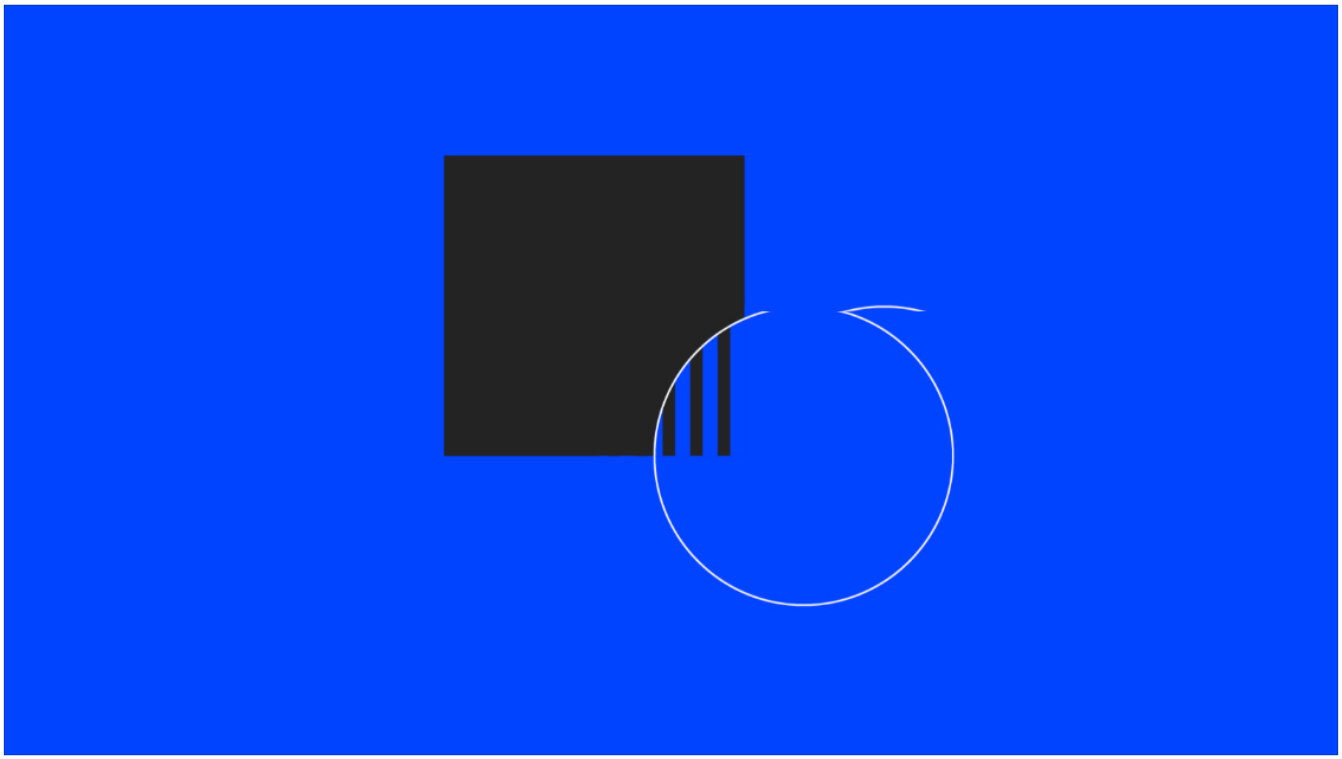
Tutorials | Responsible AI

Explainability I: local post-hoc explanations

Aug. 02, 2022 S. Prince



Two considerations

What makes a good explanation?

