Probabilistic Sufficient Explanations

Eric Wang, **Pasha Khosravi** and **Guy Van den Broeck**
University of California, Los Angeles
ericzxwang@ucla.edu, {pashak,guyvdb}@cs.ucla.edu

Abstract

Understanding the behavior of learned classifiers is an important task, and various black-box explanations, logical reasoning approaches, and model-specific methods have been proposed. In this paper, we introduce *probabilistic sufficient explanations*, which formulate explaining an instance of classification as choosing the "simplest" subset of features such that only observing those features is "sufficient" to explain the classification. That is, sufficient to give us strong probabilistic guarantees that the model will behave similarly when all features are observed under the data distribution. In addition, we leverage tractable probabilistic reasoning tools such as probabilistic circuits and expected predictions to design a scalable algorithm for finding the desired explanations while keeping the guarantees intact. Our experiments demonstrate the effectiveness of our algorithm in finding sufficient explanations, and showcase its advantages compared to Anchors and logical explanations.

1 Introduction

Machine learning models are becoming ubiquitous, and are being used in critical and sensitive areas such as medicine, loan applications, and risk assessment in courts. Hence, unexpected and faulty behaviors in machine learning models can have significant negative impact on people. As a result, there is much focus on explaining and understanding the behavior of such models. Explainable AI (or XAI) is an active area of research that aims to tackle these issues.

There have been many approaches toward explaining an instance of a classification (called a local explanation) from different perspectives, including logic-based [Shih *et al.*, 2018; Ignatiev *et al.*, 2019a; Darwiche and Hirth, 2020] or model-agnostic approaches [Ribeiro *et al.*, 2016; Lundberg and Lee, 2017; Ribeiro *et al.*, 2018]. Each of these methods have their pros and cons; some focus on scalability and flexibility, and some focus on providing guarantees.

In this work, we strive to *probabilistically* explain an instance of classification. Explanations are partial examples, where we treat the features not in the explanation as missing values. We aim for our explanations to be as simple as possible while providing the following sufficiency guarantee: given only the features in the explanation, with high probability under the data distribution $\Pr(\mathbf{X})$, the classifier makes the same prediction as on the full example. Simplicity and sufficiency are often at odds with each other, hence balancing them is a challenging act.

We briefly overview other approaches to local explanations and discuss their pros and cons in the framework of sufficiency and simplicity. Broadly, model-agnostic methods are more scalable and flexible but tend to be not as reliable as logic-based methods in providing sufficiency. On the other hand, logical explanation methods tend to sacrifice simplicity for deterministic guarantees of sufficiency. We then motivate how probabilistic notions of sufficiency can overcome these shortcomings as the foundation for sufficient explanations.

Next, we introduce the probabilistic reasoning tools needed to quantify sufficiency and define sufficient explanations: the *Same-Decision Probability* (SDP) and *Expected Prediction* (EP). We use probabilistic circuits (PCs) [Choi *et al.*, 2020] to model the probability distribution $\Pr(\mathbf{X})$ over the features due to their expressivity and tractability in answering complex probabilistic queries. We also explore connections between SDP and expected prediction and show why expected prediction is better suited for providing the probabilistic guarantees for sufficient explanations.

We then formalize the desired properties of sufficient explanations and motivate our choices for sufficiency and simplicity. Sufficiency leads to maximizing the expected prediction, while simplicity leads to constraining the size of explanations as well as choosing the subset that also maximizes the marginal probability. We capture all these with an optimization problem for finding sufficient explanations. Then, we design a scalable algorithm for finding the most likely sufficient explanation by leveraging tractability of expected prediction.

We provide experiments showcasing the empirical advantage of sufficient explanations and the effectiveness of our search algorithm. Our advantages include: (i) compared to Anchors we get better and more accurate sufficiency guarantees and (ii) our method finds simpler explanations and is more scalable than logical methods. Lastly, we show the tradeoffs between sufficiency and simplicity, and show that slightly reducing the guarantee can lead to simpler explanations with higher likelihood.

2 Background and Related Work

Notation. We use uppercase letters (X) for features (random variables) and lowercase letters (x) for their value assignments. Analogously, we denote sets of features in bold uppercase ($\mathbf{X}$) and their assignments in bold lowercase ($\mathbf{x}$). We denote the set of all possible assignments to $\mathbf{X}$ as $\mathcal{X}$. Concatenation $\mathbf{XY}$ denotes the union of disjoint sets. We focus on discrete features unless otherwise noted.

We represent a probabilistic predictor as $f : \mathcal{X} \to [0,1]$ and its decision function as $\mathcal{C} : \mathcal{X} \to \{0,1\}$, with T denoting the decision threshold. Hence, $\mathcal{C}(\mathbf{x}) = [\![f(\mathbf{x}) \geq T]\!]$. Sometimes we want to directly deal with log-odds instead of probabilities, in which case we use the log-odds predictor $\mathcal{O} : \mathcal{X} \to \mathbb{R}$ which is defined as $\mathcal{O}(\mathbf{x}) = \log \frac{f(\mathbf{x})}{1-f(\mathbf{x})}$.

2.1 Related Work

Computing explanations of classifiers has been studied from many different perspectives, including logical reasoning, black-box methods, and model-specific approaches. Some try to explain the learned model globally, making it more interpretable [Guidotti *et al.*, 2018; Liang and Van den Broeck, 2019], while others focus more locally on explaining its prediction for a single instance. Next, we go over some local explanation methods, discuss their pros and cons, and motivate how our framework might solve those issues.

Model Agnostic Approaches

These methods treat the classifier as a black box. Given an input instance to explain, they perturb the instance in many different ways and evaluate the model on those perturbed instances. Then, they use the results from the perturbations to generate an explanation. Two popular methods under this umbrella are LIME [Ribeiro *et al.*, 2016] and SHAP [Lundberg and Lee, 2017]. The main difference between these methods is the heuristics used to obtain perturbed instances and how to analyze the predictions on these local perturbations. Most provide feature attributions, which are real-valued numbers assigned to each feature, to indicate their importance to the decision and in what direction.

A benefit of these methods is that they can be used to explain any model and are generally more flexible and scalable than their alternatives. On the other hand, the downsides are that they can be very sensitive to the choice of local perturbations and might produce over-confident results [Ignatiev *et al.*, 2019b] or be fooled by adversarial methods [Slack *et al.*, 2020; Dimanov *et al.*, 2020]. One of the main reasons for these downsides is that the distribution of the local perturbations tends to be different from the data distribution the classifier was originally trained on. Hence, these approaches do not benefit from the intended generalization guarantees of machine learning models. Moreover, some of the perturbations might be low probability or even impossible inputs, and we might not care as much about their classification outcome.

