Explanation Framework for Intrusion Detection

Nadia Burkart[1], Maximilian Franz[1], and Marco F. Huber[2,3]

[1] Fraunhofer IOSB, Karlsruhe, Germany
[2] Institute of Industrial Manufacturing and Management IFF, University of Stuttgart, Germany
[3] Center for Cyber Cognitive Intelligence (CCI), Fraunhofer IPA, Stuttgart, Germany

Abstract. Machine learning and deep learning are widely used in various applications to assist or even replace human reasoning. For instance, a machine learning based *intrusion detection system (IDS)* monitors a network for malicious activity or specific policy violations. We propose that IDSs should attach a sufficiently understandable report to each alert to allow the operator to review them more efficiently. This work aims at complementing an IDS by means of a framework to create explanations. The explanations support the human operator in understanding alerts and reveal potential false positives. The focus lies on counterfactual instances and explanations based on locally faithful decision-boundaries.

Keywords: Intrusion Detection · Explainable Machine Learning · Counterfactual Explanations

1 Introduction

Advances in machine learning models are associated with an increased complexity of the models. These models appear to end users and even to their developers as black boxes. The reasoning behind the model is often opaque. The research field of explainable machine learning focuses on making models more accessible, transparent and comprehensible for users. Over the past years, there was a surge in approaches for better explainability of the models. Explainable approaches are especially sought after in critical use cases like network-security, medicine or finance. By enabling a lay system user to understand and reproduce the fundamental workings of a machine learning model, trust can be built and improved. In an IDS, explanations of the underlying model can help a system operator to easily understand the model's judgment and reveal potential false positives. In a binary classification task (e.g., classifying suspicious vs. normal behaviour), the concept of a counterfactual explanation is particularly helpful for the human operator as it formalizes a common thought process: *"Why did X happen and not Y?"*. Counterfactual questions are a tool to expose flaws in the underlying decision process. By revealing counterfactuals to the system operator, this could clarify his mental model of a black box classifier and uncover flaws in the model's judgment. We focus on three aspects:

– *Understandability*: Explaining the classification of an instance, based on some form of feature importance.

- *Actionability*: Giving practical advice how to change the classification towards the desired result.
- *Simulatability*: Outlining the decision process to allow a user to simulate the behaviour of the model.

In the following, we first give some background of existing work and introduce notations. In Section 3, we then generalize existing counterfactual approaches into the five phases we consider essential for every counterfactual explanation. We slightly adapt modules of existing work, which we evaluate on the IDS scenario.

2 Explanations for Intrusion Detection

We denote by $f : \mathcal{X} \rightarrow [0, 1]$ a binary black-box classifier that we want to explain. We assume that f is pre-trained as part of an IDS. Hence, f maps so-called attack-vectors $\vec{x}$ from a multidimensional feature-space $\mathcal{X} \subseteq \mathbb{R}^n$ onto the probability that they are malicious instances.

2.1 Surrogate Models

Surrogate models approximate black-box models either locally or globally in an interpretable fashion. One of the best known methods to locally explain black box models by training a surrogate is local model-agnostic explanations (LIME) [1]. Since their work has been thoroughly explained, tested and used [2], we will not elaborate on the specifics of the method. It suffices to note that the idea of LIME is to train a surrogate model g that approximates the original black box classifier f, $g \sim f$, based on training data sampled in a neighborhood around the instance of interest, $\vec{x}_0$. LIME provides a set of feature attributions (see Section 4) derived from the weights of the linear classifier g trained on the sampled data set. These attributions tell the user, which features contributed most significantly to the result.

