Powershap

Chen Yen-Ting

Department of Information Management and Finance

National Yang Ming Chiao Tung University

Outline

- Introduction
- Formulation to feature attribution.
- Powershap

Interpreting Model Predictions

- A local methods designed to explain a prediction $f(x)$, x: a single input.
- f: be the original prediction model,
- Find g: the explanation model.
- Simplified inputs x', map to the original inputs through a mapping function $x = h_x(x')$.
- Local methods try to ensure $g(z') \approx f(h_x(z'))$ whenever $z' \approx x'$.

Additive feature attribution methods

- An explanation model that is a linear function of binary variables:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i', \quad \text{where} z_i' \in \{0,1\} \quad \ldots(1).$$

- Calculate attribute ϕ_i to each feature, and summing the effects of all feature attributions approximates the output f (x) of the original model.
- Many current methods match equation (1), several of which are discussed below.
- For example: LIME, DeepLIFT, Shapley regression values.

Shapley Value for Feature importance

$$\phi_i(v) = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_s(x_s) \right]$$

- $\phi_i(v)$: Feature importance for feature i.
- f: predict model
- F: is the set of all features.
- S: subset of F.

Powershap

- Powershap builds upon the idea that a known random feature should have, on average, a lower impact on the predictions than an informative feature.
- The powershap algorithm consists of two components
    - Explain component
    - Core powershap component
- To hyper-parameter tuning, there are Automatic Mode.
    - Powershap analysis function
    - Automatic Powershap algorithm version

Powershap Explain algorithm…(1)

Algorithm 1: Powershap Explain algorithm

Function *Explain(*$I \leftarrow$ *Iterations,* $M \leftarrow$ *Model,* $\mathbf{D}^{n \times m} \leftarrow$ *Data,* $rs \leftarrow$ *Random seed)*

- $\mathbf{powershap}_{values} \leftarrow$size $[I, m+1]$
- **for** $i \leftarrow 1, 2, \ldots, I$ **do**
  - $RS \leftarrow i + rs$
  - $D^{n}_{random} \leftarrow \text{RandomUniform}(RS) \in [-1, 1]$ size n
  - $\mathbf{D}^{n \times m+1} \leftarrow \mathbf{D}^{n \times m} \cup D^{n}_{random}$
  - $\mathbf{D}^{0.8n \times m+1}_{train}, \mathbf{D}^{0.2n \times m+1}_{val} \leftarrow$ split $\mathbf{D}$
  - $M \leftarrow \text{Fit } M(\mathbf{D}_{train})$
  - $\mathbf{S}_{values} \leftarrow \text{SHAP}(M, \mathbf{D}_{val})$
  - $\mathbf{S}_{values} \leftarrow |\mathbf{S}_{values}|$
  - **for** $j \leftarrow 1, 2, \ldots, m+1$ **do**
    - $\mathbf{powershap}_{values}[i][j] \leftarrow \mu(\mathbf{S}_{values}[\ldots][j])$
  - **end**
- **end**
- **return** $\mathbf{powershap}_{values}$

For each iteration:

1. Generator a random feature from $U(-1,1)$.
2. Using 80 % data as training set, fit a model.
3. For every sample point calculate Shapley Value on validation set.
4. Take the absolute value of every Shapley value.
5. For every feature, calculate the mean of the absolute Shapley values.

Parameters to be determined:

I:# of iterations, M: machine learning algorithm.

Output:

An $i \times (m+1)$ matrix PS, where PS_{ij} means mean of absolute Shapley Value of j-th columns in i-th iteration.

Powershap core algorithm…(2)

Algorithm 2: Powershap core algorithm

Function *Powershap* ($I \leftarrow$ *Iterations*, $M \leftarrow$ *Model*, $\mathbf{F}_{set} \leftarrow F_1, \ldots, F_m$, $\mathbf{D} \leftarrow$ *Data size* $[n, m]$, $\alpha \leftarrow$ *required p-value*)

- $\mathbf{powershap}_{values} \leftarrow \mathbf{Explain}(I, M, \mathbf{D})$
- $S_{random} \leftarrow \mu(\mathbf{powershap}_{values}[...][m+1])$
- $\mathbf{P}^m \leftarrow$ initialize
- **for** $j \leftarrow 1, 2, \ldots, m$ **do**
  - $\mathbf{P}[j] \leftarrow Percentile(\mathbf{powershap}_{values}[...][j], S_{random})$
- **end**
- **return** $\{F_i \mid \forall\, i : \mathbf{P}[i] < \alpha\}$

1. Do *Algorithm* 1
2. Calculate the mean for random feature column, call the mean value S_r.
3. For every column of PS compare all row to S_r, and find the proportion of values that are less than S_r.
4. Return the corresponding feature names for columns where the proportion of values less than S_r is less than α.

