Model Agnostic Contrastive Explanations for Machine Learning Classification Models

Amit Dhurandhar[1], Pin-Yu Chen[1], Karthikeyan Shanmugam[1], Tejaswini Pedapati[1], Avinash Balakrishnan[1] and Ruchir Puri[2*]

December 14, 2018

Abstract

Recently, a method [5] was proposed to generate contrastive explanations for differentiable models such as deep neural networks, where one has complete access to the model. In this work, we propose a method, Model Agnostic Contrastive Explanations Method (MACEM), to generate contrastive explanations for *any* classification model assuming only oracle access, i.e., one is able to only query the class probabilities for a desired input. This allows us to generate contrastive explanations for not only neural networks, but models such as random forests, boosted trees and even arbitrary ensembles that are still heavily used when learning on structured data given their state-of-the-art performance in many domains [10]. Moreover, to obtain meaningful explanations on structured data we propose a principled approach to handle real and categorical features leading to novel formulations for computing pertinent positives and negatives that form the essence of a contrastive explanation. A detailed treatment of the different data types of this nature was not performed in the previous work, which assumed all features to be positive real valued with zero being indicative of the least interesting (i.e. feature being absent) value. This was a strong implicit assumption which of course may not true for different datasets.

1 Introduction

Given the wide spread use of deep networks [9] across various applications and their black box nature, explainability in artificial intelligence (XAI) has been one of the problems at the forefront in AI research recently [5, 7, 16, 21, 25]. Darpa's call for creating interpretable solutions [11] and the General Data Protection Regulation (GDPR) passed in Europe [33] which requires businesses to provide understandable justifications to their users for decisions that may affect them has made this need even more acute.

[*]1 and 2 indicate affiliation to IBM Research and IBM Watson Group respectively.

There have been many (posthoc) interpretability methods proposed to interpret decisions of neural networks [1, 5, 7, 16, 24, 30] which assume complete access to the model. Locally interpretable model-agnostic explanation method (LIME) [25] is amongst the few that can provide local explanations for any model with just oracle access. In other words, LIME just needs to be able to query the classification model and based on its outputs can generate an explanation. This is an extremely attractive feature as it can be used in settings where the model owner may not want to expose the inner details of the model but may desire local explanations using say a remote service. Another application is that the method can be used to interpret decisions not just of neural networks but other models such as random forests, boosted trees and ensembles of heterogeneous models which are known to perform quite well in many domains that use structured data [10].

In this paper, we thus propose the model agnostic contrastive explanations method (MACEM) that requires only oracle access to the classification model with particular focus on structured data. Structured data can be composed of real and categorical features, and we provide a principled way of creating contrastive explanations for such data. Contrastive explanations are a rich form of explanation where one conveys not only what is (minimally) sufficient to justify the class of an input i.e. pertinent positives (PPs), but also what should be (minimally) necessarily absent to maintain the original classification i.e. pertinent negatives (PNs) [5]. Such explanations are commonly used in social settings as well as in domains such as medicine and criminology [13]. For example, a patient with symptoms of cough, cold and fever (PPs) could have flu or pneumonia. However, the absence of chills or mucous (PNs) would indicate that the person has flu rather than pneumonia. Thus, in addition to the symptoms that were present, the symptoms that are absent are also critical in arriving at a decision.

In previous works [5], a method to produce such explanations was proposed. However, the method was restricted to differentiable models such as deep neural networks and strong (implicit) assumptions were made in terms of the semantics of the data used to train the models and obtain explanations. In particular, following are the key differences between our current and the prior work:

- **Gradients not available:** In this work we want to create contrastive explanations with only oracle or query access to the classification model. This is a significant step given that the prior work a) assumed complete access to the model and b) could be used only for differentiable models like deep neural networks. Our method can be used for any classification model that may be differentiable or non-differentiable (viz. decision trees, forests, ensembles), where we estimate gradients (with theoretically bounded bias) using only oracle access.

- **Adaptively estimating and using base values:** To compute PPs and PNs one needs to know what it means for a feature to be absent. In other words, what value for a feature indicates that there is no signal or is essentially the least interesting value for that feature. We refer to such values as *base values*. In the prior work the value 0 for a feature was considered as the base value, with positive deviation from it being indicative of more interesting values or values that have more signal. However, this may not be the case for many features especially those that are categorical. Ideally, the user should provide us with these values, however in many situations this may not be the case. In this paper we adapt our methods to utilize these given base values and also propose ways to estimate them from the data using best judgment in situations that they are not provided.

- **Handling categorical features:** In the prior work all features were considered to be real valued and no special consideration was given to handle categorical features. However, in this work we remain cognizant to the fact that categorical features are fundamentally different than real valued ones and propose principled approaches to handle them given our form of explanations.

- **Computing PPs and PNs:** Given the above differences we propose new ways of computing PPs and PNs that are consistent with their intuitive definitions mentioned before. Although conceptually similar to the previous work, the details of the methods to compute these are quite different as can be witnessed in the later sections.

2 Related Work

Trust and transparency of AI systems has received a lot of attention recently [11]. Explainability is considered to be one of the cornerstones for building trustworthy systems and has been of particular focus in the research community [6,18]. Researchers are trying to build better performing interpretable models [2,4,7,14,31,32] as well as improved methods to understand black box models such as deep neural networks [1,5,25].