Additionally, feature attribution methods treat each feature independently and cannot easily capture interactions between bigger subsets of features such as when two features cancel each other's effects. We refer to Camburu *et al.* [2020] for more such examples and discussion on pros/cons of attribution based explanations.

Logical Reasoning Approaches

These methods provide explanations with some principled guarantees by leveraging logical reasoning tools. Some approaches use knowledge compilation and tractable Boolean circuits [Shih *et al.*, 2018; Darwiche and Hirth, 2020; Shi *et al.*, 2020], some adopt the framework of abductive reasoning [Ignatiev *et al.*, 2019a; Ignatiev *et al.*, 2019b], and some tackle a specific family of models such as linear models [Marques-Silva *et al.*, 2020], decision trees [Izza *et al.*, 2020], or tree ensembles [Devos *et al.*, 2020].

The main benefit of these approaches is that they guarantee provably correct explanations, that is they guarantee a certain prediction for all examples described by the explanation. On the other hand, one downside is that they are generally not as scalable (in the number of features) as black-box methods. Another downside is that they need to completely remove the uncertainty from the classifier to be able to use logical tools and therefore become more rigid. In particular, in order to guarantee a certain outcome with absolute certainty, it is often necessary to include many of the features into the explanation, making the explanation more complex.

Sufficient Reasons [Shih *et al.*, 2018; Darwiche and Hirth, 2020] is one example of these methods that selects as an explanation a minimal subset of features guaranteeing that, no matter what is observed for the remaining features, the decision will stay the same. Sufficient reasons, as well as related logical explanations, ensure minimality and deterministic guarantees in the outcome, while as we see later our sufficient explanations ensure probabilistic guarantees instead.

For a recent and more comprehensive comparison of logic-based vs. model-agnostic explanation methods, we refer to Ignatiev *et al.*; Ignatiev [2019b; 2020].

3 Motivation and Problem Statement

We will overcome the limitations of both the model-agnostic and logic-based approaches by building local explanation methods that are aware of the distribution over features $\Pr(\mathbf{X})$. We assume for now that we have access to an accurate model for the feature distribution, and discuss in Section 4.1 how it can be obtained. This distribution will allow us to (i) reason about the classifier's behavior on realistic input instances and (ii) provide probabilistic guarantees on the veracity of the explanations. We thus take a principled probabilistic approach in explaining an instance of classification.

Intuitively, given an instance $\mathbf{x}$ and the classifiers outcome $\mathcal{C}(\mathbf{x})$, we would like to choose a subset of features $\mathbf{z} \subseteq \mathbf{x}$ as the "simplest sufficient explanation." Firstly, we want it to be sufficient, in that it provides strong probabilistic guarantees about the outcome of the classifier when only features $\mathbf{z}$ are observed. To this end, we want to maximize some sufficiency metric $\mathcal{F}(\mathbf{z}, \mathbf{x}, f, \Pr)$ describing the behavior of our predictor when only the explanation $\mathbf{z}$ is observed. However, naively maximizing this metric can run into the pitfall of building more complex explanations in order to squeeze out tiny improvements in the sufficiency metric. To address this, we have our second metric of simplicity $S(\mathbf{z}, \Pr)$, which we use to define a constraint when maximizing the sufficiency metric. Putting these together, we call a subset $\mathbf{z}$ of an instance $\mathbf{x}$ a sufficient explanation if it maximizes some sufficiency metric $\mathcal{F}$ under some simplicity constraint.

With this current definition of sufficient explanations, there is still the possibility that two subsets of a given instance, one a subset of the other, are both simple and have the same degree of sufficiency. This motivates us to further define a minimal sufficient explanation $\mathbf{z}$, i.e. no subset of $\mathbf{z}$ is also a sufficient explanation. This condition is exactly the meaning of "prime" in prime implicants as they are used in logical explanations. We will see in Section 5 how we can use the feature distribution to ensure this minimality property.

4 Probabilistic Notions of Sufficiency

We next introduce two notions of sufficiency using probabilistic reasoning tools and discuss their pros and cons. Then, we discuss their connections. Finally, we use these tools in Section 5 to formalize our definition and formulate finding the desired subsets of features as an optimization problem.

4.1 Probabilistic Reasoning Tools

Probabilistic reasoning is a hard task in general, so we need to choose our probabilistic model for $\Pr(\mathbf{X})$ carefully. We choose *probabilistic circuits* (PCs), which given some structural constraints enable tractable and exact computation of probabilistic reasoning queries such as marginals [Choi *et al.*, 2020]. Moreover, they do so without giving up much expressivity. Another advantage of PCs is that we can learn their structure and parameters from data, allowing us to avoid the exponential worst-case behavior of other probabilistic models.

The two main probabilistic reasoning tools that we use for our explanations are the *Same Decision Probability* (SDP) [Chen *et al.*, 2012] and *Expected Prediction* (EP) [Khosravi *et al.*, 2019b]. We introduce them next and explore their tradeoffs and connections.

First we have SDP [Chen *et al.*, 2012] which, intuitively, gives us the probability that our classifier has the same output as $\mathcal{C}(\mathbf{x})$ given only some subset of observed features $\mathbf{z}$.[1]

Definition 1 (Same Decision Probability)**.** *Given a classifier $\mathcal{C}$, a distribution $\Pr(\mathbf{X})$ over features, a partition $\mathbf{ZM}$ of features $\mathbf{X}$, and an assignment $\mathbf{x}$ to $\mathbf{X}$ (and corresponding $\mathbf{z} \subseteq \mathbf{x}$), the same decision probability (SDP) of $\mathbf{z}$ w.r.t. $\mathbf{x}$ is*

$$SDP_{\mathcal{C},\mathbf{x}}(\mathbf{z}) = \mathop{\mathbb{E}}_{\mathbf{m}\sim\Pr(\mathbf{M}|\mathbf{z})} [\![\mathcal{C}(\mathbf{zm}) = \mathcal{C}(\mathbf{x})]\!].$$

The higher the SDP, the better guarantee we have that the partial example $\mathbf{z}$ will be classified the same way as the full example $\mathbf{x}$. SDP and related notions have been successfully used in applications such as trimming Bayesian network classifiers [Choi *et al.*, 2017] and robust feature selection [Choi and Van den Broeck, 2018]. Renooij [2018] introduced various theoretical properties and bounds on the SDP.

Other explanation methods that provide sufficiency guarantees can be fit into our framework, using SDP as the sufficiency metric. Notably, Anchors [Ribeiro *et al.*, 2018] can be thought of as an empirical approximation of SDP sufficiency, as they aim to provide sufficiency guarantees based on sampling local perturbations instead of relying on the data distribution. Logical explanations also fit in this framework, as they completely prioritize having deterministic guarantees, i.e. SDP=1, over the need for simplicity. Another example of a method that uses probabilistic sufficiency can be found in Khosravi *et al.* [2019b], where they explain logistic regression models w.r.t. Naive Bayes data distributions.

