Explaining model predictions with Shapley values + conditional inference trees
R package

Annabelle Redelmeier & Martin Jullum

September 3rd, 2020
Explanation problem

► Suppose you have a black-box model predicts the price of car insurance based on some features.

► How can we explain the prediction of a black-box model to a customer?

<table>
<thead>
<tr>
<th>Features</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>$123 / month</td>
</tr>
<tr>
<td>Gender</td>
<td></td>
</tr>
<tr>
<td>Type of car</td>
<td></td>
</tr>
<tr>
<td># accidents</td>
<td></td>
</tr>
<tr>
<td>Time since car registered</td>
<td></td>
</tr>
</tbody>
</table>
Explanation problem

One way to explain a black-box model is to show how the features contribute to the overall prediction (i.e. $123$).

To calculate these contributions, we can use Shapley values.
Why are explanations important?

► **Engineer/scientist making the model:** Are there problems with the model? Edge cases? Biases?

► **Society:** Is the model fair? Legality?

► **Individual:** Do I trust the model prediction/outcome?

► **Company/group using the model:** Do customers trust me? Can I back the model up?

Peeking Inside the Black-Box:
A Survey on Explainable Artificial Intelligence (XAI)
Adadi, 2018
To put explanations in context

- **Model agnostic**
  - Local explanation:
    - LIME,
    - Shapley values,
    - Explanation Vectors,
    - Counterfactuals explanations,
    - Saliency map
  - Global explanation:
    - Partial dependence plots,
    - Activation maximization,
    - Model distillation,

- **Model specific**
  - Used for any ML model
  - Specific to a model like xgboost or regression

**Explain a specific prediction**

**Understanding the whole logic of the model**

- Specific to a model like xgboost or regression
Shapley values

► Economic game theory in 1953.
  ► Setting: A game with $M$ players cooperating to maximize the total gains of the game.
  ► Goal: Distribute the total gains in a “fair” way:
    ► Axiom 1: Players that contribute nothing get payout $= 0$.
    ► Axiom 2: Two players that contribute the same get equal payout.
    ► Axiom 3: The sum of the payouts $= \text{total gains}$.

Lloyd Shapley (1953) found a unique way to distribute the total gains in such a way that obeys these three axioms.
Shapley values

- We assume player $j$ will contribute possibly **differently** depending on who he or she is cooperating with.
- Suppose player $j$ is cooperating with players in group $S$.
- Then, we define the *marginal contribution of player $j$ with group $S$*:

\[
\nu(S \cup \{j\}) - \nu(S)
\]
Shapley values

Shapley value of player $j = \text{payout for player } j$ defined as

$$\phi_j = \sum_{S \subseteq M \setminus \{j\}} w(S) \left( \nu(S \cup \{j\}) - \nu(S) \right)$$

- $\phi_j$: Shapley value of player $j$
- $\sum_{S \subseteq M \setminus \{j\}}$: Weight function
- $w(S)$: Weight of coalition $S$
- $\nu(S \cup \{j\})$: Value of coalition $S \cup \{j\}$
- $\nu(S)$: Value of coalition $S$
- $M$: The set of all players in the game
Example

- 2 players: $x_1$ and $x_2$
- Then, $x_1$’s Shapley value is:

$$\phi_{x_1} = \sum_{S \subseteq M \setminus \{x_1\}} w(S)(v(S \cup \{x_1\}) - v(S))$$

$$= w_1[v(\{x_2, x_1\}) - v(\{x_2\})] + w_2[v(\{\emptyset, x_1\}) - v(\{\emptyset\})]$$

$$P(M \setminus \{x_1\}) = \{\{x_2\}, \{\emptyset\}\}$$
How does this translate to ML?

► Cooperative game $\rightarrow$ **individual**.

► Players $\rightarrow$ feature values of the **given individual**.

► **Total gains** $\rightarrow$ predicted value of the **given individual**.

► Given individual = **Janet**.