2.2 Counterfactual Explanation

Laugel et al. [3] note, there is another approach to the local explanation problem, which yields a slightly different interpretation. Namely, what we propose to call *decision boundary centered explanations*. While LIME illustrates which features contribute to an instance, *Local Adversarial Detection (LAD)* [4] and *Local Surrogate* [3] yield a feature attribution that is relevant at a local decision boundary. To do so, it is required to find the decision boundary first and then to train a surrogate on instances located around the decision boundary. Laugel et al. find the decision boundary through random spherical sampling around the instance $\vec{x}_0$. Wachter et al. [5] introduced another solution based on counterfactuals. A counterfactual of $\vec{x}$ is an instance $\vec{x}'$, that yields the opposite classification. Thus, given $\vec{x}$ and f we are searching for $\vec{x}'$ such that $\hat{f}(\vec{x}) \neq \hat{f}(\vec{x}')$, where $\hat{f} : \mathcal{X} \rightarrow \{0, 1\}, \hat{f}(\vec{x}) \mapsto \lfloor f(\vec{x}) \rceil$, is the binary classifier that yields the predicted class. Ideally, $\vec{x}'$ is *close* to $\vec{x}$ in the feature space $\mathcal{X}$, with respect to some distance metric $d(\cdot,\cdot)$. This formalizes the intuition that the counterfactual should be similar to the original instance. The major contribution from Wachter et al. is to consider the search for a counterfactual as an optimization problem. Formally, Watcher et al. propose to minimize a function

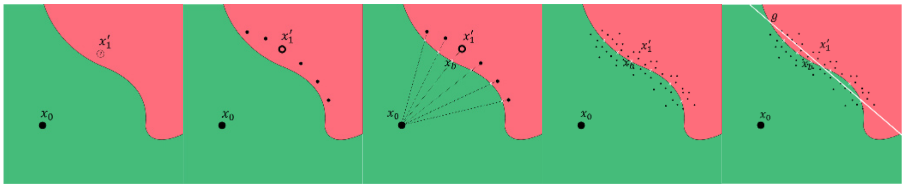
$$L(\vec{x},\vec{x}',y',\lambda) = \lambda\cdot(f(\vec{x}')-y')^2 + [(1-\lambda)\cdot d(\vec{x},\vec{x}')]\times \mathbb{I}(\vec{x}')\ , \tag{1}$$

$$\text{where } \mathbb{I}(\vec{x}') = \begin{cases} 0, & \hat{f}(\vec{x}') \neq y' \\ 1, & \hat{f}(\vec{x}') = y' \end{cases}$$

with respect to $\vec{x}'$. With $\lambda \in [0,1]$, we control the effect of locality, $y' = 1-y$ denotes the opposite class of the classifier and $\mathbb{I}$ is an optional indicator function. Since the classifier f is a black box, one has to optimize for $\vec{x}'$ using *derivative free* methods (e.g., Nelder-Mead). We elaborate on the methods in Section 3.1. In the following, we are concerned especially with the *decision boundary centered explanations* as they tend to yield more decisive results. We will see that counterfactuals are in fact a by-product of the search for the decision boundary.

3 The Modular Phases of Explanations

We dissect the method of finding *decision boundary centered explanations* into five distinct phases, containing the search for counterfactuals. Also, we present existing approaches for the single phases to give a better intuition (see Figure 1 and Table 1). We start with a given instance $\vec{x}_0$ of class A, an attack instance, of which we want to explain the classification $\hat{f}(\vec{x}_0)$. The goal is to explain why f decided $\vec{x}_0$ to be class A rather than B, a benign instance. This is the specific setting of an IDS described above. The semantic goal of the explanation is to allow the user to judge whether the decision was correct. A consideration that we wanted to keep in mind during all phases is that inference of the model f, or $\hat{f}$ for that matter, might be very expensive. Thus, we aim to keep the number of queries to the black-box small.



(a) Phase 1 (b) Phase 2 (c) Phase 3 (d) Phase 4 (e) Phase 5

Fig. 1: The five phases of explanations.

Table 1: Overview of the various approaches for the phases 1 to 5, see Section 3.1-3.5

Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
Derivate-Free [5], Growing Spheres [4], Random Sampling [3]	Magnetic Sampling, Random Sampling	Linear Search, Binary Search	Train on sample set, Train on boundary touchpoints	Explanation with interpretable model (e.g., small decision tree)

