y_i = f(\phi(t_i))$. Now let $\boldsymbol{b} = \boldsymbol{y}$ and let A be defined as follows:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & t_2 - t_1 & 0 & 0 & 0 \\ 1 & t_2 - t_1 & t_3 - t_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & t_2 - t_1 & \dots & \dots & t_n - t_{n-1} \end{bmatrix}. \quad (9)$$

Now we solve the following simple least squares problem:

$$\begin{aligned} \hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \quad & \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 \\ \text{s.t.} \quad & -L \leq x_i \leq L, \quad \forall i \in \{2, n\}. \end{aligned} \quad (10)$$

Notice that the first coordinate x_1 is unconstrained and represents $\hat{y}_1$. The rest of $\boldsymbol{x}$ correspond to the slope of a line connecting each point; thus $\hat{y}_2 = \hat{y}_1 + (t_2 - t_1)x_2$ and $\hat{y}_3 = \hat{y}_1 + (t_2 - t_1)x_2 + (t_3 - t_2)x_3$, etc. Thus, $\hat{\boldsymbol{y}} = A\hat{\boldsymbol{x}}$ and our approximation is merely a linear interpolation using $\boldsymbol{t}$ and $\hat{\boldsymbol{y}}$.

C Synthetic optimization figure

See Fig. 9.

D Expanded figure for bias detection

See Fig. 10.

E Optimization evaluation details

Synthetic experiment. We create a synthetic model f to test our optimization algorithm. Consider a function $f(\boldsymbol{x}) = \sin(2x_0) + \cos(3x_1) + \beta^T \boldsymbol{x}_{2:}$ for $x_0, x_1 \in \mathbb{R}$, and $\beta, \boldsymbol{x}_{2:} \in \mathbb{R}^{d-2}$. We consider two utilities: 1) the model comparison utility with the first-order Taylor series approximation to f, i.e., $g_{\boldsymbol{x}_0}(\boldsymbol{x}) \equiv f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^T \boldsymbol{x}$, and 2) the least monotonic utility. It can be seen that the ground-truth best linear curve, for the model f above, with respect to these two utilities, will have directions along x_0 and x_1 respectively. Sinusoidal functions are indeed less monotonic than linear functions, and they also deviate away from the first-order Taylor series approximation, which is also linear. We verify that these directions are correctly found using our optimization algorithm as seen in Figure 9.

To more generally verify that GCP finds correct direction that maximizes the least monotonic utility, we simulated random model behaviors that are non-monotonic in certain directions, and compared the direction found with GCP with the ground-truth curves. For introducing non-monotonicity, a random set of polynomial functions were used, additionally constraining that the utilities of

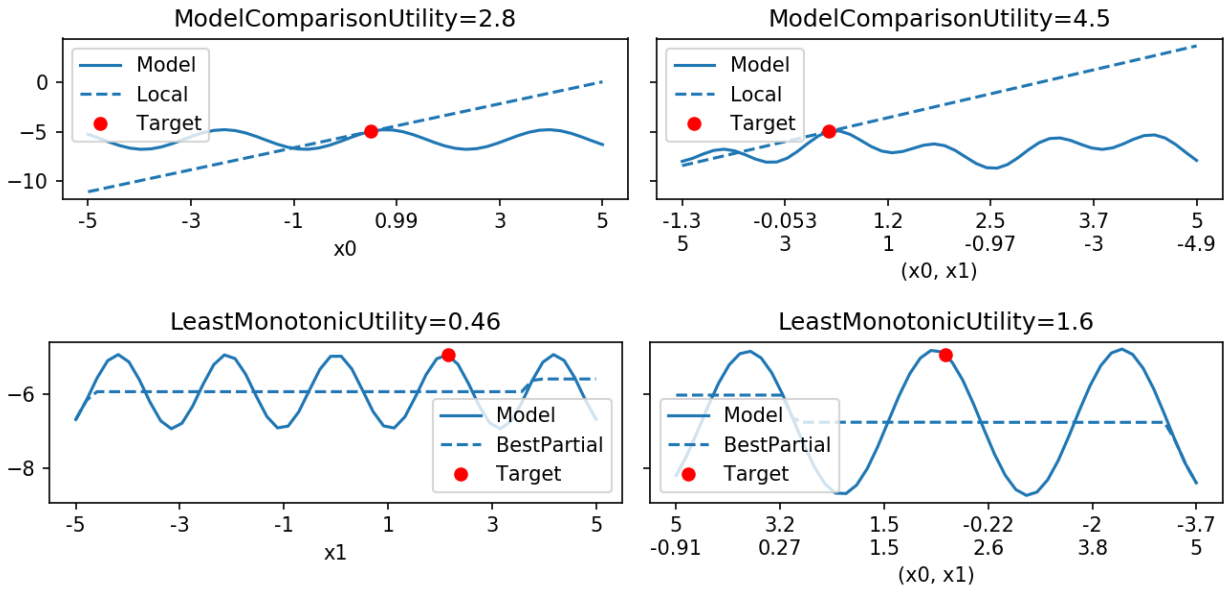
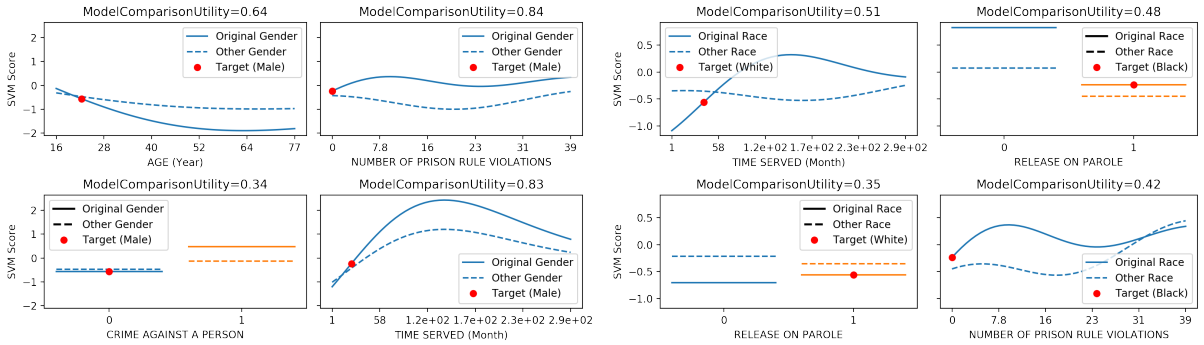