The survey [21] which is mainly focused on deep learning explainability methods looks broadly at i) prototype selection methods [22,23] to explain a particular class and ii) methods that highlight relevant features for a given input [1,16,25,28]. There are other works that fall under (i) such as [12,15] as well as those that fall under (ii) for vision [24,29,30] and NLP applications [17]. Most of these works though do not provide contrastive explanations in a model agnostic setting.

There are also interesting works which try to quantify interpretability [6,27] and suggest methods for doing so.

Two of the most relevant recent works besides [5] which we have already contrasted with are [26,34]. In [26], the authors try to find sufficient conditions to justify a classification that are global in nature. For example, the presence of the word "bad" in a sentence would automatically indicate negative sentiment irrespective of the other words. As such, they do not find input specific minimally sufficient values that would maintain the classification or minimal values that

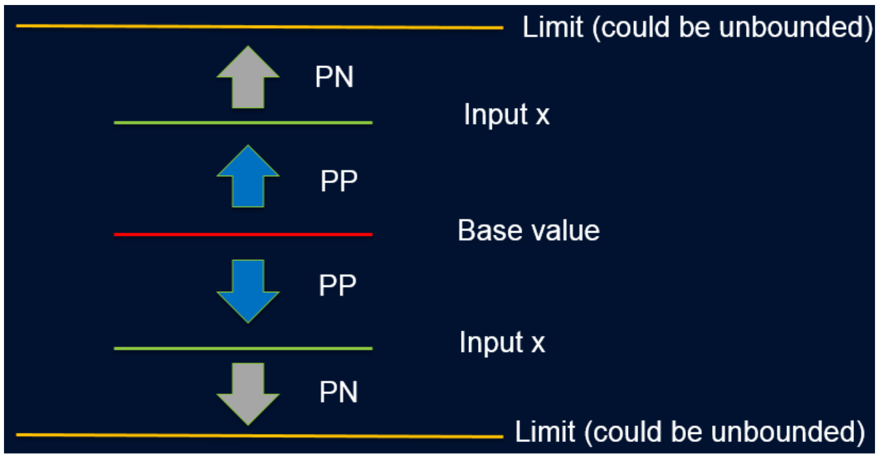

Figure 1: Above we see a visual representation of PPs and PNs. Depending on where the input x lies relative to the base value the PPs and PNs will be above or below the base value. However, in both cases the search moves away from the base value for both PPs and PNs.

would change classification. Such global anchors also may not always be present in the data that one is interested in. The other work [34] tries to provide (stable) suggestions more than local explanations for decisions based on a neural network. Moreover, the approach is restricted to neural networks using rectified linear units and is feasible primarily for smallish to medium sized neural networks in asymmetric binary settings, where suggestions are sought for a specific class (viz. loan rejected) and not the other (viz. loan accepted).

3 MACEM Method

Let $\mathcal{X}$ denote the feasible data space and let $(\mathbf{x}_0, t_0)$ denote an example $\mathbf{x}_0 \in \mathcal{X}$ and its inferred class label t_0 obtained from a black-box classification model. The modified example $\mathbf{x} \in \mathcal{X}$ based on $\mathbf{x}_0$ is defined as $\mathbf{x} = \mathbf{x}_0 + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is a perturbation applied to $\mathbf{x}_0$. Our method of finding pertinent positives/negatives is formulated as an optimization problem over the perturbation variable $\boldsymbol{\delta}$ that is used to explain the model's prediction results. We denote the prediction of the model on the example $\mathbf{x}$ by $\text{Pred}(\mathbf{x})$, where $\text{Pred}(\cdot)$ is any function that outputs a vector of confidence scores over all classes, such as the log value of prediction probability.

3.1 Pertinent Positives

Assume an example $\mathbf{x}_0$ has d features each with base values $\{b_i\}_{i=1}^d$. Let Δ_{PP} denote the space $\{\boldsymbol{\delta} : |\mathbf{x}_0 + \boldsymbol{\delta} - \mathbf{b}| \preceq |\mathbf{x}_0 - \mathbf{b}| \text{ and } \mathbf{x}_0 + \boldsymbol{\delta} \in \mathcal{X}\}$, where $\mathbf{b} = [b_1, \ldots, b_d]$, and $|\cdot|$ and $\preceq$ implies element-wise absolute value and inequality,