3.1 Phase 1: Finding the First Counterfactual

The first support point $\vec{x}'_1$, i.e., the first counterfactual, is an instance such that $\hat{f}(\vec{x}_0) \neq \hat{f}(\vec{x}'_1)$. As mentioned in Section 2.2, this can be formulated as an optimization problem. Alternatively, we can use random approaches similar to [1] or [4]. Randomly sampling instances in a neighbourhood of the instance $\vec{x}_0$ can be very expensive as the counterfactual might be far away in the feature space of possible instances. Therefore, we use the optimization approach introduced in [5] with minor adaptions. Particularly, we use the distance metric

$$d_m(\vec{x}, \vec{x}') = \frac{1}{n} \sum_i^n \frac{|x_i - x'_i|}{\text{MAD}_i} ,$$

that is robust to outliers. Here, n is the dimension of $\mathcal{X}$, x_i denotes the i-th feature value of instance $\vec{x}$ and MAD_i is the median absolute deviation of feature i in the training dataset P according to

$$\text{MAD}_i = \text{median}_{\vec{x} \in P} \left(x_i - \bar{x}_i \right) ,$$

with $\bar{x}_i = \text{median}_{\vec{x} \in P}(x_i)$. We normalize over the number of dimensions as our framework aims to be agnostic. Next, (1) retrieves the counterfactuals through

$$\phi(\vec{x}, y', \lambda) = \underset{\vec{x}'}{\text{argmin}} \quad \lambda \cdot (f(\vec{x}') - y')^2 + d_m(\vec{x}, \vec{x}') . \tag{2}$$

In our implementation, we minimize (2) with the Nelder-Mead simplex algorithm [6], which is a derivative free method. The result of (2) is the first counterfactual.

3.2 Phase 2: Finding Support Points

Given the first counterfactual $\vec{x}'_1 \in \mathcal{X}$, we want to find a set S of instances, such that all $\vec{x}'_i \in S$ are counterfactuals. Literally speaking, they are located on the "opposite side" of the decision boundary. The desired goal is to expand S in order to get a good representation of the actual area where f classifies instances as class B. The idea behind our approach named *MagneticSampling* is to expand the area stepwise into all directions across the dimensions starting from the initial sample $\vec{x}'_1$ until the newly sampled instances are no longer classified as B. For this purpose, we first determine the direction vector $\vec{d} := \vec{x}'_1 - \vec{x}_0$ between the original instance and the first counterfactual. We deterministically sample support points $\vec{x}'_i$, $i > 1$, by rotating $\vec{d}$ around $\vec{x}_0$, i.e., taking points with distance $\|\vec{d}\|$ from $\vec{x}_0$ that are in the vicinity of $\vec{x}'_1$, with a fixed discretization step size. This corresponds to taking the support points from the set

$$B(\vec{x}_0, \vec{x}'_1, a) = \{\vec{z} \in \mathcal{X} : \|\vec{z} - \vec{x}_0\| = \|\vec{x}'_1 - \vec{x}_0\| \text{ and } \|\vec{z} - \vec{x}'_1\| \leq a \text{ and } f(\vec{z}) = y'\} ,$$

with $\|.\|$ being the Euclidean norm.

Considering only instances around $\vec{x}'_1$ ensures that we find one connected decision boundary and not multiple patches. While possibly neglecting other possible boundaries, this simplifies the explanation [7].

3.3 Phase 3: Finding Decision Boundary

Given the set S of support points or counterfactual points, we approximately locate the decision boundary, which is somewhere on the line segment between $\vec{x}_0$ and any $\vec{x}'_i \in S$. We denote the segment by $L_i(v) := \vec{x}_0 + v \cdot (\vec{x}'_i - \vec{x}_0)$ with $v \in [0, 1]$. The result of this phase is some abstract representation of the possibly sophisticated decision boundary in local proximity to $\vec{x}_0$. To give an intuition, this could mean a set of points B such that each $\vec{x}_b \in B$ is on the decision boundary (a border touchpoint) [3], or it could be a polygon enclosing the decision boundary in a given segment. Considering the way we sampled our support points, we can assume that the value of $\hat{f}$ develops monotonously on the segment L_i. Note that this does not have to be true for the prediction probability f. Given this assumption we can use binary search on the segments to approximately locate $\vec{x}_b = L_i(v)$ for some v and thereby reduce the number of queries to our black-box f from $\mathcal{O}(n)$ to $\mathcal{O}(\log(n))$.