Taxonomy of XAI approaches

Local post-hoc explanations

Individual conditional expectation (ICE)

Counterfactual explanations

LIME

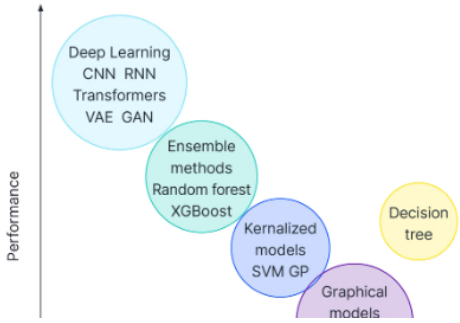
Anchors

Shapley additive explanations

Computing SHAP values

Machine learning systems are increasingly deployed to make decisions that have significant real-world impact. For example, they are used for credit scoring, to produce insurance quotes, and in various healthcare applications. Consequently, it's vital that these systems are trustworthy. One aspect of trustworthiness is explainability. Ideally, a non-specialist should be able to understand the model itself, or at the very least why the model made a particular decision.

Unfortunately, as machine learning models have become more powerful, they have also become larger and more inscrutable (figure 1). At the time of writing, the largest deep learning models have trillions of parameters. Although their performance is remarkable, it's clearly not possible to understand how they work by just examining the parameters. This trend has led to the field of *explainable AI* or *XAI* for short.



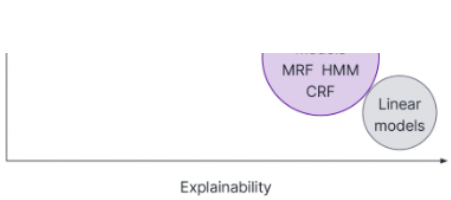


Figure 1. Explainability vs. performance. There has been a trend for models to get less explainable as they get more powerful. This is partly just because of the increasing number of model parameters (potentially trillions for large models in the top-left corner). The decision tree is a rare example of a non-linear model which is both easy to understand, and provides reasonable performance. It is unknown whether it is possible to build models in the desirable top-right corner of this chart. Adapted from AAAI XAI Tutorial 2020.

Explainable AI methods are useful for three different groups of stakeholders. For machine learning scientists, they are useful for debugging the model itself and for developing new insights as to what information can be exploited to improve performance. For business owners, they help manage business risk related to AI systems. They can provide insight into whether the decisions can be trusted and how to answer customer complaints. Finally, for the customers themselves, XAI methods can reassure them that a decision was rational and fair. Indeed, in some jurisdictions, a customer has the right to demand an explanation (e.g., the under the GDPR regulations in Europe).

Two considerations

There are two related and interesting philosophical points. Firstly, it is well known that humans make systematically biased decisions, use heuristics, and cannot explain their reasoning processes reliably. Hence, we might ask whether it is fair to demand that our machine learning systems are explicable. This an interesting question, because at the current time, it is not even known whether it is possible to find models that are explicable to humans and still have good performance. However, regardless of the flaws in human decision making, explainable AI is a worthy goal even if we do not currently know to what extent it is possible.

Secondly, Rudin 2019 has argued that the whole notion of model explainability is flawed, and that explanations of a model must be wrong. If they were completely faithful to the original model, then we would not need the original model, just the explanation. This is true, but even if it is not possible to explain how the whole model works, this does not mean that we cannot get insight into how a particular decision is made. There may be an extremely large number of local explanations pertaining to different inputs, each of which can be understood individually, even if we cannot collectively understand the whole model. However, as we shall see, there is also a strand of XAI that attempts to build novel *transparent* models which have high performance, but that are inherently easy to understand.

What makes a good explanation?

In this section, we consider the properties that a good explanation should have (or alternatively that a transparent model should have). First, it should be easily understandable to a non-technical user. Models or explanations based on linear models have this property; it's clear that the output increases when certain inputs increase and decreases when other inputs increase, and the magnitude of these changes differs according to the regression coefficient. Decision trees are also easy to understand as they can be described as a series of rules.

Second, the explanation should be succinct. Models or explanations based on small decision trees are reasonably comprehensible, but become less so as the size of the tree grows. Third, the explanation should be accurate and have high-fidelity. In other words, it should predict unseen data correctly, and in the same way as the original model. Finally, an explanation should be complete; it should be applicable in all situations.

Adherence to these criteria is extremely important. Indeed, Gilpin *et al.* (2019) argue that it is fundamentally unethical to present a simplified description of a complex system to increase trust, if the limits of that approximation are not apparent.