Expected Prediction is another probabilistic reasoning task that has been shown to be successful in handling missing values in classification [Khosravi *et al.*, 2019a; Khosravi *et al.*, 2019b; Khosravi *et al.*, 2020]. It provides a promising alternative for SDP in explanations. Intuitively, given some partial observation, expected prediction can be thought of as trying all possible ways of imputing the remaining features and computing an average of all subsequent predictions weighted by the probability of each imputation. In many cases, our classifiers directly output a probability and in those cases we can compute expected prediction as follows:

Definition 2 (Expected Prediction)**.** *Given a probabilistic predictor f, a distribution $\Pr(\mathbf{X})$ over features, a partition $\mathbf{ZM}$ of features $\mathbf{X}$, and an assignment $\mathbf{z}$ to $\mathbf{Z}$, the expected prediction of f on $\mathbf{z}$ is*

$$EP_f(\mathbf{z}) = \mathop{\mathbb{E}}_{\mathbf{m}\sim\Pr(\mathbf{M}|\mathbf{z})} f(\mathbf{zm}).$$

For some tasks, we may care more about odds rather than probabilities. In those cases, the predictor usually outputs the log-odds $\mathcal{O}(\mathbf{x})$, so we also define expected log-odds:

Definition 3 (Expected Log-Odds)**.** *Given a log-odds predictor $\mathcal{O}$, a distribution $\Pr(\mathbf{X})$ over features, a partition $\mathbf{ZM}$ of features $\mathbf{X}$, and an assignment $\mathbf{z}$ to $\mathbf{Z}$, the expected log-odds of f on $\mathbf{z}$ is*

$$EP_{\mathcal{O}}(\mathbf{z}) = \mathop{\mathbb{E}}_{\mathbf{m}\sim\Pr(\mathbf{M}|\mathbf{z})} \mathcal{O}(\mathbf{zm}).$$

In this paper, unless otherwise noted, expected prediction denoted as EP($\mathbf{z}$) could refer to both cases.

4.2 Connections between SDP and EP

The choice between using SDP and expected prediction as a sufficiency metric is important, as it will decide what kind of guarantees our explanations provide and how efficient they are to compute. Here we explore connections between SDP and expected prediction and provide intuition on some advantages of using expected prediction in defining sufficient explanations.

One of the main differences between SDP and expected prediction is their computational complexity. While SDP is an appealing criteria to use for selecting explanations, computing the SDP exactly is computationally hard. In particular, it is PP^{PP}-hard on Bayesian networks [Choi *et al.*, 2012]. Even on a simple Naive Bayes model for both Pr and the classifier, computing SDP is NP-hard [Chen *et al.*, 2013].

On the other hand, expected predictions can be tractably computed for many different pairs of discriminative and generative models. For example, it is known to be tractable for the following cases: (i) logistic regression using a conformant

[1] SDP was originally defined for the classifier being a conditional probability test in distribution Pr. Here, we slightly generalize SDP to apply to a distribution Pr with a separate classifier $\mathcal{C}$.

naive Bayes distribution [Khosravi *et al.*, 2019b], (ii) decision trees w.r.t. probabilistic circuits (PCs) [Khosravi *et al.*, 2020], (iii) discriminative circuits w.r.t. PCs [Khosravi *et al.*, 2019a], and (iv) when both the feature distribution and predictor are defined by the same PC distribution $\Pr$. In the latter case, the predictor is the conditional probability $\Pr(c \mid \mathbf{X})$, and the feature distribution is $\Pr(\mathbf{X})$. Then, expected prediction can be reduced to probabilistic marginal inference in PCs which is tractable for decomposable circuits [Choi *et al.*, 2020].

Another difference comes from how SDP and expected prediction handle the two distinct uncertainties that arise from the feature distribution and the classifier. Although both are aware of the uncertainty from the feature distribution $\Pr$, SDP ignores the uncertainty in the classifier since it only deals with the decision function $\mathcal{C}(\mathbf{x})$. For example, SDP cannot distinguish between cases when the classifier has low confidence $f(\mathbf{x}) = T + \epsilon$ (only slightly above decision threshold) and cases when the classifier has high confidence such as $f(\mathbf{x}) = 0.99$. In many domains this distinction is vital. For example, doctors care more about the odds of a patient having cancer rather than binary decisions.

For these reasons, we choose to use expected predictions to define and optimize for our explanations. The good news is that optimizing EP also allows us to maximize a lower bound on SDP. The next theorem provides a theoretical lower bound for SDP using expected predictions.

Theorem 1. *Given a predictor f (or $\mathcal{O}$), its thresholded classifier $\mathcal{C}$, a positively classified instance $\mathbf{x}$ (i.e. $\mathcal{C}(\mathbf{x}) = 1$), a distribution $\Pr$, and some subset of the features $\mathbf{z} \subseteq \mathbf{x}$, we have:*

$$SDP_{\mathcal{C},\mathbf{x}}(\mathbf{z}) > \frac{EP(\mathbf{z}) - T}{U(\mathbf{z}) - T}. \qquad (1)$$

where $EP(\mathbf{z})$ refers to expected prediction, and $U(\mathbf{z})$ is an upper bound for the predictor after fixing $\mathbf{z}$, i.e. $\forall \mathbf{m}\, U(\mathbf{z}) \geq f(\mathbf{zm})$. Moreover, if $U(\mathbf{z})$ is a tight bound, then Equation 1 is also tight.

Proof. The proof is included in the appendix.[2] □

Theorem 1 gives us two ways of computing a lower bound for SDP: using a probabilistic predictor f or a log-odds predictor $\mathcal{O}$. Additionally, the theorem can be easily generalized to cases when $\mathcal{C}(\mathbf{x}) = 0$. See appendix for a more detailed discussion on which bound is better.

5 Probabilistic Sufficient Explanations

In this section, we use the aforementioned sufficiency metrics and introduce some desirable constraints to ensure simplicity. Then, we put everything together to formalize sufficient explanations and introduce an optimization problem for finding them. Finally, we introduce a search algorithm for finding sufficient explanations by modifying beam search and leveraging tractability of expected predictions.

To simplify the definitions, we assume without loss of generality that the instance we want to explain is positively classified, i.e. $\mathcal{C}(\mathbf{x}) = 1$. In Section 4.2, we saw two candidates that we can use for probabilistic sufficiency guarantees. The first was SDP, which focuses on the final decision of the model. The second was expected prediction, which takes the confidence of the model into account. Because of this, along with its computational tractability, we choose to use expected prediction for our sufficiency guarantees. More specifically, we want to maximize the expected prediction of our explanation to ensure our model is confident in its classification. In addition, maximizing the expected prediction also maximizes a lower bound for SDP, so in practice we will still get good SDP guarantees.