Figure 9: Our dependence plots when optimizing the synthetic function defined in E using the model comparison utility to a first-order Taylor series approximation (top) and using the least monotonic utility (bottom). In both cases, our optimization algorithm indeed finds the correct direction along x_0 and x_1.



(a) Flipping gender

(b) Flipping race

Figure 10: Our dependence plots showing top two biased features (rows) of two target instances (columns) for flipping (a) gender and (b) race. Notice that bias between groups is quite evident and is far from uniform; sometimes the bias even switches depending on the feature values (top left of subfigure (b)).

the curves along the ground-truth directions are non-zero and above certain threshold, while the utilities along the non-ground-truth directions are relatively small or almost zero.

Utility histogram. Another empirical way to check our optimization method is to randomly sample curves and compute their utilities; then we can compare to the utility of our optimized curve. We generate directions using a random forest on the loan application data (see section 5 for more data details). For interpretability, we restrict the curve parameter $\boldsymbol{v}$ to only have three non-zeros. We sample uniformly from all directions that have at most three non-zeros, i.e., $\boldsymbol{v} \in \{\boldsymbol{v} \in \mathbb{R}^d : \sum_i \mathbb{1}(v_i \neq 0) \leq 3\}$. We can see in the histogram of utility values (log scale) shown in Fig. 11 that the utility of our optimized curve (red line) is clearly better than the utility of random directions. In addition, we note that even if we do not find the global optimum, our optimized diagnostic curves can still be useful in highlighting interesting parts of the model—see use-case experiments. Thus, This shows that even though the optimization problem is quite difficult, we can perform well empirically.

Table 1: Parameter Values for Models

Model Name	Parameters
Decision tree	Max leaf nodes $\in \{5, 10, 20, 40\}$, max depth $\in \{1, 2, \cdots, 10\}$
Gradient boosted trees	Learning rate $\in \{0.05, 0.1, 0.2\}$, Number of estimators $\in \{25, 50, 100, 200, 500\}$
Deep NN	Max epoch = 1000, Learning rate = 0.0001, batch size $\in \{100, 200, 400\}$, two hidden layers of size 128 with relu activations and softmax final activation, ADAM optimizer and BCE loss.

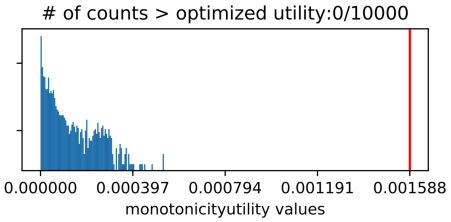


Figure 11: The utility found by our optimization (red line) is clearly finding a large value for utility compared to random directions (blue histogram with counts in log scale) demonstrating that our optimization method performs well empirically.

F More experiment details

Selecting model for loan prediction. The data used in this experiment is German loan application data.[1] This dataset has 7 numeric attributes and 13 categorical attributes ranging from the amount of the loan requested to the status of the applicant's checking account.

We train a random forest, a gradient boosted tree, and a deep neural network on only the numeric attributes (since monotonicity isn't well-defined for categorical attributes). We tune each model via cross validation using `scikit-learn`. We optimize each model over the parameters in Table 1 (where other parameters are defaults in `scikit-learn`).

Evaluating robustness under covariate shift in loan prediction. To simulate this setup, we split the German loan dataset based on amount: a training dataset with 884 users with `amount` $>$ 1,000 DMR and a separate unseen test dataset of 116 users with `amount` $\leq$ 1,000 DMR—note that these will give two different data distributions. We train via the deep neural network parameters and cross validation in Table 1.

Understanding out-of-sample behavior for traffic sign detection. We train a convolution neural network on German Traffic Sign Recognition Benchmark (GTSRB) dataset [29] which achieves 97% test accuracy. We consider directions based on five image transformations: rotate, blur, brighten, desaturate, and increase contrast. Each image transformation will create images outside of the training distribution—hence, we can view the behavior of the model outside of the training data.

[1] https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)