► Feature values are:
  - Age = 55 years
  - Gender = woman
  - Type of car = Buick
  - # Accidents = 3
  - Time since car registered = 3.2 years

► Predicted value = $\$123$. 
How does this translate to ML?

\[ \phi_j = \sum_{S \subseteq M \setminus \{j\}} w(S) (v(S \cup \{j\}) - v(S)) \]

Contribution of feature \( j \)
Set of all features in the ML model
Subset of features
Contribution of feature set \( S \)

We have access to almost all we need. We’re only missing \( v(S) \)…

\[ v(S) = \mathbb{E}[f(x) | x_S = x^*_S] \]

ML model
Set of all features
Here we condition on the features in \( S \) equal to the feature values of the individual (Janet)

\( S = \{ \text{Age, gender} \} \)
\( \tilde{S} = \{ \text{Type of car, # Accidents, Time since registration, gender} \} \)
Problems with Shapley values in ML

\[ \phi_j = \sum_{S \subseteq M \setminus \{j\}} w(S) (v(S \cup \{j\}) - v(S)) \]

1. Calculating \( \phi_j \) when \( M \) is large.

2. Estimating \( v(S) \) when \( M \) is large.

- If \( M = 10 \), there are \( 2^{10} = 1024 \) combinations.
- If \( M = 30 \), there are \( 2^{30} = 1.1 \) million!

Problem solved with "kernelSHAP"
Computing $\nu(S)$

$\nu(S) = \mathbb{E}[f(x)|x_S = x_S^*]$ is rarely known.

- (Lundberg & Lee, 2017a) assume the features are independent.
- $\nu(S)$ can then be estimated by sampling from the full data set and calculating an average.

- (Aas et al., 2019) estimate $\nu(S)$ parametrically and non-parametrically in various methods.
Computing \( v(S) \)

- (Aas et al., 2019)'s methods to estimate

1. **Empirical method:**
   1. Calculate a distance between the set of features explained and every training instance
      \[ D_S(x^*, x^i) \]
   2. Use this \( D \) to calculate a weight
      \[ w_S(x^*, x^i) = \exp(-\frac{D^2}{2\sigma^2}) \]
   3. Estimate

\[
v(S) \approx \frac{\sum w_S f(x^*_S, x^*)}{\sum w_S (x^*, x^k)}
\]
Computing $v(S)$

- (Aas et al., 2019)’s methods to estimate

$$v(S) = \mathbb{E}[f(x) | x_S = x_S^*]$$

2. **Gaussian method:**
   1. Assume the features are jointly Gaussian. This means we have an explicit form for the conditional distribution
   $$p(x_S | x_S = x_S^*).$$
   2. Estimate the conditional mean and covariance matrix.
   3. Sample $k$ times from this conditional Gaussian distribution with the estimated mean and covariance matrix.
   4. Estimate
   $$v(S) \approx \frac{1}{K} \sum_{k=1}^{K} f(x_S^k, x_S^*).$$
Computing $\nu(S)$

The problem is that (Aas et al., 2019)'s methods assume the features are continuously distributed.

We extend (Aas et al., 2019)'s method to handle mixed (i.e., continuous, categorical, ordinal) features using conditional inference trees.
**ctree: Conditional Inference Trees**

► **ctree** (Hothorn et al., 2006) is a tree fitting statistical model like CART and C4.5.
  - What is so great about a tree?

► **Differences:**
  - Solves for the splitting feature and split point using **hypothesis tests**.

[Diagram of a tree structure showing decisions based on gender, age, and dog preference.]
ctree: Conditional Inference Trees

How do we build a ctree?

1. Test each feature with the partial hypothesis test:
   \[ H_0^j: F(Y|X_j) = F(Y) \]

2. If the global \( p \)-value is < \( \alpha \), choose the feature that is the least dependent of \( Y \).

3. Find a split point based on this feature (and your favourite splitting algorithm).

Can handle *multivariate responses!*
To estimate $v(S) = \mathbb{E}[f(x)|x_S = x_s^*]$: 

1. We fit a ctree where the *tree features* are the features in $S$ and the *tree response* are the features **not** in $S$. 