Having chosen a suitable sufficiency metric, we must now choose a simplicity constraint. There are multiple candidates to choose from for this as well. For example, we can impose a cardinality constraint or require explanations to have a high enough likelihood. We choose the cardinality constraint as it is easy to decide on a threshold on the explanation size. We can now formalize the notion of sufficient explanations.

Definition 4 (Sufficient Explanations)**.** *Given a predictor f, distribution $\Pr$, and a positively classified instance $\mathbf{x}$ (i.e. $\mathcal{C}(\mathbf{x}) = 1$), the set of sufficient explanations for $\mathbf{x}$ is defined as the solution of the following optimization problem:*

$$\underset{\mathbf{z} \subseteq \mathbf{x}}{\operatorname{argmax}}\, EP(\mathbf{z}) \quad \textit{s.t. } |\mathbf{z}| \leq k$$

which we denote as $SE_k(\mathbf{x})$.

Having defined sufficient explanations, and following the same line of thought as before, we are now interested in choosing sufficient explanations which are minimal. For logical explanations, any minimal sufficient subset can be chosen. However, we can be more selective in our choices using the feature distribution. A natural choice is to choose the most likely sufficient explanations, as these are the most realistic. By maximizing the marginal probability of the explanation, we also ensure the desired minimality. This is because, for subsets $\mathbf{z}_1$ and $\mathbf{z}_2$ of an instance $\mathbf{x}$, if $\mathbf{z}_1 \subseteq \mathbf{z}_2$ then $\Pr(\mathbf{z}_1) \geq \Pr(\mathbf{z}_2)$. We thus arrive at the following definition.

Definition 5 (Most Likely Sufficient Explanations)**.** *Given a predictor f, a distribution $\Pr$, and an instance $\mathbf{x}$, the most likely sufficient explanations for $\mathbf{x}$ are given by:*

$$MLSE_k(\mathbf{x}) = \underset{\mathbf{z} \in SE_k(\mathbf{x})}{\operatorname{argmax}}\ \Pr(\mathbf{z})$$

Next, we devise a search algorithm for finding sufficient explanations. In Section 6, we show that our algorithm works well in practice, ensuring both sufficiency and simplicity.

Finding Probabilistic Sufficient Explanations

To find the most likely sufficient explanations, we use a beam search algorithm to greedily search the space of potential explanations. We do so by keeping track of the top b candidates (the beam) for explanations for each cardinality based on their expected predictions and marginal probabilities.

In more detail, we begin at level zero with the empty candidate, i.e. no features selected. For each subsequent level, we expand the top b candidates of the previous level by considering all feature subsets with one more feature than in the previous level, formed by adding a previously unselected feature to each candidate. We then select the new top b candidates by ranking the expanded states based on EP, breaking

[2] Available at http://starai.cs.ucla.edu/papers/WangIJCAI21.pdf

ties using Pr, and keeping the top b. The search stops after level k. At the same time we keep track of the most likely candidate satisfying the current maximum EP.

Due to the nature of the search, each level of the beam search is highly parallelizable, which can be leveraged to speed up the search. Additionally, by running the algorithm for k levels, we can also keep track of the best sufficiency guarantee at each level between 1 and k. Hence, with a little extra book keeping we can keep track of explanations with different sizes, and thus degrees of simplicity, and their corresponding sufficiency guarantees.

Note that the search framework can also be easily adapted for different use cases. For example, we can provide a fixed sufficiency constraint and then find the simplest and most likely explanation with an expected prediction higher than the sufficiency threshold. However, choosing a good threshold for expected prediction is not always straightforward.

6 Experiments

In this section, we provide several experiments to showcase the effectiveness of our search algorithm in finding sufficient explanations.[3] Additionally, we aim to highlight the advantages of our method in comparison with Anchors and logical explanation methods. More specifically, we would like to answer the following questions:

– Can we find explanations with good sufficiency guarantees? How do they compare with Anchors?

– Does relaxation from logical to probabilistic guarantees lead to much simpler explanations?

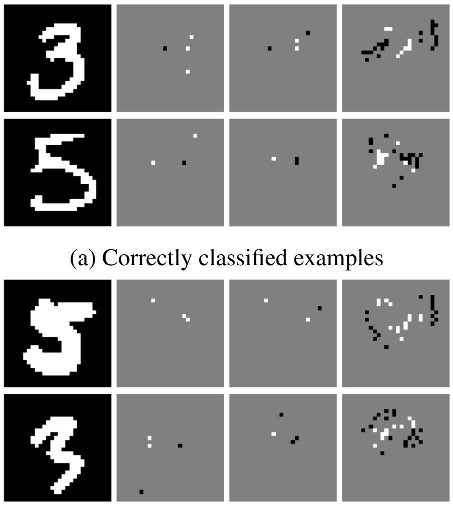
– What are the tradeoffs between different sufficiency levels and explanation complexity?

We use the adult and MNIST datasets [Kohavi, 1996; Yann *et al.*, 2009] for our experiments. For each dataset, we model the feature distribution by learning a probabilistic circuit (PC) using the open source Juice library [Dang *et al.*, 2021]. We choose decision forests learned by XGBoost [Chen and Guestrin, 2016] as our classifier, as they are popular and because expected prediction is tractable for forests w.r.t. PCs [Khosravi *et al.*, 2020]. For more detailed information on the datasets, preprocessing steps, learned models, and computing infrastructure, please refer to the appendix.

6.1 Comparison with Anchors

To demonstrate the scalability of our method and showcase some advantages of our method in comparison with Anchors, we ran our algorithm on the binarized MNIST dataset with a binary classification task of distinguishing between 3s and 5s.

Some images are shown in Figure 1 along with a comparison between explanations found using our method and Anchors. For Anchors, we used an SDP (precision) threshold of 0.95, a tolerance (δ) of 0.05, and a beam size of 5. On average it took Anchors 454s to generate explanations. Our algorithm with the same beam size and cardinality constraint $k = 30$ took 347s using 16 threads; the sufficient explanations with the same size as Anchors took 45s. See appendix for more details on the run-times.



(a) Correctly classified examples

(b) Misclassified examples

Figure 1: Explanations for selected MNIST images. From left to right: 1) original image; 2) Anchors explanation; 3) our explanation with same number of features 4) our explanation with $k = 30$. Gray pixels were not chosen for the explanation. Pixels chosen for the explanation are colored the same color as the original image.