3.4 Phase 4: Train Explainer on Sample Set

Using the representation of the local decision boundary from Phase 3 we sample a set T of instances around the decision boundary. Given T we train a simple model g, called surrogate, to approximate the decision boundary locally. Similar to [1], we constrain the complexity $\Omega(g)$ of the model by imposing constraints like maximum depth for decision trees or number of non-zero weights for linear classifiers. Formally, we obtain g out of a class of models G (e.g. decision trees, linear models, ...) through

$$\varphi(T, f, L') = \operatorname*{argmin}_{g \in G} \sum_{\vec{x} \in T} L'\big(f(\vec{x}), g(\vec{x})\big) ,$$

where L' is some loss function (e.g. Mean-Squared-Error loss).

The framework allows manually or automatically limiting the number of features considered by the surrogate g. If no previous knowledge is available to select features, Least-Angular Regression (LARS, [8]) can be used to determine a restricted feature set.

3.5 Phase 5: Present Explanation & Give *Advice*

Given the results of the previous phases we can now present various explanations. The three major examples are

- *Feature Importance*: As Ribeiro et al. [1] verified, feature importance or attribution, can be a useful way to understand a decision post-hoc.
- *Relative Differences*: We use counterfactual instances revealed in phase one to provide actionable explanations for a user in form of relative differences. See Sec. 4 for an example.
- *Surrogate Visualization*: For the aspect of *simulability*, it is desirable to show a representation of the model to the user. Due to their computational simplicity, decision trees are favorable for this task.

4 Experiment

In this chapter the fidelity of the surrogates and their configuration is evaluated on different data sets . Furthermore, we exemplary present possible explanations for the use case of an IDS. For the IDS2017 [9] and the KDD[10] we trained the MLP classifier on the subset of Web and DOS attacks. The fidelity quantifies how well the surrogate model mimics the behavior of the MLP. Fidelity is the percentage of test examples on which the prediction made by the surrogate matches with the prediction of the trained black box (MLP) [11].

The results for the different configuration by using 10-fold cross-validation are displayed in Table 2 and Table 3. Looking at the results from Table 2 for the IDS data set, we observe that the tree surrogate proposed by the framework consistently outperforms linear approximations trained in LIME fashion and according to our linear approach explained in Section 3. As shown in [12], decision trees also far better in terms of human interpretability. In short, the decision tree trained on the decision boundary (DB-tree) is both more accurate and more interpretable. For the random configuration illustrated in Table 3 mostly LIME outperforms DB-Linear and DB-tree. The results of Table 2 and 3 illustrate that the systematic approach (Nelder Mead/Magnetic Sampling) is more effective than LIME and the random approaches.

We continue with a visualization of the possible explanations of our framework, but limit ourselves to the rather novel approaches of *relative difference*

Table 2: Fidelity for Nelder Mead/Magnetic Sampling

Data set	LIME	DB-Linear	DB-Tree
IDS [9]	0.85	0.87	**0.97**
KDD [10]	0.93	0.96	**0.99**
Heloc [13]	0.86	0.96	**0.97**
Credit [14]	0.95	**0.99**	**0.99**

Table 3: Fidelity for Random

Data set	LIME	DB-Linear	DB-Tree
IDS [9]	0.84	**0.91**	0.85
KDD [10]	**0.92**	0.84	0.83
Heloc [13]	**0.92**	0.78	0.82
Credit [14]	**0.96**	0.87	0.92

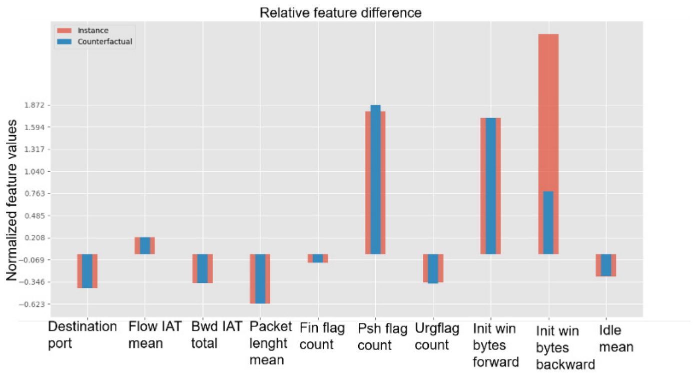


Fig. 2: Relative feature difference between instance and counterfactual (Data set: IDS)