Parameters to be determined:

I:# of iterations, M: machine learning algorithm, α: required p-value.

Output:

Some features are selected.

Iterations: 1

x_1	x_2	x_3	x_4	x_5	x_r
7.34	8.12	3.15	2.48	1.56	2.19
6.91	7.43	1.89	2.74	1.65	3.21
8.24	9.31	2.11	3.48	2.37	2.01
7.89	8.76	2.25	3.02	1.95	2.68
6.78	7.55	1.48	2.89	1.73	2.56
7.96	8.45	2.31	3.14	2.18	2.74
6.87	7.67	1.92	2.36	1.52	3.17
8.02	9.14	2.45	3.25	2.09	2.33
7.73	8.54	2.18	2.97	1.86	2.67
6.59	7.32	1.74	2.49	1.58	2.39
7.48	8.23	2.05	2.84	1.72	2.65
8.11	9.01	2.37	3.36	2.11	2.43
7.56	8.42	2.09	3.08	1.94	2.53
6.83	7.61	1.83	2.57	1.67	2.84
8.05	9.22	2.46	3.18	2.19	2.38
7.67	8.71	2.29	3.04	1.98	2.62
6.75	7.48	1.76	2.64	1.59	2.47
8.09	9.19	2.41	3.22	2.14	2.36
7.81	8.63	2.22	3.10	1.89	2.59
6.64	7.35	1.71	2.53	1.55	2.41

• • •

Iterations: 10

x_1	x_2	x_3	x_4	x_5	x_r
8.21	9.05	3.14	2.58	1.89	2.33
7.34	8.22	2.07	2.91	1.75	2.64
8.56	9.48	3.02	2.79	1.99	2.45
7.89	8.65	2.44	2.67	1.84	2.52
6.98	7.92	2.15	2.46	1.63	2.31
8.14	9.12	3.26	2.85	1.94	2.49
7.72	8.36	2.34	2.63	1.76	2.58
8.23	9.18	3.18	2.91	2.08	2.66
7.95	8.71	2.55	2.74	1.87	2.42
6.87	7.78	2.01	2.39	1.59	2.27
7.56	8.49	2.29	2.66	1.83	2.37
8.34	9.27	3.08	2.96	2.14	2.51
7.68	8.59	2.43	2.68	1.79	2.46
6.92	7.83	2.11	2.45	1.61	2.32
8.17	9.14	3.22	2.88	1.96	2.53
7.81	8.67	2.49	2.71	1.85	2.40
6.79	7.74	2.07	2.43	1.58	2.25
8.29	9.22	3.14	2.90	2.12	2.48
7.93	8.63	2.51	2.72	1.91	2.36
6.84	7.79	2.03	2.41	1.57	2.29

	x_1	x_2	x_3	x_4	x_5	x_r
1	7.47	8.31	2.13	2.92	1.86	2.56
10	7.71	8.59	2.58	2.69	1.84	2.43
M						2.82

Discussion to Algorithm (1) and (2)

- In Algorithm (1) we calculate Shapley value of every features for all sample points.
- In Algorithm (2), we provide a method to find features that have more attribution to the prediction than a random feature.
- In Algorithm (1) and (2), there are two hyper-parameter needed to be determined
  - I: # of iterations.
  - α: required p-value.

Trade-off between runtime and quality

- Consider a null hypothesis:

  H_0: random feature do not have fewer attribution than informative features.

- In fact, we know H_0 is wrong.
- Trade-off:
  - There should be enough iterations to avoid Type II error for a given α.
  - Adding iterations increases the time complexity.
- In Algorithm (3) and (4) we provide a method to optimizes I, by statistical power, hence this algorithm named Powershap.

Prerequisite to understand Algorithm 3

- Recall that, $1-\beta$: statistical power, β: the probability of Type II error.
- We use $\alpha(x) = F_{H_0}(x)$ to calculate the probability reject H_0 in the current data.
- If the data in the statistical test is small, it is possible to have a very low α but a large β.
- For a given α, we hope β should be as close to 0 as possible to avoid any Type II error.
- We can use $Power(\alpha) = F_{H_1}(F_{H_0}^{-1}(\alpha))$ to calculate power $(1-\beta)$.
- But we do not know the cumulative distribution under H_1.

Optimize Iteration times

- Powershap mapping the F_{H_0}, F_{H_1} to two student-t distributions.
- F_{H_0}: t distribution with degree of freedom $I-1$.
- F_{H_1} : non-central t distribution with degree of freedom $I-1$, and $\mu = -d$.