Method	$\lvert \mathrm{EP}_{\mathcal{O}}(\mathbf{z}) \rvert$	$\mathrm{SDP}_{\mathcal{C},\mathbf{x}}(\mathbf{z})$	$\log P(\mathbf{z})$
Anchors	0.75 ± 0.37	0.66 ± 0.08	-3.29 ± 0.88
MLSE_s	1.57 ± 0.29	0.86 ± 0.05	-3.05 ± 0.65
MLSE_{10}	3.11 ± 0.23	0.99 ± 0.01	-6.98 ± 1.37
MLSE_{20}	3.60 ± 0.15	1.00 ± 0.00	-9.90 ± 2.14
MLSE_{30}	3.75 ± 0.13	1.00 ± 0.00	-11.77 ± 2.88

Table 1: Comparison of average expected log-odds, SDP, and marginals between Anchors and MLSE, averaged over 50 random MNIST test images. We take the absolute value of $\mathrm{EP}_{\mathcal{O}}(\mathbf{z})$ to measure confidence of the explanations (since it could be negative). MLSE_s sets the cardinality constraint to the same size of the Anchors explanation for each image. The ± denotes one standard deviation. The SDP values are approximated.

The last image of each row in Figure 1 is the explanation found using our algorithm with cardinality constraint $k = 30$. We see that these explanations were able to pick up on certain features we would naturally use to distinguish between 3s and 5s. In particular the chosen pixels were mostly in the upper portion of each image. This makes sense as both 3s and 5s have a similar arch shape in their lower portions, so the upper portion would be more useful for distinguishing between the two. Additionally, the explanations contain not only some white pixels showing an outline of the predicted number, but also some black pixels, where a number of the opposite label may be present. Finally, by looking only at the explanation in the rightmost column of the last two rows we can guess that the classifier will misclassify those examples.

Table 1 provides data for the expected log-odds, SDP, and marginal probabilities for generated explanations using Anchors and our method in a few different scenarios. Since SDP

[3] Code at github.com/UCLA-StarAI/SufficientExplanations

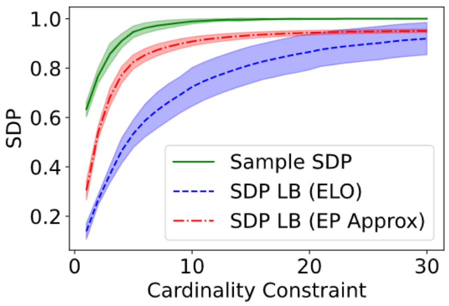


Figure 2: For sufficient explanations (sizes 1-30), we plot SDP estimates (green) vs SDP lower bounds calculated based on expected log-odds (blue), and lower bounds based on approximate expected prediction (red), averaged over 50 test images of MNIST. Shaded regions represent one standard deviation.

is intractable to compute exactly, we estimate it by computing the SDP on 10000 samples drawn from the probabilistic circuit conditioned on the explanation. One advantage of using PCs as our generative model is that drawing conditional samples from $\Pr(\cdot \mid \mathbf{z})$ is very fast. For example, generating 10K samples takes 1 second. We see that the Anchors are quite overconfident, giving explanations with much lower sample SDPs compared to the desired 0.95. This trend was also observed in Ignatiev *et al.* [2019b].

We plot the sample SDPs for our explanations, along with lower bounds calculated using Theorem 1, in Figure 2. The green line shows that, even when optimizing the EP of our explanations, the SDP still tends to be very high. Moreover, the blue line shows that the simple and efficient SDP lower bound can also provide this guarantee for some of the larger explanations with high EP. The red line is yet another way to estimate the SDP lower bound. See appendix for more details.

6.2 Comparison with Logical Explanations

We also compared explanations found using our method to logical explanations, i.e. minimal explanations with SDP $=$ 1. We used the adult dataset, which has much fewer features, for this task in order for the logical explanation computation to become feasible. We removed some features to allow for brute force computation of minimum cardinality logical explanations. We chose to use brute force because, as far as we know, there is no method for finding logical explanations for our use case. In total we removed 3 features, leaving us with 11 features. From our brute force search on some test examples, we found that logical explanations needed on average 39% of the features in order reach an SDP of one. By using our algorithm of maximizing the expected prediction, we found that in most cases selecting only 18% of the features was already enough to guarantee an SDP of 0.95 on average. More detailed numbers are provided in Table 2. We see that the strict requirement imposed by logical explanation methods can lead to selecting more features leading to more complex explanations. We expect this gap between the complexities of logical and probabilistic explanations to widen for datasets with more features or more complex models.

Method	$\mathrm{SDP}_{\mathcal{C},\mathbf{x}}(\mathbf{z})$	$\log P(\mathbf{z})$	size
Logical	1.0	-7.12 ± 2.11	4.30 ± 1.13
Anchors	0.98 ± 0.02	-4.27 ± 2.61	2.02 ± 1.26
MLSE_s	0.97 ± 0.03	-3.85 ± 2.37	2.17 ± 1.18
MLSE_1	0.88 ± 0.19	-2.23 ± 1.34	1.0 ± 0.0
MLSE_2	0.95 ± 0.08	-3.88 ± 1.88	2.0 ± 0.0
MLSE_3	0.98 ± 0.05	-4.77 ± 2.31	2.99 ± 0.06
MLSE_4	0.99 ± 0.03	-5.63 ± 2.59	3.96 ± 0.22

Table 2: SDP, marginal probability, and size statistics for explanations found using different methods for the adult dataset.

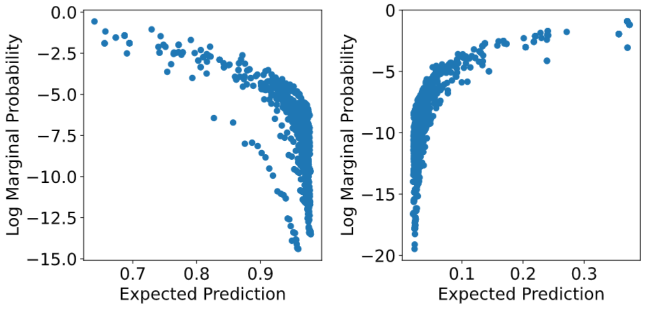


Figure 3: Tradeoff between expected prediction and marginal probability for MLSEs. The first plot is for positive label images (5s); the second is for negative label images (3s). Expected predictions were computed using a first order approximation.

6.3 Tradeoffs between Sufficiency and Simplicity

Finally, we examine the tradeoff between maximizing sufficiency and explanation simplicity. We compare the expected predictions with marginal likelihoods of explanations. Figure 3 shows a scatter plot comparing these two quantities for different explanations generated for 50 MNIST images. As we see, the general trend is that by enforcing less strict sufficiency requirements we can get more likely (and also smaller) explanations. In particular, the trend is very steep around EPs of 0 and 1, meaning that making the guarantees even slightly probabilistic will lead to significant simplification of the explanations, thus validating our probabilistic approach.