Algorithm 1 Model Agnostic Contrastive Explanations Method (MACEM)

```
Input: example $(\mathbf{x}_0, t_0)$, black box model $\mathcal{M}$ and optionally base values $\mathbf{b}$,
allowed space $\mathcal{X}$, $(\gamma > 0)$ an autoencoder $AE$.
1) Solve and obtain,
$\boldsymbol{\delta}^{\text{pos}} \leftarrow \text{argmin}_{\boldsymbol{\delta} \in \Delta_{PP}}\ c \cdot f_\kappa^{\text{pos}}(\mathbf{x}_0, \boldsymbol{\delta}) + \beta \|\mathbf{x}_0 + \boldsymbol{\delta} - \mathbf{b}\|_1 + \|\mathbf{x}_0 + \boldsymbol{\delta} - \mathbf{b}\|_2^2 +$
    $\gamma \|\mathbf{x}_0 + \boldsymbol{\delta} - \text{AE}(\mathbf{x}_0 + \boldsymbol{\delta})\|_2^2.$
2) Solve and obtain,
$\boldsymbol{\delta}^{\text{neg}} \leftarrow \text{argmin}_{\boldsymbol{\delta} \in \Delta_{PN}}\ c \cdot f_\kappa^{\text{neg}}(\mathbf{x}_0, \boldsymbol{\delta}) + \beta \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2 + \gamma \|\mathbf{x}_0 + \boldsymbol{\delta} - \text{AE}(\mathbf{x}_0 + \boldsymbol{\delta})\|_2^2.$

return $\boldsymbol{\delta}^{\text{pos}}$ and $\boldsymbol{\delta}^{\text{neg}}$. {Explanation: The input $\mathbf{x}_0$ would still be classified
into class $t_0$ even if its feature values were (closer to base values as in) $\boldsymbol{\delta}^{\text{pos}}$.
However, its class would change if it were perturbed (away from base values)
by $\boldsymbol{\delta}^{\text{neg}}$, i.e., if the input became $\mathbf{x}_0 + \boldsymbol{\delta}^{\text{neg}}$. Code at https://github.ibm.
com/tejaswinip/CEM }
```

respectively. To solve for PP, we propose the following problem formulation:

$$\min_{\boldsymbol{\delta} \in \Delta_{PP}} \ c \cdot f_\kappa^{\text{pos}}(\mathbf{x}_0, \boldsymbol{\delta}) + \beta \|\mathbf{x}_0 + \boldsymbol{\delta} - \mathbf{b}\|_1 + \|\mathbf{x}_0 + \boldsymbol{\delta} - \mathbf{b}\|_2^2 + \tag{1}$$

$$\gamma \|\mathbf{x}_0 + \boldsymbol{\delta} - \text{AE}(\mathbf{x}_0 + \boldsymbol{\delta})\|_2^2. \tag{2}$$

The first term $f_\kappa^{\text{pos}}(\mathbf{x}_0, \boldsymbol{\delta})$ is a designed loss function that encourages the modified example $\mathbf{x} = \mathbf{x}_0 + \boldsymbol{\delta}$ relative to the base value vector $\mathbf{b}$, defined as $\mathbf{x} - \mathbf{b}$, to be predicted as the same class as the original top-1 label $t_0 = \arg\max_i [\text{Pred}(\mathbf{x}_0)]_i$. The loss function is defined as:

$$f_\kappa^{\text{pos}}(\mathbf{x}_0, \boldsymbol{\delta}) = \max\{\max_{i \neq t_0} [\text{Pred}(\mathbf{x}_0 + \boldsymbol{\delta})]_i - [\text{Pred}(\mathbf{x}_0 + \boldsymbol{\delta})]_{t_0}, -\kappa\}. \tag{3}$$

The loss function f_κ^{pos} is a hinge-like loss and the term $\kappa \geq 0$ controls the gap between $[\text{Pred}(\mathbf{x}_0 + \boldsymbol{\delta})]_{t_0}$ and the other most probable class. In particular, the loss attains its minimal value when $[\text{Pred}(\mathbf{x}_0 + \boldsymbol{\delta})]_{t_0}$ is κ larger than $\max_{i \neq t_0} [\text{Pred}(\mathbf{x}_0 + \boldsymbol{\delta})]_i$. The parameter $c \geq 0$ (1) is the regularization coefficient associated with f_κ^{pos}. The second and third terms in (1) are jointly called the elastic-net regularizer [35], which aids in selecting a set of highly relevant features from $\mathbf{x} - \mathbf{b}$, and the parameter $\beta \geq 0$ controls the sparsity of the vector $\mathbf{x} - \mathbf{b}$. In other words, if the i-th element of $\mathbf{x} - \mathbf{b}$ is 0, this means the i-th is not significant for constituting PP.

Optionally, an autoencoder also maybe learned on the data which could be used to further direct the search so that we produce realistic or high probability $\mathbf{x}$.

3.2 Pertinent Negatives

Analogous to PP, for PN let Δ_{PN} denote the space $\{\boldsymbol{\delta} : |\mathbf{x}_0 + \boldsymbol{\delta} - \mathbf{b}| \succ |\mathbf{x}_0 - \mathbf{b}| \text{ and } \mathbf{x}_0 + \boldsymbol{\delta} \in \mathcal{X}\}$. To solve for PN, we propose the following problem formulation:

$$\min_{\boldsymbol{\delta} \in \Delta_{PN}} \; c \cdot f_\kappa^{\text{neg}}(\mathbf{x}_0, \boldsymbol{\delta}) + \beta \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2 + \gamma \|\mathbf{x}_0 + \boldsymbol{\delta} - \text{AE}(\mathbf{x}_0 + \boldsymbol{\delta})\|_2^2, \tag{4}$$

where

$$f_\kappa^{\text{neg}}(\mathbf{x}_0, \boldsymbol{\delta}) = \max\{[\text{Pred}(\mathbf{x}_0 + \boldsymbol{\delta})]_{t_0} - \max_{i \neq t_0} [\text{Pred}(\mathbf{x}_0 + \boldsymbol{\delta})]_i, -\kappa\}. \tag{5}$$

In other words, for PN, we aim to find the least modified changes in $\boldsymbol{\delta} \in \Delta_{PN}$, evaluated by the elastic-net loss on $\boldsymbol{\delta}$, such that its addition to $\mathbf{x}_0$ leads to a different top-1 prediction from t_0,

3.3 Method Details

We now describe the details of how the optimization of the above objectives is implemented along with estimation and modeling of certain key aspects.