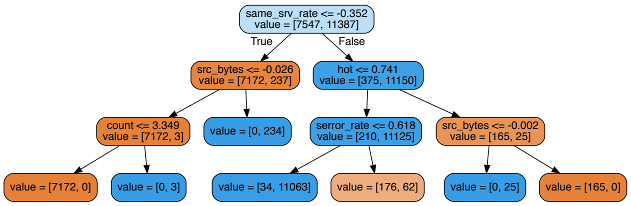


Fig. 3: Decision Tree trained on the decision boundary (Accuracy 0.99) (Data set: KDD)

and *surrogate visualization* for brevity. The feature attribution we can retrieve, matches in its nature that of LIME and can help a user to *understand* a decision. The *Relative Difference* method on the other hand, makes use of the counterfactual to give actionable advice. Figure 2 shows the differences between the instance and its counterfactual for the ten most significant features. It quickly reveals that the high value of *Init win bytes backward* caused the erroneous classification as an attack.

Surrogate visualization on the other hand helps the user to *simulate* the decision process. For this task, the decision tree depiected in Fig. 3 is suited best, as the effort for manually inferring a prediction is low [12].

5 Summary

In this paper, a theoretical framework for modular decision boundary focused explanations was proposed. By distributing the training of an explainable surrogate in different modules, flexibility and variety is introduced. The aspect of generating decision boundary centered explanations allows easily generating counterfactuals. Due to the increasing demand for explainable machine learning systems, various approaches should be pursued in parallel. With this work we contribute to the field of model-agnostic analysis, for which many methods have been proposed before [15]. Depending on the requirements of the application, other approaches like those proposed by Pearl et al. [16] ought to be pursued in parallel. By reviewing the literature on explainable machine learning, we have encountered a confusing ambiguity when it comes to terminology. Clear research directions and notation ought to be introduced. More user studies like [12] are needed to gain more insights of how understanding and actionability really can be obtained.

References

1. M. Tulio Ribeiro, S. Singh, and C. Guestrin, "“Why Should I Trust You?”: Explaining the Predictions of Any Classifier," *ArXiv e-prints*, Feb. 2016.
2. S. Mishra, B. L. Sturm, and S. Dixon, "Local interpretable model-agnostic explanations for music content analysis.," in *ISMIR*, pp. 537–543, 2017.
3. T. Laugel, X. Renard, M.-J. Lesot, C. Marsala, and M. Detyniecki, "Defining Locality for Surrogates in Post-hoc Interpretablity," *ArXiv e-prints*, June 2018.
4. X. Renard, T. Laugel, M.-J. Lesot, C. Marsala, and M. Detyniecki, "Detecting Potential Local Adversarial Examples for Human-Interpretable Defense," *ArXiv e-prints*, Sept. 2018.
5. S. Wachter, B. D. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *CoRR*, vol. abs/1711.00399, 2017.
6. J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.
7. C. Molnar, "Interpretable machine learning," *A Guide for Making Black Box Models Explainable*, 2018.
8. B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.
9. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSP*, pp. 108–116, 2018.
10. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, IEEE, 2009.
11. M. Craven and J. W. Shavlik, "Extracting tree-structured representations of trained networks," in *Advances in neural information processing systems*, pp. 24–30, 1996.
12. S. A. Friedler, C. D. Roy, C. Scheidegger, and D. Slack, "Assessing the local interpretability of machine learning models," *arXiv preprint arXiv:1902.03501*, 2019.
13. "Heloc explainable ml challenge." https://community.fico.com/s/explainable-machine-learning-challenge. Accessed: 2019-03-01.
14. H. Hofmann, "Statlog data set." https://archive.ics.uci.edu/ml/datasets/statlog. Accessed: 2019-06-13.
15. A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," *arXiv:1704.02685*, 2017.
16. J. Pearl *et al.*, "Causal inference in statistics: An overview," *Statistics surveys*, vol. 3, pp. 96–146, 2009.