7 Conclusion

We introduced a new framework for reasoning about the local behavior of classifiers. We formulated the problem as finding simplest and most likely explanations that maximize a probabilistic guarantee, and discussed advantages of our framework compared to model agnostic and logical explanation methods. We provided experiments to validate our claims. We conclude that probabilistic sufficient explanations are a valuable addition to the arsenal of local explanation methods.

Acknowledgments

The authors would like to thank YooJung Choi for helpful discussions regarding SDP. This work is partially supported by NSF grants #IIS-1943641, #IIS-1633857, #CCF-1837129, DARPA grant #N66001-17-2-4032, a Sloan Fellowship, Intel, and Facebook.

References

[Camburu *et al.*, 2020] Oana-Maria Camburu, Eleonora Giunchiglia, Jakob Foerster, Thomas Lukasiewicz, and Phil Blunsom. The struggles of feature-based explanations: Shapley values vs. minimal sufficient subsets. *arXiv*, 2020.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, 2016.

[Chen *et al.*, 2012] Suming Chen, Arthur Choi, and Adnan Darwiche. The same-decision probability: A new tool for decision making. In *PGM*, 2012.

[Chen *et al.*, 2013] Suming Jeremiah Chen, Arthur Choi, and Adnan Darwiche. An exact algorithm for computing the same-decision probability. In *IJCAI*, 2013.

[Choi and Van den Broeck, 2018] YooJung Choi and Guy Van den Broeck. On robust trimming of bayesian network classifiers. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, July 2018.

[Choi *et al.*, 2012] Arthur Choi, Yexiang Xue, and Adnan Darwiche. Same-decision probability: A confidence measure for threshold-based decisions. *International Journal of Approximate Reasoning*, 53(9):1415–1428, 2012.

[Choi *et al.*, 2017] YooJung Choi, Adnan Darwiche, and Guy Van den Broeck. Optimal feature selection for decision robustness in bayesian networks. In *Proceedings of IJCAI*, August 2017.

[Choi *et al.*, 2020] YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. 2020.

[Dang *et al.*, 2021] Meihua Dang, Pasha Khosravi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. Juice: A julia package for logic and probabilistic circuits. In *AAAI Conference (Demo Track)*, 2021.

[Darwiche and Hirth, 2020] Adnan Darwiche and Auguste Hirth. On the reasons behind decisions. In *European Conference on Artificial Intelligence (ECAI)*, 2020.

[Devos *et al.*, 2020] Laurens Devos, Wannes Meert, and Jesse Davis. Additive tree ensembles: Reasoning about potential instances. *arXiv*, 2020.

[Dimanov *et al.*, 2020] Botty Dimanov, Umang Bhatt, Mateja Jamnik, and Adrian Weller. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. In *ECAI*, 2020.

[Guidotti *et al.*, 2018] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 2018.

[Ignatiev *et al.*, 2019a] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. Abduction-based explanations for machine learning models. *Proceedings of the AAAI Conference*, 2019.

[Ignatiev *et al.*, 2019b] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. On validating, repairing and refining heuristic ml explanations. *arXiv*, 2019.

[Ignatiev, 2020] Alexey Ignatiev. Towards Trustable Explainable AI. In *IJCAI*, 7 2020.

[Izza *et al.*, 2020] Yacine Izza, Alexey Ignatiev, and Joao Marques-Silva. On explaining decision trees. *arXiv*, 2020.

[Khosravi *et al.*, 2019a] Pasha Khosravi, YooJung Choi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. On tractable computation of expected predictions. In *Advances in Neural Information Processing Systems*, 2019.

[Khosravi *et al.*, 2019b] Pasha Khosravi, Yitao Liang, YooJung Choi, and Guy Van den Broeck. What to expect of classifiers? reasoning about logistic regression with missing features. In *Proceedings of IJCAI*, Aug 2019.

[Khosravi *et al.*, 2020] Pasha Khosravi, Antonio Vergari, YooJung Choi, Yitao Liang, and Guy Van den Broeck. Handling missing data in decision trees: A probabilistic approach. In *The Artemiss Workshop at ICML*, 2020.

[Kohavi, 1996] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *KDD*, 1996.

[Liang and Van den Broeck, 2019] Yitao Liang and Guy Van den Broeck. Learning logistic circuits. In *Proceedings of the 33rd AAAI Conference*, 2019.

[Lundberg and Lee, 2017] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *NeurIPS*, 2017.

[Marques-Silva *et al.*, 2020] Joao Marques-Silva, Thomas Gerspacher, Martin C. Cooper, Alexey Ignatiev, and Nina Narodytska. Explaining naive bayes and other linear classifiers with polynomial time and delay. In *NeurIPS*, 2020.

[Renooij, 2018] Silja Renooij. Same-decision probability: threshold robustness and application to explanation. In *Proceedings of the Ninth International Conference on Probabilistic Graphical Models (PGM)*. PMLR, 2018.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *KDD*, August 2016.

[Ribeiro *et al.*, 2018] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of AAAI*, 2018.

[Shi *et al.*, 2020] Weijia Shi, Andy Shih, Adnan Darwiche, and Arthur Choi. On Tractable Representations of Binary Neural Networks. In *Proceedings of the 17th KR*, 2020.

[Shih *et al.*, 2018] Andy Shih, Arthur Choi, and Adnan Darwiche. A symbolic approach to explaining bayesian network classifiers. In *Proceedings of IJCAI*, 2018.

[Slack *et al.*, 2020] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, 2020.

[Yann *et al.*, 2009] LeCun Yann, Cortes Corinna, and Christopher JC Burges. The MNIST database of handwritten digits, 2009.

A Appendix

A.1 Proof of Theorem 1

Proof. We provide the proof for the case of using $\mathrm{EP} = \mathrm{EP}_f$, and $U(\mathbf{z})$ providing an upper bound on the probabilistic predictor $f(\mathbf{zM})$. The proof is mostly identical in the case of having $\mathrm{EP} = \mathrm{EP}_{\mathcal{O}}$ and $U(\mathbf{z})$ being an upper bound for the log-odds predictor $\mathcal{O}(\mathbf{zM})$.