Taxonomy of XAI approaches

There are a wide range of XAI approaches that are designed for different scenarios (figure 2). The most common case is that we have already trained a complex model like a deep neural network or random forest and do not necessarily even have access to its internal structure. In this context, we refer to this as a *black box* model, and we seek insight into how it makes decisions. Explanations of existing models are referred to as *post-hoc* explanations.

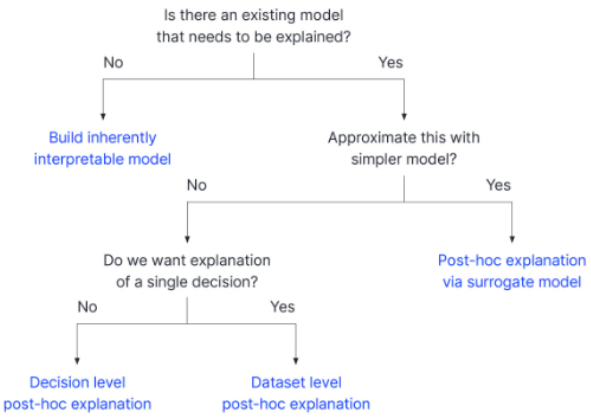


Figure 2. Taxonomy of XAI methods. If we do not already have a model that we need to explain, we can develop a model that is inherently interpretable. If we already have a model, then we must use a post-hoc method. One approach is to distill this into a simpler and more interpretable model. However, if we only use this for explanations, then the explanations are unreliable to the extent that the results differ. If we replace the original model entirely, then we may sacrifice performance. If we decide to work with just the existing model, then there are two main families of methods. Local models explain a single decision at a time, whereas global models attempt to explain the entire model behaviour. See also Singh (2019).

There are three main types of post-hoc explanation. First, we can distill the black box model into a surrogate machine learning model that is intrinsically interpretable. We could then either substitute this model (probably sacrificing some performance) or use it to interpret the original model (which will differ, making the explanations potentially unreliable). Second, we can try to summarize, interrogate, or explain the full model in other ways (i.e., a global interpretation). Third, we can attempt to explain a single prediction (a local interpretation).

If we do not already have an existing model, then we have the option of training an intrinsically interpretable model from scratch. This might be standard ML model that is easy to understand like a linear model or tree. However, this may have the disadvantage of sacrificing the performance of more modern complex techniques. Consequently, recent work has investigated training models with high performance but which are still inherently interpretable.

In part I of this blog, we consider local post-hoc methods for analyzing black box models. In part II, we consider methods approximate the entire model with a surrogate, and models that provide global explanations at the level of the dataset. We also consider families of model that are designed to be inherently interpretable.

Local post-hoc explanations

Local post-hoc explanations sidestep the problem of trying to convey the entire in an interpretable way by focusing on just explaining a particular decision. The original model consists of a very complex function mapping inputs to outputs. Many local post-hoc models attempt to just describe the part of the function that is close to the point under consideration rather than the entire function.

In this section we'll describe the most common local post-hoc methods for a simple model (figure 3) with two inputs and one output. Obviously, we don't really need to 'explain' this model, since the whole function relating inputs to outputs can easily be visualized. However, it will suffice to illustrate methods that can explain much more complex models.

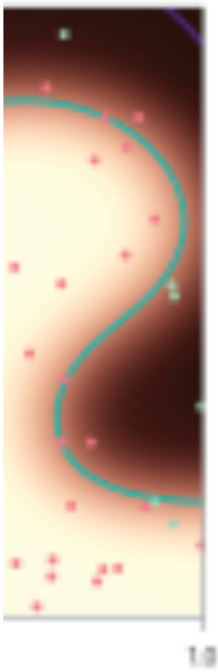


Figure 3. Model used to describe XAI techniques. The model has two inputs x_1 and x_2 and returns a probability $Pr(y = 1)$ that indicates the likelihood that the input belongs to class 1 (brighter means higher probability). The red points are positive training points and the green points are negative training points. The green line represents the decision boundary. Obviously, we do not need to "explain" this model as we can visualize it easily. Nonetheless, it can be used to elucidate local post-hoc XAI methods.