3.3.1 Optimization Procedure

If the gradient of the designed loss functions for PP and PN with respect to $\boldsymbol{\delta}$ can be obtained, then one can readily apply We apply a projected fast iterative shrinkage-thresholding algorithm (FISTA) [3] to solve problems (4) and (1). However, in our black-box setting such gradient is inadmissible. We will illustrate how to get around this problem in the following subsection.

Here we first illustrate how FISTA solves for PP and PN, assuming the gradient is available. FISTA is an efficient solver for optimization problems involving L_1 regularization. Take pertinent negative as an example, let $g(\boldsymbol{\delta}) = f_\kappa^{\text{neg}}(\mathbf{x}_0, \boldsymbol{\delta}) + \|\boldsymbol{\delta}\|_2^2$ denote the objective function of (4) without the L_1 regularization term. Given the initial iterate $\boldsymbol{\delta}^{(0)} = \mathbf{0}$, projected FISTA iteratively updates the perturbation I times by

$$\boldsymbol{\delta}^{(k+1)} = \Pi_{\Delta_{PN}}\{S_\beta(\mathbf{y}^{(k)} - \alpha_k \nabla g(\mathbf{y}^{(k)}))\}; \tag{6}$$

$$\mathbf{y}^{(k+1)} = \Pi_{\Delta_{PN}}\{\boldsymbol{\delta}^{(k+1)} + \frac{k}{k+3}(\boldsymbol{\delta}^{(k+1)} - \boldsymbol{\delta}^{(k)})\}, \tag{7}$$

where $\Pi_{\Delta_{PN}}$ denotes the vector projection onto the set Δ_{PN}, α_k is the step size, $\mathbf{y}^{(k)}$ is a slack variable accounting for momentum acceleration with $\mathbf{y}^{(0)} = \boldsymbol{\delta}^{(0)}$, and $S_\beta : \mathbb{R}^p \mapsto \mathbb{R}^p$ is an element-wise shrinkage-thresholding function defined as

$$[S_\beta(\mathbf{z})]_i = \begin{cases} \mathbf{z}_i - \beta, & \text{if } \mathbf{z}_i > \beta; \\ 0, & \text{if } |\mathbf{z}_i| \le \beta; \\ \mathbf{z}_i + \beta, & \text{if } \mathbf{z}_i < -\beta, \end{cases} \tag{8}$$

for any $i \in \{1, \ldots, d\}$. The final perturbation $\boldsymbol{\delta}^{(k^*)}$ for pertinent negative analysis is selected from the set $\{\boldsymbol{\delta}^{(k)}\}_{k=1}^{I}$ such that $f_\kappa^{\text{neg}}(\mathbf{x}_0, \boldsymbol{\delta}^{(k^*)}) = 0$ and $k^* = \arg\min_{k \in \{1,\ldots,I\}} \beta \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2$. A similar projected FISTA optimization approach is applied to pertinent positive analysis.

3.3.2 Gradient Estimation

In the black-box setting, in order to balance the model query complexity and algorithmic convergence rate using zeroth-order optimization, in this paper we use a two-point evaluation based gradient estimator averaged over q different random directions [8,19,20]. Specifically, given a scalar function $f(\cdot)$, its gradient at a point $\mathbf{x} \in \mathbb{R}^d$ is estimated by

$$\widehat{\nabla} f(\mathbf{x}) = \frac{d}{q\mu} \sum_{j=1}^{q} \frac{f(\mathbf{x} + \mu \mathbf{u}_j) - f(\mathbf{x})}{\mu} \cdot \mathbf{u}_j, \tag{9}$$

where $\{\mathbf{u}_j\}_{j=1}^q$ is a set of i.i.d. random directions drawn uniformly from a unit sphere, and $\mu > 0$ is a smoothing parameter.

The estimation error between $\widehat{\nabla} f(\mathbf{x})$ and the true gradient $\nabla f(\mathbf{x})$ can be analyzed through a smoothed function $f_\mu(\mathbf{x}) = \mathbb{E}_{\mathbf{u} \in U_b}[f(\mathbf{x} + \mu \mathbf{u})]$, where U_b is a uniform distribution over the unit Euclidean ball. Assume f is an L-smooth function, that is, its gradient ∇f is L-Lipschitz continuous. It has been shown in [20, Lemma 2] that $\widehat{\nabla} f(\mathbf{x})$ is an unbiased estimator of the gradient $\nabla f_\mu(\mathbf{x})$, i.e., $\mathbb{E}_{\mathbf{u}}[\widehat{\nabla} f(\mathbf{x})] = \widehat{\nabla} f(\mathbf{x})$. Moreover, using the bounded error between f and f_μ as specified in [20, Lemma 1], one can show that the mean squared error between $\widehat{\nabla} f(\mathbf{x})$ and $\nabla f(\mathbf{x})$ is upper bounded by

$$\mathbb{E}_{\mathbf{u}}[\|\widehat{\nabla} f(\mathbf{x}) - \nabla f(\mathbf{x})\|_2^2] \le O\left(\frac{q+d}{q}\right) \|\nabla f(\mathbf{x})\|_2^2 + O(\mu^2 L^2 d^2). \tag{10}$$

3.3.3 Determining Base Values

As mentioned before, ideally, we would want base values as well as allowed ranges or limits for all features be specified by the user before running our method to obtain contrastive explanations. This should in all likelihood provide the most useful explanations. However, this may not always be feasible given the dimensionality of the data and the level of expertise of the user. In such situations we compute base values using our best judgment.