   $\bar{S} \sim f(S)$

   ![Diagram of ctree: Conditional Inference Trees]

   **Example:**

   $S = \{\text{Age, gender}\}$
   $\bar{S} = \{\text{Type of car, # Accidents, Time since registration}\}$
2. Given the tree, we find the leaf node based on Janet’s features values:

\[ S = \{ \text{Age}=55, \text{gender}=\text{woman} \} \]

Other train observations in the leaf:

<table>
<thead>
<tr>
<th>gender</th>
<th>age</th>
<th>car</th>
<th>accidents</th>
<th>Time since registration</th>
</tr>
</thead>
<tbody>
<tr>
<td>woman</td>
<td>55</td>
<td>Volvo</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>woman</td>
<td>54</td>
<td>Tesla</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>woman</td>
<td>52</td>
<td>BMW</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>woman</td>
<td>60</td>
<td>BMW</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>woman</td>
<td>55</td>
<td>Nissan</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
ctree: Conditional Inference Trees

3. We sample from the leaf and use these samples to estimate:

\[ v(S) \approx \frac{1}{K} \sum_{k=1}^{K} f(x_S^k, x_S^*) \]

ML model

<table>
<thead>
<tr>
<th>gender</th>
<th>age</th>
<th>car</th>
<th>accidents</th>
</tr>
</thead>
<tbody>
<tr>
<td>woman</td>
<td>55</td>
<td>Volvo</td>
<td>0</td>
</tr>
<tr>
<td>woman</td>
<td>54</td>
<td>Tesla</td>
<td>2</td>
</tr>
<tr>
<td>woman</td>
<td>52</td>
<td>BMW</td>
<td>1</td>
</tr>
<tr>
<td>woman</td>
<td>60</td>
<td>BMW</td>
<td>3</td>
</tr>
<tr>
<td>woman</td>
<td>55</td>
<td>Nissan</td>
<td>0</td>
</tr>
</tbody>
</table>
Simulation studies

1. Simulate \textit{dependent} categorical data to act as our features.

2. Define linear response model.

   \[
   y_i = \alpha + \sum_{j=1}^{M} \sum_{l=2}^{L} \beta_{jl} \mathbf{1}(x_{ij} = l) + \varepsilon_i,
   \]

   Fixed coefficients

   Indicator function

   \[
   \mathbf{1}(x_{ij} = l)
   \]

   Normal\((0, 1)\)

3. Convert categorical data to numerical data using \textbf{one-hot encoding} so that we can use methods in (Aas et al., 2019).
Simulation studies

4. Calculate the true Shapley values using the true expectation:

\[ \nu(S) = \mathbb{E}[f(x)|x_S = x^*_S] \]

5. Evaluate the methods using mean absolute error (MAE):

\[
 MAE(\text{method } q) = \frac{1}{M} \sum_{j=1}^{\# \text{ features}} \sum_{i=1}^{\# \text{ test observations}} p(x_i) | \phi_{j,\text{true}}(x_i) - \phi_{j,q}(x_i) |
\]
# Simulation study 1

(Aas et al., 2019)

<table>
<thead>
<tr>
<th># features</th>
<th># categories</th>
<th>Method</th>
<th>0</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>empirical</td>
<td>0.0307</td>
<td>0.0277</td>
<td>0.0358</td>
<td>0.0372</td>
<td>0.0419</td>
<td>0.0430</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gaussian</td>
<td>0.0307</td>
<td>0.0236</td>
<td>0.0354</td>
<td>0.0327</td>
<td>0.0318</td>
<td>0.0383</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ctree</td>
<td><strong>0.0274</strong></td>
<td><strong>0.0191</strong></td>
<td><strong>0.0302</strong></td>
<td><strong>0.0310</strong></td>
<td><strong>0.0244</strong></td>
<td><strong>0.0259</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>independence</td>
<td><strong>0.0274</strong></td>
<td><strong>0.0191</strong></td>
<td>0.0482</td>
<td>0.0777</td>
<td><strong>0.1546</strong></td>
<td><strong>0.2062</strong></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>empirical</td>
<td>0.0312</td>
<td>0.0385</td>
<td>0.0330</td>
<td>0.0435</td>
<td>0.0480</td>
<td>0.0410</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gaussian</td>
<td>0.0312</td>
<td>0.0385</td>
<td>0.0330</td>
<td>0.0435</td>
<td>0.0480</td>
<td>0.0410</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ctree</td>
<td><strong>0.0223</strong></td>
<td><strong>0.0414</strong></td>
<td><strong>0.0387</strong></td>
<td><strong>0.0453</strong></td>
<td><strong>0.0329</strong></td>
<td><strong>0.0253</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>independence</td>
<td><strong>0.0223</strong></td>
<td><strong>0.0355</strong></td>
<td>0.0961</td>
<td>0.1515</td>
<td>0.2460</td>
<td>0.2848</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>ctree</td>
<td>0.0169</td>
<td><strong>0.0505</strong></td>
<td><strong>0.0617</strong></td>
<td><strong>0.0607</strong></td>
<td><strong>0.0627</strong></td>
<td><strong>0.0706</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>independence</td>
<td><strong>0.0153</strong></td>
<td>0.0544</td>
<td>0.1593</td>
<td>0.2180</td>
<td><strong>0.3017</strong></td>
<td><strong>0.3412</strong></td>
</tr>
</tbody>
</table>