Individual conditional expectation (ICE)

An *individual conditional expectation* or *ICE* plot (Goldstein *et al.* 2015) takes an individual prediction and shows how it would change as we vary a single feature. Essentially, it answers the question: what if feature x_j had taken another value? In terms of the input output function, it takes a slice through a single dimension for a given data point (figure 4).

at a time and they do not take into account the relationships between features. It's also possible that some combinations of input features never occur in real-life, yet ICE plots display these and do not make this clear.

Counterfactual explanations

ICE plots create insight into the behaviour of the model by visualizing the effect of manipulating one of the model inputs by an arbitrary amount. In contrast, counterfactual explanations manipulate multiple features, but only consider the behaviour within the vicinity of the particular input that we wish to explain.

Counterfactual explanations are usually used in the context of classification. From the point of view of the end user, they pose the question "what changes would I have to make for the model decision to be different?". An oft-cited scenario is that of someone whose loan application has been declined by a machine learning model whose inputs include income, debt levels, credit history, savings and number of credit cards. A counterfactual explanation might indicate that the loan decision would have been different if the applicant had three fewer credit cards and an extra $5000 annual income.

From an algorithmic point of view, counterfactual explanations are input data points $\mathbf{x}^*$ that trade off (i) the distance $dist1[f[\mathbf{x}^*], y^*]$ a between the actual function output $f[\mathbf{x}^*]$ and the desired output y^* and (ii) the proximity $dist2\left[\mathbf{x}, \mathbf{x}^*\right]$ to the original point $\mathbf{x}$ (figure 5). To find these points we define a cost function of the form:

$$\hat{\mathbf{x}}^*, \hat{\lambda} = \underset{\lambda}{\operatorname{argmax}} \left[\underset{\mathbf{x}^*}{\operatorname{argmin}} \left[dist1\left[f[\mathbf{x}*], y^*\right] + \lambda \cdot dist2\left[\mathbf{x}, \mathbf{x}^*\right]\right]\right] \tag{1}$$

where the positive constant λ controls the relative contribution of the two terms. We want λ to be as large as possible so that the counterfactual examples is as close as possible to the original example.