For real valued features, we set the base value to be the median value of the feature. This without knowing anything more about the feature is possibly the least interesting value for that feature. Moreover, medians are known to be robust to outliers and are thus preferable to using means. They also are a point estimate that has minimum L_1 error w.r.t. the values for that feature. Medians also make intuitive sense where for sparse features 0 would rightly be chosen as the base value as opposed to some other value which would be the case for means.

For categorical features, we set the base value to be the mode or most frequent value for that feature. Here again we hypothesize that a more uniquely occurring value is probably more interesting to the user. For example in a dataset containing health records most people will probably not have cancer and so having cancer is something that should stand out as it indicates a state away

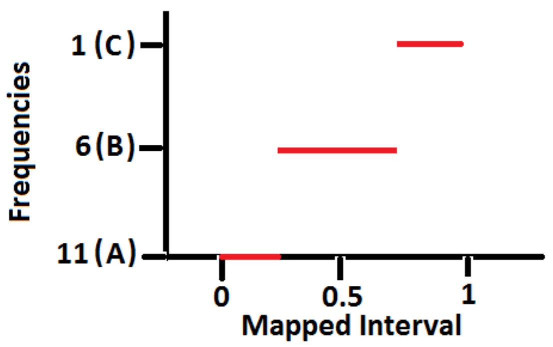

Figure 2: Above we see a categorical feature taking three values A, B and C with frequencies 11, 6 and 1 respectively as indicated on the vertical axis. Our mapping function in equation 11 for FMA maps these frequencies and hence the categorical values to 0, 0.5 and 1 in the $[0, 1]$ interval. The red horizontal lines depict the function $h(.)$ showcasing the range of values that map back to either A, B or C.

from the norm. Such states or behaviors we believe carry information that is more likely to surprise the user and draw attention, which could be a prelude to further actions.

3.3.4 Modeling Categorical Features

Given that categorical features do not impose an explicit ordering like real features do along with the fact that only the observed values have semantic meaning, there is a need to be model them differently than real features when obtaining explanations. We now present two different strategies for handling categorical features in our framework along with a formal interpretation of what these strategies are actually accomplishing.

3.3.4.1 Frequency Map Approach (FMA)

In this approach we want to directly leverage the optimization procedure described above where we need to define an ordered set/interval in which to find the perturbations δ for both PPs and PNs.

Mapping: As described above for categorical features we set the base value to be the mode of the values that occur. Given this a natural ordering can be created based on the frequencies of the different values. Thus, the least frequent value would be considered to be the farthest from the base value. Based on this we can map the k discrete values $v_{i1}, ..., v_{ik}$ of the i^{th} feature occurring with frequencies/counts $c_{i1}, ..., c_{ik}$ to real values $r_{i1}, ..., r_{ik}$ respectively in the $[0, 1]$ interval using the following mapping for any $j \in \{1, ..., k\}$:

$$r_{ij} = \frac{c_{\max} - c_{ij}}{c_{\max} - 1} \tag{11}$$

where $c_{\max} = \max_{j \in \{1,...,k\}} c_{ij}$. This maps the discrete value with the highest frequency to 0 making it the base value, while all other values with decreasing frequencies lie monotonically away from 0. Every candidate value has to have a frequency of at least 1 and so every discrete value gets mapped to the $[0, 1]$ interval. We divide by $c_{\max} - 1$, rather than $c_{\max} - c_{\min}$, where $c_{\min} = \min_{j \in \{1,...,k\}} c_{ij}$ since, we do not want values that occur with almost equal frequency to be pushed to the edges of the interval as based on our modeling they are of similar interest. For example, a feature may just have two values occurring with frequencies of 50 and 49. We want to be able to switch between these values easily when finding PPs and PNs rather than mapping them to two ends of the interval i.e. 50 to 0 and 49 to 1. This is particularly important when we notice the fact that a different categorical feature may have more extreme frequencies such as 50 and 10, and we want to maintain this relative difference between features in our mapping.

Method and Interpretation: Based on equation 11, we run our algorithm for categorical features in the interval $[0, 1]$, where every time we query the model we round the $\mathbf{x}_0 + \boldsymbol{\delta}$ to the closest r_{ij} so that a *valid* categorical value v_{ij} can be mapped back to and sent as part of the query input.