Let T be the decision threshold. Then we have $\mathrm{SDP}_{\mathcal{C},\mathbf{x}}(\mathbf{z}) = \Pr(f(\mathbf{zm}) \geq T)$ and $\mathrm{EP}_f(\mathbf{z}) = \mathbb{E}[f(\mathbf{zm})]$ where $\mathbf{m} \sim \Pr(\mathbf{M}|\mathbf{z})$. Thus,

$$\begin{aligned}
\mathrm{EP}_f(\mathbf{z}) &= \mathbb{E}[f(\mathbf{zm})] \\
&= \mathbb{E}[f(\mathbf{zm})|f(\mathbf{zm}) < T] \cdot \Pr(f(\mathbf{zm}) < T) \\
&\quad + \mathbb{E}[f(\mathbf{zm})|f(\mathbf{zm}) \geq T] \cdot \Pr(f(\mathbf{zm}) \geq T) \\
&< T(1 - \Pr(f(\mathbf{zm}) \geq T)) \\
&\quad + U(\mathbf{z}) \Pr(f(\mathbf{zm}) \geq T) \\
&= T + (U(\mathbf{z}) - T) \Pr(f(\mathbf{zm}) \geq T) \\
&= T + (U(\mathbf{z}) - T)\, \mathrm{SDP}_{\mathcal{C},\mathbf{x}}(\mathbf{z}).
\end{aligned}$$

Rearranging the terms leads to Equation 1.

Next we show how to construct a distribution for $f(\mathbf{zM})$ to demonstrate the tightness of the bound. Assume we are given $\mathrm{EP}_f(\mathbf{z}) = F$, $U(\mathbf{z}) = U$, and a decision threshold T with $F < T < U$. Then let $\epsilon > 0$ and consider the distribution given by

$$P(f(\mathbf{zM}) = k) = \begin{cases} \dfrac{F - T}{U - T} + \epsilon & \text{if } k = U \\ \dfrac{U - F}{U - T} - \epsilon & \text{if } k = A \\ 0 & \text{otherwise} \end{cases}$$

where A is some constant which makes $\mathrm{EP}_f(\mathbf{z}) = F$ and $A < T$ so that $\mathrm{SDP}_{\mathcal{C},\mathbf{x}}(\mathbf{z}) = \frac{F-T}{U-T} + \epsilon$. We now show how to solve for A.

We start by computing expected prediction (Definition 2), we have:

$$\begin{aligned}
\mathrm{EP}_f(\mathbf{z}) &= \mathop{\mathbb{E}}_{\mathbf{m} \sim \Pr(\mathbf{M}|\mathbf{z})} f(\mathbf{zm}) \\
&= U \times P(f(\mathbf{zM}) = U) + A \times P(f(\mathbf{zM}) = A) + 0 \\
&= U \left(\frac{F - T}{U - T} + \epsilon \right) + A \left(\frac{U - F}{U - T} - \epsilon \right) \\
&= F
\end{aligned}$$

and then solve for A:

$$\begin{aligned}
A &= \frac{F - U \left(\frac{F-T}{U-T} + \epsilon \right)}{\left(\frac{U-F}{U-T} - \epsilon \right)} \\
&= \frac{F(U - T) - U(F - T) - U(U - T)\epsilon}{U - F - (U - T)\epsilon} \\
&= \frac{T(U - F) - U(U - T)\epsilon}{U - F - (U - T)\epsilon} \\
&< \frac{T(U - F) - T(U - T)\epsilon}{U - F - (U - T)\epsilon} \\
&= T
\end{aligned}$$

Thus we have $\mathrm{EP}_f(\mathbf{z}) = F$ and, since $A < T$, $\mathrm{SDP}_{\mathcal{C},\mathbf{x}}(\mathbf{z}) = \frac{F-T}{U-T} + \epsilon$. □

A.2 Beam Search Algorithm

Pseudocode

Algorithm 1 Finding Most likely Sufficient Explanations

```
Input: instance $\mathbf{x}$, feature distribution $\Pr(\mathbf{z})$, expected prediction function $\mathrm{EP}(\mathbf{z})$, max features $k$, beam size $b$
MLSE ← $\emptyset$
beam ← $\{\emptyset\}$
for $i = 1 \ldots k$ do
    candidates ← $\bigcup_{c \in \text{beam}} \text{expand}(c)$
    compute $\mathrm{EP}(c)$ and $\Pr(c)$ for $c \in$ candidates
    beam ← top_$b$(candidates)
    update MLSE
end for
```

In the above pseudocode, the expand function generates a new set of candidates, each formed by adding one yet unobserved feature to the input candidate. The top_b function selects the b candidates with the highest EP, with ties broken using $\Pr$. Finally, the MLSE is updated if an explanation with a higher EP is found, or if one has the same EP as the previous MLSE but a higher $\Pr$.

Computational Complexity

When using PCs to model the feature distribution, the runtimes for tractable expected prediction (EP) and marginal ($\Pr$) algorithms do not depend on how many features are observed. $\Pr$ can be computed in linear time w.r.t. circuit size. Linearity of decision forest classification also allows for expected prediction computations in time linear w.r.t. circuit size. Thus we treat EP and $\Pr$ as oracles and measure our algorithm's runtime by the number of calls to them. In each level of the beam search, we expand the previous level's top b candidates, giving us at most nb unique states, where n is the number of features. We then call EP and $\Pr$ once for each state. Since the search stops after level k, overall we make in total $O(nkb)$ calls to the EP and $\Pr$ oracles.

A.3 More Experiment Details

Computing Infrastructure

All experiments were run on a Linux server with 40 CPU cores and 500 GB of RAM, albeit not all the memory or CPU cores where needed for generating one explanation. Our method is highly parallizable as we see adding more threads speeds up the process substantially. No GPUs were used in this paper.

Although we did not use any GPUs, it should be easy to re-implement some parts of our method to support GPUs. Some of the main ingredients of our method such as computing marginal probabilities using Probabilistic Circuits already support GPUs and enjoy huge speed ups. Furthermore, computing expected predictions and the search at each level should be amenable to GPU parallization. So, overall we should be able to speed up the algorithm by 1-2 orders of magnitude. This allows us to scale to even more complicated models with more features in the future.

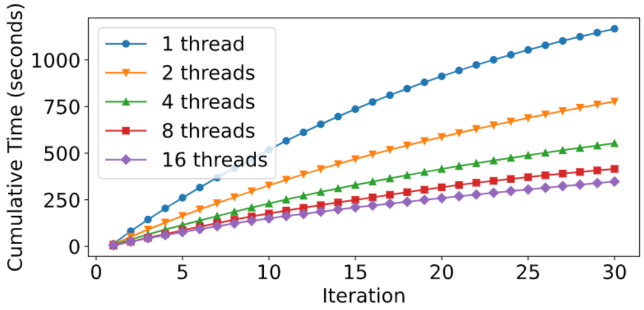


Figure 4: MNIST Runtimes. Average cumulative time taken until completion of each iteration of the beam search algorithm for different numbers of threads.