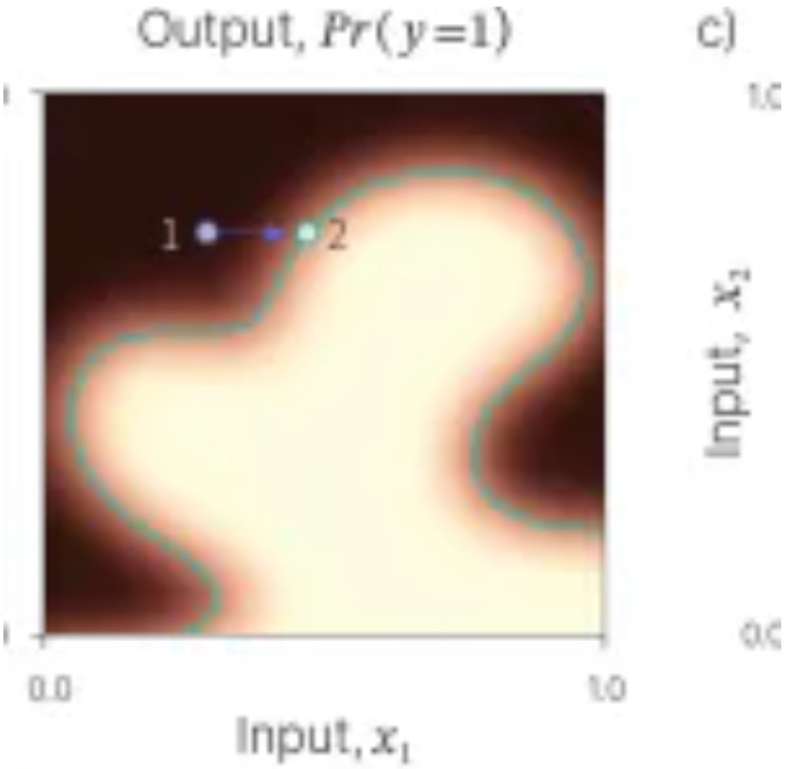


Figure 5. Counterfactual explanations. a) We want to explain a data point (purple point 1) which was classified negatively. One way to do this is to ask, how we would have to change the input so that it is classified positively. In practice, this means finding and returning the closest point on the decision boundary (cyan point 2). In a real-life situation, a customer might be able to take remedial action to move the inputs to this position. b) This remedial action may be impractical if there are many input features, and so usually we seek sparse counterfactual examples where we have only changed a few features (here just feature x_1). c) One problem with counterfactual examples is that there be many potential ways to modify the input (brown point 1) to change the classification (points 2, 3 and 4). It's not clear which one should be presented to the end user.

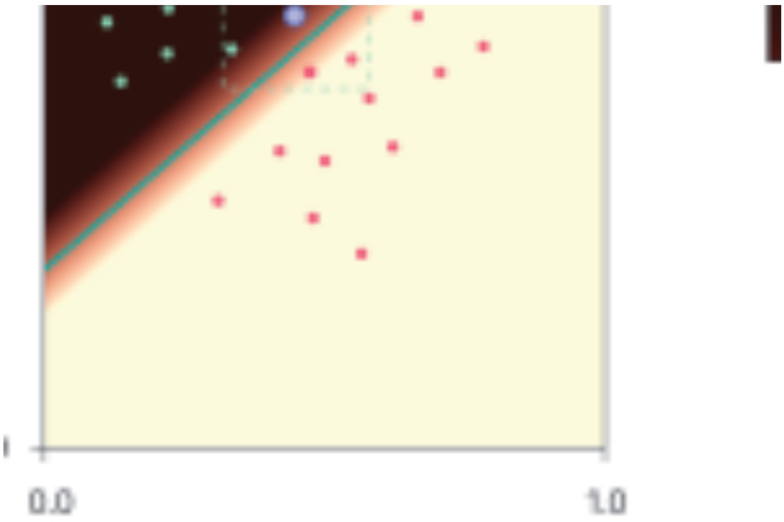
This formulation was introduced by Wachter *et al.* (2017) who used the squared distance for $dist1[\bullet]$ and the Manhattan distance weighted with the inverse median absolute deviation of each feature for $dist2[\bullet]$. In practice, they solved this optimization problem by finding a solution for the counterfactual point for a range of different values of λ and then choosing the largest λ for which the proximity to the desired point was acceptable. This method is only practical if the model output is a continuous function of the input and we can calculate the derivatives of the model output with respect to the input efficiently.

The above formulation has two main drawbacks that were addressed by Dandl *et al.* (2018). First, we would ideally like counterfactual examples where only a small number of the input features have changed (figure 5); this is both easier to understand and more practical in terms of taking remedial action. To this end, we can modify the function $dist2$ to encourage sparsity in the changes.

Second, we want to ensure that the counterfactual example falls in a plausible region of input space. Returning to the loan example, the decision could be partly made based on two different credit ratings, but these might be highly correlated. Consequently, suggesting a change where one remains low, but the other increases is not helpful as this is not realizable in practice. To this end, Dandl *et al.* (2018) proposed adding a second term that penalizes the counterfactual example if it is far from the training points. A further important modification was made by McGragh *et al.* (2018) who allow the user to specify a weight for each input dimension that effectively penalizes changes more or less. This can be used to discourage finding counter-factual explanations where the change to the input are not realisable. For example, proposing a change in an input variable that encodes an individuals age is not helpful as this cannot be changed.

A drawback of counterfactual explanations is that there may be many possible ways to modify the model output by perturbing the features locally and it's not clear which is most useful. Moreover, since most approaches are based on optimization of a non-linear function, it's not possible to ensure that we have find the local minimum; even if we fail to find any counterfactual examples within a predefined distance from the original, this does not mean that they do not exist.