The question now is what are we exactly doing in the mathematical sense. It turns out that rather than optimizing $f_\kappa^{\text{neg}}(\mathbf{x}_0, \boldsymbol{\delta})$ or $f_\kappa^{\text{pos}}(\mathbf{x}_0, \boldsymbol{\delta})$, we are optimizing $f_\kappa^{\text{neg}}(h(\mathbf{x}_0, \boldsymbol{\delta}))$ or $f_\kappa^{\text{pos}}(h(\mathbf{x}_0, \boldsymbol{\delta}))$ respectively, where $h(.)$ is the identity map for real features, that is $f_\kappa^{\text{neg}}(h(\mathbf{x}_0, \boldsymbol{\delta})) = f_\kappa^{\text{neg}}(\mathbf{x}_0, \boldsymbol{\delta})$ and $f_\kappa^{\text{pos}}(h(\mathbf{x}_0, \boldsymbol{\delta})) = f_\kappa^{\text{pos}}(\mathbf{x}_0, \boldsymbol{\delta})$, but a step function defined over the $[0, 1]$ interval for categorical features. Let $h_i(.)$ denote the application of the function $h(.)$ to the categorical feature i. If $\mathbf{x} = \mathbf{x}_0 + \boldsymbol{\delta}$ and $\mathbf{x}_i$ denotes the value of the feature in the mapped $[0, 1]$ interval then,

$$h_i(\mathbf{x}_0, \boldsymbol{\delta}) = v_{ij}, \text{ if } |\mathbf{x}_i - r_{ij}| \leq |\mathbf{x}_i - r_{im}| \ \forall m \in \{1, ..., k\} \text{ and } m \neq j \tag{12}$$

where $|.|$ denotes absolute value. An example function $h(.)$ is depicted in figure 2, where we see how the mapping occurs and how values in the $[0, 1]$ interval are mapped back to valid categorical values by rounding to the closest r_{ij}.

3.3.4.2 Simplex Sampling Approach (SSA)

In this method of handling categorical variables, we will assume that a one-hot encoding of the input. Let $\mathbf{x} = [\mathbf{x}_C \mathbf{x}_R]$ be the input feature vector where $\mathbf{x}_C$ denotes the categorical part while $\mathbf{x}_R$ denote the set of real features. Let there be C categorical features in total. Then for all $c \in [1 : C], x_c \in [1 : d_c]$ where x_c is the c-th categorical variable and it takes one of d_c values. Note that we imply no ordering amongst the d_c values. Generically they are assumed to have distinct integer values from 1 till d_c.

We assume that input $\mathbf{x}$ is processed into into $\tilde{\mathbf{x}} = [\tilde{\mathbf{x}}_C \tilde{\mathbf{x}}_R]$ where $\tilde{\mathbf{x}}_R = \mathbf{x}_R$ while $\tilde{\mathbf{x}}_C \in \mathbb{R}^{1 \times \prod_{c \in C} d_c}$. And each component $\tilde{\mathbf{x}}_c \in \mathbb{R}^{1 \times d_c}$ is set to $\mathbf{e}_i$, the canonical unit vector with 1 in the i-th coordinate, if x_c takes the value i.

Now, we provide an interpretation when every categorical component c lies in the d_c dimensional simplex, i.e. when $\tilde{\mathbf{x}}_c \in \Delta_{d_c}$. Here, Δ_N denotes the N-dimensional simplex. The actual function can be evaluated only on the inputs where each categorical component takes values from one of the corner points on the simplex, namely $\mathbf{e}_i,\ i \in [1:d_c]$. Therefore, we interpolate the function when $\tilde{\mathbf{x}}_c$ is assigned a real vector in the simplex.

Let $f(\cdot)$ that captures the soft output of the classifier when the one-hot encoded categorical variables take the values at the corner points of their respective simplices. Now, we extend the definition of f as follows:

$$f([\tilde{\mathbf{x}}_C \mathbf{x}_R]) = \mathbb{E}_{\mathbf{e}_{i_c} \sim \tilde{\mathbf{x}}_c,\ \forall c \in [1:C]} \left[f(\mathbf{e}_{i_1}, \ldots \mathbf{e}_{i_c}, \ldots \mathbf{e}_{i_C}, \mathbf{x}_R) \right]. \tag{13}$$

Essentially, we sample the c-th unit vector from the distribution represented by $\tilde{\mathbf{x}}_\mathbf{c} \in \Delta_{d_c}$ on the simplex independent of other categorical variable. The function value value is the expected value of the functional evaluated on unit vectors obtained from this product distribution along with the fixed real coordinates $\mathbf{x}_R$.

When we perform the gradient descent as a part of algorithm 1, we actually do a projected gradient descent for $\tilde{\mathbf{x}}_C$ in the product of simplices $\Delta_{d_1} \times \ldots \Delta_{d_c}$. We cannot evaluate the function exactly, hence we can average over a certain number of samples drawn from the product distribution for every function evaluation on a candidate $\tilde{\mathbf{x}}$.

3.3.4.2 Approach Tradeoffs

As such the SSA strategy has stronger theoretical grounding, however from a practical standpoint it requires a lot of additional averaging through sampling for every function evaluation along with repeated projections to the simplices defined for every categorical feature during gradient descent to optimize the objective in Algorithm 1.

Additionally, FMA can take more general format of inputs as they don't need to be one-hot-encoded which guards against further explosion of the feature space which could potentially result from categorical features having many distinct values.