  Where $d = EffecSize(s_1, s_2) = \frac{\mu(s_1) - \mu(s_2)}{PooledStd(s_1, s_2)}$, and

  $$PooledStd(s_1, s_2) = \frac{\sqrt{(\sigma^2(s_1) + \sigma^2(s_2)}}{2}$$

- And we calculate statistical power by

  $$TTestPower(\alpha, I, d_{calc}) = F_{NCT}(F_{CT}^{-1}(\alpha,\ k = I-1),\ k = I-1,\ -d)$$

- Then we can use some numerical methods to find I.

Powershap analysis function…(3)

Algorithm 3: Powershap analysis function

Function *Analysis*($\alpha \leftarrow$ *required p-value*, $\beta \leftarrow$ *required power*, **powershap**$_{values}$)

- $\mathbf{S}_{random} \leftarrow$ **powershap**$_{values}[...][m+1]$
- $\mathbf{P} \leftarrow$ size $[m]$
- $\mathbf{N}_{required} \leftarrow$ size $[m]$
- **for** $j \leftarrow 1, 2, \ldots, m$ **do**
  - $\mathbf{S}_i \leftarrow$ **powershap**$_{values}[...][j]$
  - $\mathbf{P}[j] \leftarrow Percentile(\mathbf{S}_i, \mu(S_{random}))$
  - effectsize $\leftarrow$ EffectSize($\mathbf{S}_i$, $\mathbf{S}_{random}$)
  - $\mathbf{N}_{required} \leftarrow$ **SolveTTestPower**(effectsize, α, β)
- **end**
- **return** $\mathbf{P}$, $\mathbf{N}_{required}$

1. For every column of PS compare all row to S_r, and find the proportion of values that are less than S_r.
2. For every column of PS calculate d with random feature column.
3. For every column of PS calculate I with random feature column.

Parameters to be determined:

α: required p-value, β: for required power, PS

Output:

$P : 1 \times M$, from step 2., $N : 1 \times M$ from step 3.

Automatic Powershap algorithm version

Algorithm 4: Automatic Powershap algorithm version

Function *Powershap* ($M \leftarrow$ *Model*, $\mathbf{F}_{set} \leftarrow F_1, \ldots, F_m$, $\mathbf{D}^{n \times m} \leftarrow$ *Data*, $\alpha \leftarrow$ *required p-value*, $\beta \leftarrow$ *required power*)

- $\mathbf{powershap}_{values} \leftarrow$ **Explain**($I \leftarrow 10$, M, $\mathbf{D}$, $rs \leftarrow 0$)
- $\mathbf{P}, \mathbf{N}_{required} \leftarrow$ **Analysis**(α, β, $\mathbf{powershap}_{values}$)
- $I_{max} \leftarrow \text{ceil}(\mathbf{N}_{required}[\text{MaxArg}(\mathbf{P} < \alpha)])$
- $I_{old} \leftarrow 10$
- **while** $I_{max} > I_{old}$ **do**
  - **if** $I_{max} - I_{old} > 10$ **then**
    - $\mathbf{auto}_{values} \leftarrow$ **Explain**($I \leftarrow 10$, M, $\mathbf{D}$, $rs \leftarrow 0$)
    - $I_{old} \leftarrow I_{old} + 10$
  - **else**
    - $\mathbf{auto}_{values} \leftarrow$ **Explain**($I \leftarrow I_{max} - I_{old}$, M, $\mathbf{D}$, $rs \leftarrow 0$)
    - $I_{old} \leftarrow I_{max}$
  - **end**
  - $\mathbf{powershap}_{values} \leftarrow \mathbf{powershap}_{values} \cup \mathbf{auto}_{values}$
  - $\mathbf{P}, \mathbf{N}_{required} \leftarrow$ **Analysis**(α, β, $\mathbf{powershap}_{values}$)
  - $I_{max} \leftarrow \text{ceil}(Max(\mathbf{N}_{\mathbf{required}}[i, \forall i : \mathbf{P}[i] < \alpha]))$
- **end**
- **return** $[F_i, \forall\, i : \mathbf{P}[i] < \alpha]$

1. Do *Algorithm* 1 with $I = 10$.
2. Do *Algorithm 3*.
3. Searches for the largest required number of iterations $I_{\max}$ of all tested features having a p -value below the threshold α.
4. Do *Algorithm* 1 with remaining iterations. And renew $I_{\max}$ until # of iterations achieve $I_{\max}$.
5. Return the corresponding feature names for columns where the proportion of values less than S_r is less than α.

Parameters to be determined:

α: required p-value, β: for required power, M: machine learning algorithm,

Output:

Some features are selected.

Parameter setting in my research

- M: XGBoost
- α: 0.01
- β: 0.01