LIME



For image data, we might explain the output of a classifier based on a convolutional network by approximating it locally with a weighted sum of the contributions of different superpixels. We first divide the image into superpixels, and then perturb the image multiple times by setting different combinations of these superpixels to be uniform and gray. These images are passed through the classifier and we store the output probability of the top-rated class. Then we build a sparse linear model that predicts this probability from the presence or absence of each superpixel (figure 7).

Figure 7. LIME explanations for image classification. a) Original image to be explained which was classified as `tree frog'. b) Grouping input features by dividing into superpixels. c-e) Replace randomly selected subsets of superpixels with blank gray regions, run through model, and store model probability. f) Build sparse linear model explaining model probability in terms of presence or absence of superpixels. It appears that the head region is mainly responsible for the classification as tree-frog. g) Repeating this process for the class "billiards" which was also assigned a relatively high probability by the model. Adapted from Ribeiro et al. (2016)

Anchors

Ribeiro *et al.* (2018) proposed *anchors* which are local rules defined on the input values around a given point that we are trying to explain (figure 8). In the simplest case, each rule is a hard threshold on an input value of the form $x_1 < \tau$. The rules are added in a greedy way with the aim being to construct rules where the precision is very high (i.e., when the rule is true, the output is almost always the same as the original point). We cannot exhaustively evaluate every point in the rule region and so this is done by considering the choice of rules as a multi-armed bandit problem. In practice, rules are extended in a greedy manner by adding more constraints on them to increase the precision for a given rule complexity. In a more sophisticated solution, beam search is

Figure 8. Anchors. One way to explain the purple point is to search for simple set of rules that explain the local region. In this case, we can tell the user that 100% of points the where x_1 is between 0.51 and 0.9 and x_2 is between 0.52 and 0.76 are classified as positive.

Ribeiro *et al.* (2018) applied this message to an LSTM model that predicted the sentiment of reviews. They perturbed the inputs by replacing individual tokens (words) with other random words with the same part of speech tag, with a probability that is proportional to their similarity in the embedding space and measure the response of the LSTM to each. In this case their rules just consist of the presence of words, and for the sentence *This movie is not bad*, they retrieve the rule that when the both the words "not" and "bad" are present, the sentence has positive sentiment 95% of the time.

Shapley additive explanations

Shapley additive explanations describe the model output $\boldsymbol{f}[\mathbf{x}_i]$ for a particular input $\mathbf{x}_i$ as an additive sum:

$$f[\mathbf{x}_i] = \psi_0 + \sum_{d=1}^{D} \psi_d \tag{2}$$

of D contributing factors ψ_D associated with the D dimensions of the input. In other words, the change in performance from a baseline ψ_0 is attributed to a sum of changes ψ_d associated with the input dimensions.

Consider the case where there are only two input variables (figure 9), choosing a particular ordering of the input variables and a constructing the explanation piece by piece. So, we might set:

$$\begin{aligned}
\psi_0 &= \mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]\right] \\
\psi_1 &= \mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]|x_1\right] - \left(\mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]\right]\right) \\
\psi_2 &= \mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]|x_1, x_2\right] - \left(\mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]\right] - \mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]|x_1\right]\right).
\end{aligned} \tag{3}$$

The first term contains the expected output without observing the input. The second term is the change to the expected output given that we have only observed the first dimension x_1. The third term gives the additional change to expected output now that we have observed dimensions x_1 and x_2. In each line, the term in brackets is the right hand side from the previous line, which is why these represent changes.