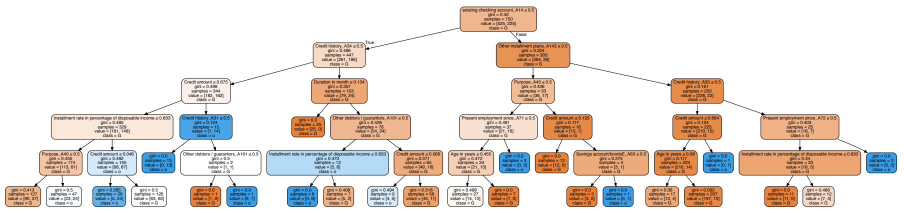
existing checking account_A14 ≤ 0.5 gini = 0.42 samples = 750 value = [525, 225] class = G
True
False
Credit history_A34 ≤ 0.5 gini = 0.486 samples = 447 value = [261, 186] class = G
Other installment plans_A143 ≤ 0.5 gini = 0.224 samples = 303 value = [264, 39] class = G
Credit amount ≤ 0.675 gini = 0.498 samples = 344 value = [182, 162] class = G
Duration in month ≤ 0.134 gini = 0.357 samples = 103 value = [79, 24] class = G
Purpose_A43 ≤ 0.5 gini = 0.436 samples = 53 value = [36, 17] class = G
Credit history_A33 ≤ 0.5 gini = 0.161 samples = 250 value = [228, 22] class = G
Installment rate in percentage of disposable income ≤ 0.833 gini = 0.495 samples = 329 value = [181, 148] class = G
Credit history_A31 ≤ 0.5 gini = 0.124 samples = 15 value = [1, 14] class = o
gini = 0.0 samples = 25 value = [25, 0] class = G
Other debtors / guarantors_A101 ≤ 0.5 gini = 0.426 samples = 78 value = [54, 24] class = G
Present employment since_A71 ≤ 0.5 gini = 0.491 samples = 37 value = [21, 16] class = G
Credit amount ≤ 0.135 gini = 0.117 samples = 16 value = [15, 1] class = G
Credit amount ≤ 0.664 gini = 0.124 samples = 225 value = [210, 15] class = G
Present employment since_A72 ≤ 0.5 gini = 0.403 samples = 25 value = [18, 7] class = G
Purpose_A40 ≤ 0.5 gini = 0.455 samples = 174 value = [113, 61] class = G
Credit amount ≤ 0.046 gini = 0.492 samples = 155 value = [68, 87] class = o
gini = 0.0 samples = 13 value = [0, 13] class = o
Other debtors / guarantors_A101 ≤ 0.5 gini = 0.5 samples = 2 value = [1, 1] class = G
Installment rate in percentage of disposable income ≤ 0.833 gini = 0.473 samples = 13 value = [5, 8] class = o
Credit amount ≤ 0.066 gini = 0.371 samples = 65 value = [49, 16] class = G
Age in years ≤ 0.455 gini = 0.472 samples = 34 value = [21, 13] class = G
gini = 0.0 samples = 3 value = [0, 3] class = o
gini = 0.0 samples = 12 value = [12, 0] class = G
Savings account/bondsE_A63 ≤ 0.5 gini = 0.375 samples = 4 value = [3, 1] class = G
Age in years ≤ 0.08 gini = 0.117 samples = 224 value = [210, 14] class = G
gini = 0.0 samples = 1 value = [0, 1] class = o
Installment rate in percentage of disposable income ≤ 0.833 gini = 0.34 samples = 23 value = [18, 5] class = G
gini = 0.0 samples = 2 value = [0, 2] class = o
gini = 0.413 samples = 127 value = [90, 37] class = G
gini = 0.5 samples = 47 value = [23, 24] class = o
gini = 0.285 samples = 29 value = [5, 24] class = o
gini = 0.5 samples = 126 value = [63, 63] class = G
gini = 0.0 samples = 1 value = [1, 0] class = G
gini = 0.0 samples = 1 value = [0, 1] class = o
gini = 0.0 samples = 6 value = [0, 6] class = o
gini = 0.408 samples = 7 value = [5, 2] class = G
gini = 0.494 samples = 9 value = [4, 5] class = o
gini = 0.316 samples = 56 value = [45, 11] class = G
gini = 0.499 samples = 27 value = [14, 13] class = G
gini = 0.0 samples = 7 value = [7, 0] class = G
gini = 0.0 samples = 3 value = [3, 0] class = G
gini = 0.0 samples = 1 value = [0, 1] class = o
gini = 0.36 samples = 17 value = [13, 4] class = G
gini = 0.092 samples = 207 value = [197, 10] class = G
gini = 0.0 samples = 11 value = [11, 0] class = G
gini = 0.486 samples = 12 value = [7, 5] class = G

Figure 3: We see above a decision tree built for the German Credit dataset.

4 Experimental Results

Our initial results indicate that there is roughly 75% overlap between features identified by our PPs and the relevant path of the points lying in that class.

Moreover, PNs show negative correlation with the true paths which makes sense since changes to low level attributes will result in changes to the class in a minimal sense. The PPs generally have positive correlation.