Datasets and Preprocessing Steps

The MNIST [Yann *et al.*, 2009] dataset consists of 60,000 grayscale images (28 × 28 pixels) of handwritten digits (0-9). We limit the dataset into digits of 3 and 5 to ensure we have a binary classification task. The dataset is already split between train and test images, so we used the same split, selecting from each digits of 3s and 5s. Since our probabilistic circuit library currently only supports binary feature values, we also binarized pixel values. indepedently for each image, by applying a threshold at 0.05 standard deviations above the mean.

The classification task for the adult dataset [Kohavi, 1996] is to determine whether a given individual makes over $\$50,000$ per year. Features include age, sex, working class, hours worked per week, education level, nationality, etc. We perform discretization of continuous features by applying standard binning. We then perform one-hot-encoding for all features. Again, this is because the probabilistic circuit library only supports binary features. We then used an 80-20 split for our training and test data. In general, our method should be straightforward to extend to multi-category features and also continuous features.

Source code for processing the datasets will be included in the final version of paper as supplementary.

Details on models

For our MNIST experiments, we used a PC with 10124 nodes, 15640 edges, and 5916 parameters. It took approximately 35 minutes to train, stopping after 410 iterations. For our classifier, we trained a decision forest with 6 trees, each with a maximum depth of 6. The final model has a total of 297 leaves. The classification accuracy on our test images is 98.63%.

For our adult experiments, we used a PC with 22899 nodes, 44152 edges, and 14828 parameters. It took approximately 17 minutes to train, stopping after 190 iterations of structure learning. For our classifier, we trained a decision forest with the same constraints as above. The final model has a total of 255 leaves. The classification accuracy on our test dataset is 84.20%.

Metric Calculation Details

The classifiers used in our experiments were decision forests, which allow for efficient computation of expected log-odds. Classification using these models is done by summing the weights of one leaf from each tree, where the chosen leaf from each tree has a path from the root not contradicting with the instance to be classified. This additive model allows us to utilize linearity of expectation to compute expected log-odds [Khosravi *et al.*, 2020].

This model does not, however, allow for efficient computation of expected predictions, where prediction refers to taking the sigmoid of the classifier output. This is due to the non-linear nature of the sigmoid function. Instead, when we give expected prediction values, we use a first order approximation by taking the sigmoid of the expected log-odds.

In Figure 2 we show a plot with SDP lower bounds computed using Theorem 1. Graphing of the blue curve requires computation of an upper bound $U(\mathbf{z})$ on the predictions of a decision forest. The upper bound used is a loose one which we computed by adding the weights of one leaf from each tree in the forest, where each leaf is the maximum weighted leaf in its tree whose path from the root does not contradict with the explanation. This bound is a loose once since paths for leaves from different trees may contradict. For the red curve, since our model outputs log-odds, we used the first order approximation mentioned above.

Effects of Parallelism on Performance

One advantage of our method is that we can parallelize computation of expected predictions at each level of the beam search. The effects of can be seen in Figure 4, which plots the cumulative time taken to complete each level of the beam search for different numbers of threads. We see that more threads do indeed allow for much faster runtimes. We leave investigating benefits of GPU acceleration for future work.

More on SDP Lowerbounds

In the main text we saw that Theorem 1 gives us two ways of computing a lower bound for SDP: using a probabilistic predictor f or a log-odds predictor $\mathcal{O}$. One of them might give a tigher bound, depending on how our predictor is defined. Generally, the tighter bound we can find for $U(\mathbf{z})$ the tighter bound we get for SDP.

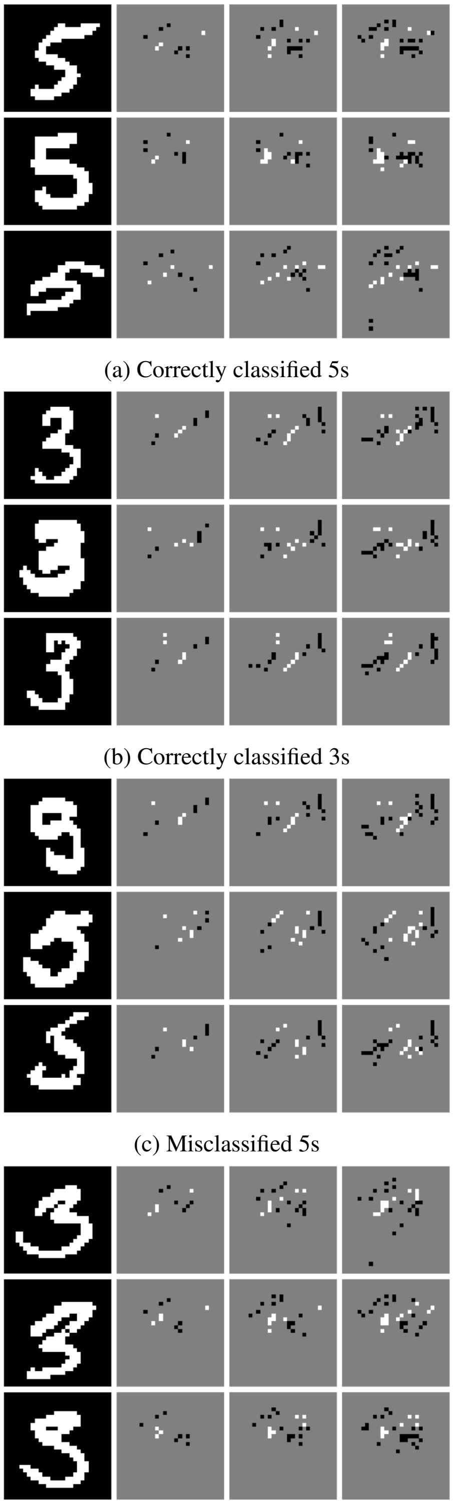
For example, in the case of having a probabilistic predictor, the decision threshold is usually $T = 0.5$, and we can easily get a trivial upper bound of $U(\mathbf{z}) = 1$. Depending on the model family we might be able to get a tighter bound for $U(\mathbf{z})$, and hence a tighter lower bound for SDP. For the case of a log-odds predictor, we usually take $T = 0$, but getting an upper-bound $U(\mathbf{z})$ might not be trivial. However, in many cases an upper bound $U(\mathbf{z})$ can be computed efficiently for log-odds predictor, but might not be tight.

Limitations

Some knowledge of probabilities is needed to fully interpret the results. However, as seen with the MNIST example, visual representations can be designed to give users a feel for what the model is doing.

More MNIST examples

Figure 5 provides some extra examples on sufficient explanations for MNIST. We see examples for both correctly and incorrectly classified images, and see the effects of cardinality constraints on the explanations.



(a) Correctly classified 5s

(b) Correctly classified 3s

(c) Misclassified 5s

(d) Misclassified 3s

Figure 5: More MNIST examples along with explanations with cardinality constraint $k = 10, 20, 30$.