A Unified Approach to Interpreting Model Predictions

Scott Lundberg and Su-In Lee

Max Nadeau, Max Li, and Xander Davies

February 22, 2023

Outline

1 Additive Explanations
- Overview and relation to
- LIME Desiderata

2 Shapley Values
- Introduction to Shapley
- values Removing features

3 Approximations

4 Experiments

5 Extensions
- Global
- interpretability

Inner

Introduction to Additive Feature Attribution Methods

- This paper unifies 6 previous methods for local interpretability (i.e. explaining the output of a model f on a particular input x) as "additive feature attribution methods", (AFAMs).
- An additive feature attribution method consists of:
  - An enumeration of the features present in x
  - A protocol for "removing some of the features" from x, and therefore a definition of "the input that has no features", here called y.
  - An approximation of $f(x)$ called $g(x')$, and an approximation of $f(y)$ called $g(y')$
  - A partition of $g(x') - g(y')$ among the enumerated features of x, **indicating how important each feature was for the model's output on** x.

The importance of a feature i is denoted ϕ so

LIME as an Additive Feature Attribution Method

- You all remember LIME from the last presentation.
- LIME (for explaining a classification of some image x) is an AFAM.
  - The set of superpixels is the set of features of x
  - We remove a superpixel (i.e. feature) by replacing its pixels with grey. So the image containing no
  - features is all grey. LIME outputs a function g that approximates $f(x)$ and $f(y)$ as $g(x')$ and $g(y')$.
  - g provides a weighting g_i for the importance of each superpixel
    in determining $f(x)$; these serve as the ϕ_i .

DeepLIFT as an Additive Feature Attribution Method

- DeepLIFT is another local interpretability method, proposed in (Shrikumar et al., 2019).
- DeepLIFT (for explaining a classification of some image x) is an AFAM.
  - The set of pixels is the features of x .
  - We pick some “reference value” to serve as y , the image with no features. Removing a feature of x consists of setting a pixel to the value of that pixel in
  - the reference image.
    DeepLIFT doesn’t approximate $f(x)$ and $f(y)$, it just uses $g(x') = f(x)$ and $g(y') = f(y)$.
  - DeepLIFT calculates a value $C_{\Delta x_i \Delta o}$ for each pixel x_i such that $\sum_i C_{\Delta x_i \Delta o} = f(x) - g(y') = f(y)$. Each $C_{\Delta x_i \Delta o}$ represents the importance of x_i to classification f the (x).

Desiderata for Additive Feature Attribution Methods

- They propose three desirable properties for an
- AFAM. Local accuracy: $g(x')=f(x)$. As we saw on the last two slides, DeepLIFT meets this criterion but LIME does not necessarily.
- Consistency: For some input x, let $x\setminus i$ denote “removing feature i from x”. Let z be an alternative input produced by removing some of x’s features. Suppose you have two models, f and f'. If, $\forall z$, $f(z)-f(z\setminus i)\geq f'(z)-f'(z\setminus i)$, then our AFAM should have $\phi_i(f,x)\geq\phi_i(f',x)$.
  - In other words, if including feature i in the input always makes a bigger difference in model f than in model f',
- then the AFAM should give a higher importance ϕ_i for the model f than for f'.

Cooperative games

- Suppose we have a game with d players, where each player can choose whether or not to cooperate.
- Assume a reward function $g : \mathcal{P}([d]) \rightarrow \mathbb{R}$. If $S \subset [d]$ is the set of players that choose to cooperate, then the group receives reward $g(S)$.
- We want to determine how much each player "contributes" to the reward. However, the marginal contribution of player i may depend on which other players have also chosen to cooperate. g does not need to be monotonic!

Shapley values

Maybe we can just take the average of player i's marginal contribution over all subsets. In fact, this calculation gives player i's **Banzhaf power index**:

$$\frac{1}{2^{d-1}} \sum_{S \subset [d] \setminus \{i\}} g(S \sqcup \{i\}) - g(S) \quad (1)$$

The **Shapley value** reweights the marginal contributions based on the size of the subset S:

$$\frac{1}{d} \sum_{j=0}^{d-1} \binom{d-1}{j}^{-1} \sum_{S \subset [d] \setminus \{i\},\ |S|=j} \left[g(S \sqcup \{i\}) - g(S) \right] \quad (2)$$

or, equivalently, $\sum_{\sigma \in S_d} g(\sigma([\sigma(i)])) - g(\sigma([\sigma(i) -$

From games to local interpretability

- Suppose that for some input x, a model produces prediction $f(x)$, and we would like to measure how "important" each feature was for the model prediction. We can treat the features as players in a cooperative game, and ask how much each contributed to the output.
- However, to prompt the model, we need to provide all the input features S. How do we measure what the model would have predicted if it only had access to a subset of the features?
  - In other words, the function $g : \mathcal{P}([d]) \to \mathbb{R}$ is not well-defined.

Previously proposed: separate models

- In "Shapley regression values," for every subset of features $S \subset [d]$, we can train a model f_S that tries to predict the labels.
- The resulting Shapley values are a good metric for how important each feature is for good prediction of the labels.
- However, they do not provide interpretability for the specific model f we were working with, i.e. what f might do without any information about the features in $[d] \subset S$ might be very different than optimal prediction.

Feature ablation

- We want to capture what our *particular* model on f would do if it had no access to features $[d] \subset S$.
- This notion is captured by the expectation of the model output given features S.

$$E[f(x) \mid x_S] \tag{3}$$

- We can approximate this quantity by sampling over the conditional distribution:

$$\frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}), \quad x^{(i)} \sim x|x_S \tag{4}$$

Model Agnostic: Feature independence

- Computing SHAP values requires calculating 2^d differences
- $g(s \sqcup \{i\}) - g(s)$.