	SM-3	SM-5	SM-7	SM-9
Standard	73.15($\pm$ 0.7)	75.78($\pm$0.5)	78.76($\pm$0.35)	79.9($\pm$0.34)
ConfWeight	76.27 ($\pm$0.48)	78.54 ($\pm$0.36)	**81.46**($\pm$0.50)	82.09 ($\pm$0.08)
Distillation	65.84($\pm$0.60)	70.09 ($\pm$0.19)	73.4($\pm$0.64)	77.30 ($\pm$0.16)
ProfWeight$^{\text{ReLU}}$	**77.52** ($\pm$0.01)	78.24($\pm$0.01)	80.16($\pm$0.01)	81.65 ($\pm$0.01)
ProfWeight$^{\text{AUC}}$	76.56 ($\pm$0.62)	**79.25**($\pm$0.36)	**81.34**($\pm$0.49)	**82.42** ($\pm$0.36)

Table 1: Averaged accuracies (%) of simple model trained with various weighting methods and distillation. The complex model achieved 84.5% accuracy. Weighting methods that average confidence scores of higher level probes perform the best or on par with the best in all cases. In each case, the improvement over the unweighted model is about $3 - 4\%$ in test accuracy. Distillation performs uniformly worse in all cases.

5 Discussion

In this paper we provided a model agnostic black box explanation method specifically tailored for structured data that is able to handle real as well as categorical features in a meaningful manner. We see that our method is quantitatively as well as qualitatively superior to other black box explainability approaches in this regime.

In the future, we would like to extend our approach here to be applicable to also unstructured data. In a certain sense, the current approach could be applied to such data if it is already vectorized or the text is embedded in a feature space. In such cases, although recovering a semantically correct sentence would be hard, one could identify important words or phrases which a language model or human could use as the basis for creating a valid sentence. In the text domain one could also envision using sampling approaches developed in the multi-armed bandit literature to produce meaningful explanations as an alternative to zeroth order optimization as we have done for structured data.

References

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[2] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpreting blackbox models via model extraction. *arXiv preprint arXiv:1705.08504*, 2017.

[3] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[4] 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*, KDD '15, pages 1721–1730, New York, NY, USA, 2015. ACM.

[5] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems*, 2018.

[6] Amit Dhurandhar, Vijay Iyengar, Ronny Luss, and Karthikeyan Shanmugam. Tip: Typifying the interpretability of procedures. *arXiv preprint arXiv:1706.02952*, 2017.

[7] Amit Dhurandhar, Karthikeyan Shanmugam, Ronny Luss, and Peder Olsen. Improving simple models with confidence profiles. In *Advances in Neural Information Processing Systems*, 2018.

[8] John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.

[10] Ben Gorman. A kaggle master explains gradient boosting. In *Kaggle blog*, 2017.

[11] David Gunning. Explainable artificial intelligence (xai). In *Defense Advanced Research Projects Agency*, 2017.

[12] Karthik Gurumoorthy, Amit Dhurandhar, and Guillermo Cecchi. Protodash: Fast interpretable prototype selection. *arXiv preprint arXiv:1707.01212*, 2017.

[13] Amy Herman. Are you visually intelligent? what you don't see is as important as what you do see. *Medical Daily*, 2016.

[14] Tsuyoshi Idé and Amit Dhurandhar. Supervised item response models for informative prediction. *Knowl. Inf. Syst.*, 51(1):235–257, April 2017.

[15] Been Kim, Rajiv Khanna, and Oluwasanmi Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *In Advances of Neural Inf. Proc. Systems*, 2016.

[16] Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. In *Intl. Conference on Learning Representations (ICLR)*, 2018.

[17] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*, 2016.

[18] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[19] Sijia Liu, Jie Chen, Pin-Yu Chen, and Alfred O Hero. Zeroth-order online alternating direction method of multipliers: Convergence analysis and applications. *AISTATS*, 2018.

[20] Sijia Liu, Bhavya Kailkhura, Pin-Yu Chen, Paishun Ting, Shiyu Chang, and Lisa Amini. Zeroth-order stochastic variance reduction for nonconvex optimization. *NIPS*, 2018.

[21] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 2017.

[22] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems*, pages 3387–3395, 2016.

[23] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*, 2016.

[24] Jose Oramas, Kaili Wang, and Tinne Tuytelaars. Visual explanation by interpretation: Improving visual feedback capabilities of deep neural networks. In *arXiv:1712.06302*, 2017.

[25] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, 2016.

[26] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[27] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. In *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[28] Su-In Lee Scott Lundberg. Unified framework for interpretable methods. In *In Advances of Neural Inf. Proc. Systems*, 2017.

[29] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *See https://arxiv.org/abs/1610.02391 v3*, 2016.

[30] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[31] Guolong Su, Dennis Wei, Kush Varshney, and Dmitry Malioutov. Interpretable two-level boolean rule learning for classification. In *https://arxiv.org/abs/1606.05798*, 2016.

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

[33] Philip N. Yannella and Odia Kagan. Analysis: Article 29 working party guidelines on automated decision making under gdpr. 2018. https://www.cyberadviserblog.com/2018/01/analysis-article-29-working-party-guidelines-on-automated-decision-making-under-gdpr/.

[34] Xin Zhang, Armando Solar-Lezama, and Rishabh Singh. Interpreting neural network judgments via minimal, stable, and symbolic corrections. 2018. https://arxiv.org/abs/1802.07384.

[35] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.