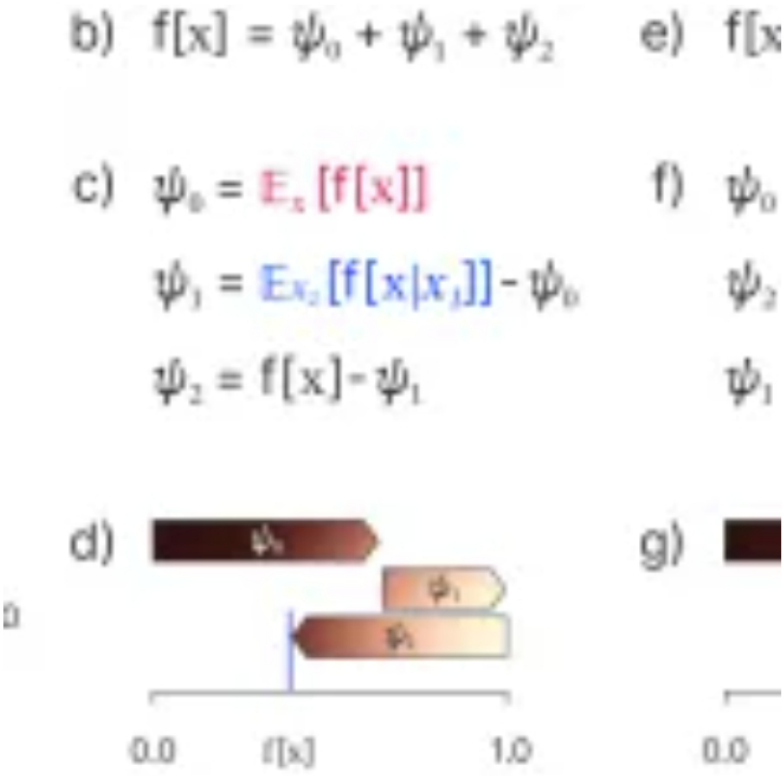


Figure 9. Shapley additive explanations. a) Consider explaining the model output $\mathbf{f}[\mathbf{x}]$ for point $\mathbf{x}$. b) We construct the explanation as a linear combination of three terms. c) The first term is what we know before considering the data at all. This is the average model output (bottom left corner of panel a). The second term is the change due to what we know from observing feature x_1. This is calculated by marginalizing over feature x_2 and reading off the prediction (bottom of panel a). We then subtract the first term in the sum to measure the change that was induced by feature x_1. The third term consists of the remaining change that is needed to get the true model output and is attributable to x_2. d) We can visualize the cumulative changes due to each feature. e-g) If we repeat this procedure, but consider the features in a different order, then we get a different results. Shapley additive explanations take a weighted average of all possible orderings and return the additive terms $\psi_\bullet$ that explain the positive or negative contribution of each feature.

Assuming we could calculate these terms, they would obviously have the form of equation 2. However, the input order was arbitrary. If we took a different ordering of the variables so that x_2 is before x_1, then we would get different values

$$\begin{aligned}
\psi_0 &= \mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]\right] \\
\psi_2 &= \mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]|x_2\right] - \left(\mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]\right]\right)
\end{aligned}$$

$$\psi_1 = \mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]|x_1, x_2\right] - \left(\mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]\right] - \mathbb{E}_{\mathbf{x}}\left[f[\mathbf{x}]|x_2\right]\right). \tag{4}$$

The idea of Shapley additive explanations is to compute the values ψ_d in equation 2 by taking a weighted average of the ψ_i over all possible orderings. If the set of indices is given by $\mathcal{D} = \{1, 2, \ldots, D\}$, then the final Shapley values are

Figure 10. Force diagram for Shapley additive explanations. The final result is explained by an additive sum of a scalars associated with each input feature (PTRATIO, LSTAT, RM etc.). The red features increase the output and the blue feature decrease it. The horizontal size associated with each feature represents the magnitude of change. Via Lundberg and Lee (2017).

Computing SHAP values

When we use this latter assumption, the model can replicate LIME and this is known as kernel SHAP. Recall that LIME fits a linear model based on weighted samples, where the weights are based on the proximity of the sample to the point that we are trying to

Royal Bank of Canada.
Open Source
Publications
Tutorials
ATOM
NOMI
Aiden
RESPECT AI
Partnerships
News
Blog
ML Research Internships
Let's SOLVE it
Fellowships
Locations
© 2025 RBC Borealis
Privacy Policy
Terms of Use
Site map