  One way to improve this is to assume *features are independent*, $\forall S$

$$E[f(x) \mid x_S] = E_{x_{\bar{S}} \mid x_S}[f(x)] \approx E_{x_{\bar{S}}}[f(x)] \quad (5)$$

- We can then estimate SHAP values via sampling approximations (the Shapley sampling values method) which require fewer than 2^d difference calculations.

  But, still requires lots of computations.

Model Agnostic: Kernel SHAP via LIME

- LIME: The kernel weights ($\pi_{x'}$), loss function (L), and simplicity function (Ω) proposed in LIME aren't consistent with our desired properties (local accuracy & consistency).
- We can adjust these choices to fix that, forming the Shapely kernel:

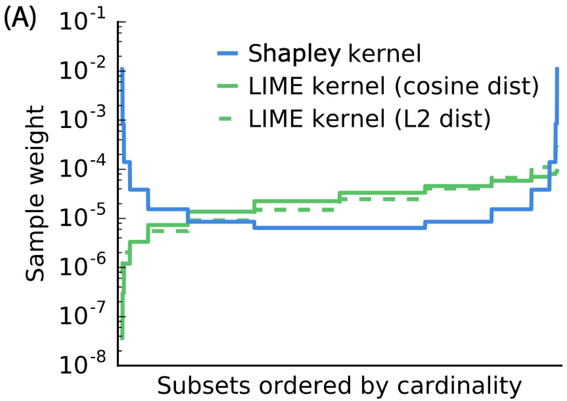
$$\Omega(g) = 0, \tag{6}$$

$$\pi_{x'}(z') = \frac{(M-1)}{(M \text{ choose } |z'|)\,|z'|\,(M-|z'|)}, \tag{7}$$

$$L(f, g, \pi_{x'}) = \sum_{z' \in Z} \left[f\left(h_x(z')\right) - g(z') \right]^2 \pi_{x'}(z'), \tag{8}$$

where $|z'|$ is the number of non-zero

Model Agnostic: Kernel SHAP (cont.)



Model Agnostic: Kernel SHAP (cont.)

- We can consider LIME's feature removing protocol an *approximation* of SHAP values that *assumes f is linear* (eek), so that $E[f(x) \mid x_S] = f(x^*)$, where

$$x_i^* = \begin{cases} x_i & \text{if } i \in S \\ E[x_i] & \text{otherwise} \end{cases} \quad (9)$$

- Since we can solve for g in (8) as a weighted linear regression problem, we have a regression-based, model-agnostic estimation of SHAP values!
- This is more efficient than previously, since we jointly solve for SHAP values.

Model-Specific Approximations: Linear SHAP

- We can do better by looking for model-specific approximations.
- If the model is affine and we assume feature independence, feature i's importance for x is its difference from the mean multiplied by its weight.
- That is, if $f(x) = \sum_{j=1}^{M} w_j x_j + b$, then $\phi_0(f, x) = b$ and $\phi_i(f, x) = w_j (x_j - E[x_j])$

Model-Specific Approximations: Deep SHAP

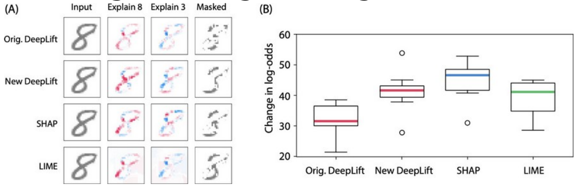
- DeepLift approximates SHAP values assuming that the input features are independent of one another and the deep model is linear, since it:
  1. linearizes the non-linear components of a network ("heuristically chosen")
  2. replaces values with a reference value, which we can consider $E[x]$ (like LIME)
- Currently, this satisfies local accuracy (and missingness), but not consistency.
- We can choose new linearizations which satisfy consistency → Deep SHAP!

User Experiments

- The authors run a experiment in which they tell a story about people playing a cooperative game, and find that human assignments of credit align better with SHAP's assignment than with LIME's or DeepLIFT's.
- This is a very different setting from attribution in neural networks, seemingly selected to make SHAP look good, so this experiment is unimpressive evidence that SHAP aligns with human intuition for NN credit assignment.

Class Difference Experiments

- Using an image of an “8” and an MNIST classifier, the authors identified which pixels (according to SHAP, LIME, and DeepLIFT) are most important for the model’s log-odds (i.e. logit difference) of 8 versus 3.
- Removing the pixels identified by SHAP produced larger changes in log-odds from 8 to 3 than



Shapley values for the whole model

Rather than attributing Shapley values for the model's prediction on a particular value x, we can instead attribute Shapley values for the model's prediction over the entire input distribution. Naively, we can define $g : \mathcal{P}([d]) \to \mathbb{R}$ by $g(S) = E[E[f(x)|x_S]]$, which reduces trivially to $E[f(x)]$ by Adam's law. Instead, to capture the amount of the model's behavior we can explain with only a subset of the features, we need to impose a symmetric loss function, for instance

$$g(S) = \mathrm{Var}(E[f(x) \mid x_S])$$

Methods that do this include SAGE (Covert et al, 2020) and Shapley Effects (Owen, 2014).

Neuron Shapley

We can treat neurons as features (Ghorbani and Zou, 2020). This paper uses zero ablation.
We can also use multi-armed bandit sampling algorithms to estimate Shapley values.