(Lundberg & Lee, 2017a)

Mean absolute error
## Computation time

<table>
<thead>
<tr>
<th># features</th>
<th># categories</th>
<th>Method</th>
<th>Mean time per test obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>empirical</td>
<td>0.086</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gaussian</td>
<td><strong>13.295</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ctree</td>
<td>0.040</td>
</tr>
<tr>
<td></td>
<td></td>
<td>independence</td>
<td>0.013</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>empirical</td>
<td>0.293</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gaussian</td>
<td><strong>12.983</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ctree</td>
<td>0.052</td>
</tr>
<tr>
<td></td>
<td></td>
<td>independence</td>
<td>0.012</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>ctree</td>
<td>6.718</td>
</tr>
<tr>
<td></td>
<td></td>
<td>independence</td>
<td>2.066</td>
</tr>
</tbody>
</table>

**Note:** Gaussian is slow because
- It has to call “predict” function more than empirical + ctree
- Matrix inversions and sampling
## Simulation study 2

(Aas et al., 2019)

<table>
<thead>
<tr>
<th>$M$</th>
<th>$L$</th>
<th>Method</th>
<th>0</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>empirical</td>
<td>0.0853</td>
<td>0.0852</td>
<td>0.0898</td>
<td>0.0913</td>
<td><strong>0.0973</strong></td>
<td>0.1027</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gaussian (100)</td>
<td>0.0570</td>
<td><strong>0.0586</strong></td>
<td><strong>0.0664</strong></td>
<td><strong>0.0662</strong></td>
<td>0.1544</td>
<td>0.2417</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ctree</td>
<td><strong>0.0093</strong></td>
<td>0.0848</td>
<td>0.1073</td>
<td>0.1060</td>
<td><strong>0.0977</strong></td>
<td><strong>0.0917</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>independence</td>
<td><strong>0.0093</strong></td>
<td>0.0790</td>
<td><strong>0.2178</strong></td>
<td>0.3520</td>
<td>0.5524</td>
<td>0.6505</td>
</tr>
</tbody>
</table>

(Lundberg & Lee, 2017a)

### Mean absolute error

### Dependence of features

(Lundberg & Lee, 2017a)
## Computation time

Over all dependence, $\rho$, and test observations

<table>
<thead>
<tr>
<th>$M$</th>
<th>$L$</th>
<th>$T$</th>
<th>Method</th>
<th>Mean time per test obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>500</td>
<td>empirical</td>
<td>0.758</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gaussian (100)</td>
<td>5.914</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ctree</td>
<td>0.082</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>independence</td>
<td>0.057</td>
</tr>
</tbody>
</table>
Limitations

- The ctree/Gaussian/empirical methods cannot be used for more than 23 (30?) features due to computational problems (regardless of one-hot encoding…). What can we do to improve this?
  - GroupSHAP?
  - New approach to sampling: (Grah and Thouvenot, 2020)\(^1\)?

---

\(^1\)A Projected Stochastic Gradient Algorithm for Estimating Shapley Value Applied in Attribute Importance